THE TEXTBOOK OF

COMPUTER SCIENCE

For Grade 10

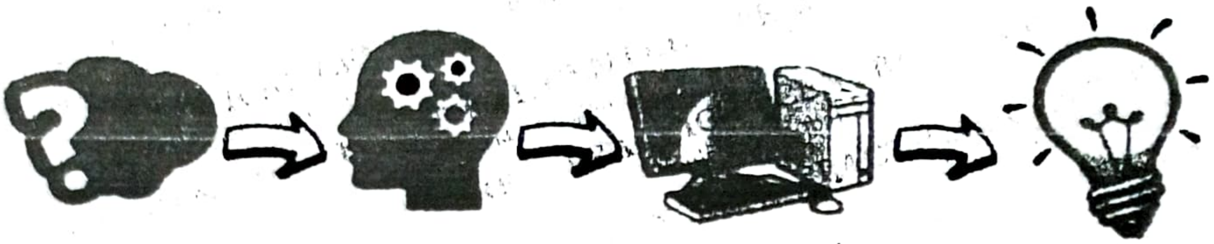
SINDH TEXTBOOK BOARD, JAMSHORO

PROBLEM SOLVING AND ALGORITHM DESIGNING 1

Q.1 WHAT IS PROBLEM SOLVING?

PROBLEM SOLVING:-

Problem solving is the process of finding solutions of difficult or complex issues. It is the process by which any kind of problem is solved.



Solving problems is the core feature of computers. A computer is not intelligent. It cannot analyze a problem and come up with a solution. A human (the programmer) must analyze the problem, develop the instructions for solving the problem, and then make the computer carry out the instructions. The major responsibility of a programmer is to provide the solution of the problems by using computers.

Q.2 WHAT IS PROBLEM?

PROBLEM:-

A problem is a situation that is unsatisfactory and causes difficulties for people. It can be a task, a situation or any other thing. In simple language, a problem is a question which requires an answer or a solution.

Q3. DESCRIBE THE STEPS INVOLVED IN PROBLEM SOLVING.

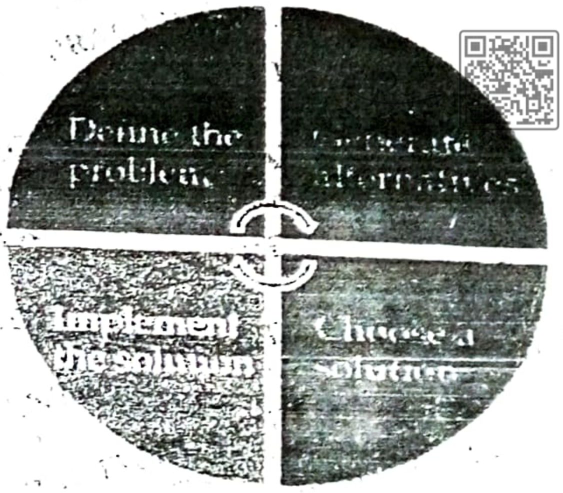
PROBLEM SOLVING PROCESS:

Problem solving is a step-by-step process. Given below, are four basic steps involved in finding a solution for a problem:

1. Define the problem
2. Generate alternative solutions
3. Evaluate and select an alternative
4. Implement and follow up on the solution

1. DEFINE THE PROBLEM:

In problem solving process, the first step is defining and identifying the problem. It is the most difficult and the most important of all the steps. It



PROBLEM SOLVING

involves diagnosing the situation so that the focus should be on the real problem and not on its symptoms. During this first stage of problem solving, it is important to describe the problem. A well-described problem will also help others to understand the problem.

2. GENERATE ALTERNATIVE SOLUTIONS:

For any problem, there are more solutions to it than one that is thought of first. Postpone the selection of one solution, until several problem-solving alternatives have been proposed. Considering multiple alternatives can significantly enhance the value of your ideal solution. So, it is best to develop a list of all feasible solutions that can be assessed and decide which one will be the best for the particular problem. Thinking and team problem-solving techniques are both useful tools at this stage of problem solving.

3. EVALUATE AND SELECT AN ALTERNATIVE:

Many alternative solutions to the problem should be generated before final evaluation .A common mistake in problem solving is that alternatives are evaluated as they are proposed ,so the first acceptable solution is chosen, we may miss the chances for learning something new which is required for real improvements in the problem-solving process.

4. IMPLEMENT AND FOLLOW UP ON THE SOLUTION:

The plan for the best solution also includes planning on what happens next if something goes wrong with the solution, if it does not work out the way it was required. When the best solution is implemented, it is important to track and measure the result to be able to answer questions such as: Did it work? Was this a good solution? Did we learn something here in the implementation that we could apply to other potential problems?

When choosing most appropriate solution, the problem-solver should consider the possible impacts of that solution. For example, will this solution solve the current problem without creating new ones or if this solution is acceptable by everyone involved in this situation or if the solution is within the budget and achievable within a given time.

Q.4 WHAT ARE THE ADVANTAGES OF DEVELOPING ALGORITHMS?

ADVANTAGES OF ALGORITHMS:

1. It is a step-wise representation of a solution to a given problem, which makes it easy to understand.
2. An algorithm uses a definite procedure.
3. It is not dependent on any programming language, so it is easy to understand for anyone even without programming knowledge.
4. Every step in an algorithm has its own logical sequence so it is easy to debug.
5. By using algorithm, the problem is broken down into smaller pieces or steps hence, it is easier for the programmer to convert it into an actual program.

Q.5 EXPLAIN THE CRITERIA FOR MEASURING EFFICIENCY OF AN ALGORITHM.

MEASURING EFFICIENCY OF AN ALGORITHM:

The efficiency of an algorithm is the properties which relate the algorithm to the amount of computational resources used in it. An algorithm must be analyzed to determine its resource usage (e.g., time, memory and storage space).

For maximum efficiency, we wish to minimize resource usage. However, the various resources cannot be compared directly, so which of two algorithms is considered to be more efficient often depends on which measure of efficiency is considered the most important, e.g. the requirement for high speed, minimum memory usage is considered. There may be several algorithms for solving a particular problem. How fast a problem can be solved, depends on efficient algorithm.

Therefore, analysis of algorithms is very important to compare algorithms and conclude which one is better than the other.

Analyzing the efficiency of algorithms means comparing the efficiency of different methods of the solutions of a problem.

Q.6 WHAT IS ALGORITHM AND WHAT IS THE ROLE OF ALGORITHM IN PROBLEM SOLVING?

ALGORITHM:-

Algorithm means method, procedure, technique or plan. Algorithm is a step by step problem solving method that is easy to understand and follow .It is a set of steps that clearly defined a sequence of operations to solve a problem.

ROLE OF ALGORITHM IN A PROBLEM SOLVING:-

Algorithm plays an important role in computer programming. Computer programming is the process of taking an algorithm and coding it in a programming language. Formulating an algorithm is the first step for developing a computer program.

Q.7 WHAT ARE THE DISADVANTAGES OF ALGORITHM?

DISADVANTAGES OF ALGORITHMS:

1. Algorithms is Time consuming.
2. Branching and Looping is difficult to show in Algorithms.
3. Big tasks are difficult to put in Algorithms.

Q.8 WHAT ARE THE CHARACTERISTICS OR QUALITIES OF ALGORITHM?

CHARACTERISTICS OF ALGORITHMS:

Precision – steps are precisely stated (defined).

Uniqueness – results of each step are uniquely defined and depend only on the input and the result of the preceding steps.

Finiteness – algorithm stops after a finite number of instructions are executed.

Input – algorithm receives input.

Output – algorithm produces output.

Generality – algorithm applies to a set of inputs.

Q.9 DEFINE THE GUIDELINES FOR DEVELOPING AN ALGORITHM.

GUIDELINES FOR DEVELOPING AN ALGORITHM:

Following guidelines must be followed while developing an algorithm:

1. An algorithm will be enclosed by START (or BEGIN) and STOP (or END).
2. To accept data from the user, generally used statements are INPUT, READ, GET or OBTAIN.

3. To display result or any message, generally used statements are PRINT, DISPLAY or WRITE.
4. Generally, COMPUTE or CALCULATE is used while describing mathematical expressions and based on the situation, relevant operators can be used.

Q.10 WRITE THE ALGORITHM FOR THE PROBLEM TO MAKE A CUP OF TEA.

ALGORITHM: Making a cup of tea

- **Step 1:** Start
- **Step 2:** Place fresh water in a pot or a kettle.
- **Step 3:** Boil the water.
- **Step 4:** Put black tea leaves in that pot.
- **Step 5:** After that, add some milk into that pot.
- **Step 6:** Add for some sugar.
- **Step 7:** Boil for some time.
- **Step 8:** Stop

Q.11 WRITE THE ALGORITHM TO FIND THE SUM OF TWO NUMBERS.

ALGORITHM: Sum of two numbers

- **Step 1:** Start
- **Step 2:** Declare variables num1, num2, and sum.
- **Step 3:** Read values num1 and num2.
- **Step 4:** Add num1 and num2 and assign the result to sum.

  Sum = num1 + num2
- **Step 5:** Display sum
- **Step 6:** Stop

Q.12 WRITE THE ALGORITHM TO FIND THE AVERAGE OF THREE NUMBERS.

ALGORITHM: Average of three numbers

- **Step 1:** Start
- **Step 2:** Declare variables num1, num2, num3 and avg.
- **Step 3:** Read values num1 and num2 and num3.
- **Step 4:** Apply formula (Average = Sum / No. of values)

  avg = (num1 + num2 + num3) / 3
- **Step 5:** Display avg
- **Step 6:** Stop

Q.13 WRITE THE ALGORITHM TO FIND THE VOLUME OF A BOX.

ALGORITHM: Volume of a box

- **Step 1:** Start
- **Step 2:** Declare the variables length, width, height and volume.
- **Step 3:** Read the values length, width and height.

- **Step 4:** Apply the formula {Volume = length x width x height}

  Volume = length x width x height
- **Step 5:** Display the volume
- **Step 6:** Stop

Q.14 WRITE THE ALGORITHM TO CALCULATE THE PERCENT.

ALGORITHM: Percent calculate.

- **Step 1:** Start
- **Step 2:** Declare the variables part, total and percentage.
- **Step 3:** Read the values part and total.
- **Step 4:** Apply the formula {Percentage = (part / total) x 100}

  Percentage = (part / total) x 100
- **Step 5:** Display the percentage
- **Step 6:** Stop

Q.15 WHAT IS A FLOWCHART?

FLOWCHART:-

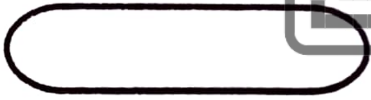
Flowchart is a diagrammatic representation of algorithm. It describes what operations are required to solve a given problem.

Q.16 Draw & Define symbols of flowchart

Symbols of Flow charts

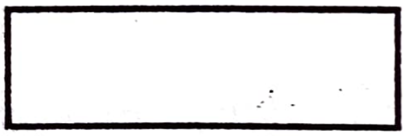
i. **Terminal (Oval)**

The terminal symbol represents the starting or ending of the flow chart.



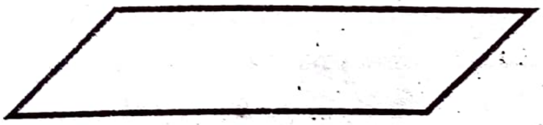
ii. **Rectangle(Process Box)**

This symbol is used for calculation / processing in a program.



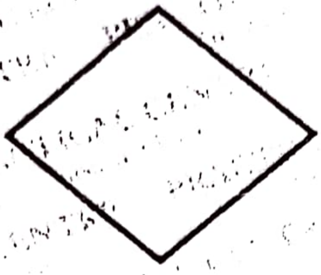
iii. **Parallelogram (I / O Box)**

This symbol is used for input / output statement in a program.



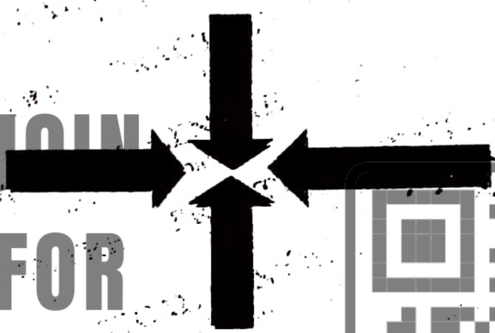
iv. **Diamond(Decision Box)**

A diamond represents a decision or branching point. Lines coming out from the diamond indicates different possible situations, leading to different sub-processes.



v. **Flow Line(Arrow Keys , Direction Keys)**

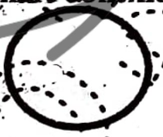
Flow line show the direction of the process flows.



vi. **Connectors**

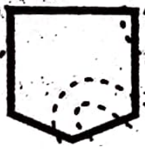
(a) **On Page Connector.**

When the remaining flow chart continue on same page ,we used on page connector.



(b) **Off Page Connector.**

When the remaining flow chart continue on next page ,we used on page connector



Q.17 LIST ANY THREE ADVANTAGES OF DESIGNING FLOWCHART.

ADVANTAGES OF USING FLOWCHARTS:

- **Communication:** Flowcharts are better way of communicating the logic of a system to all concerned or involved.
- **Effective analysis:** With the help of flowchart, problem can be analyzed in more effective way.
- **Proper documentation:** Program flowcharts serve as a good program documentation, which is needed for various purposes, making things more efficient.
- **Efficient Coding:** Flowcharts act as a guide or blueprint during the systems analysis and program development phase.
- **Proper Debugging:** Flowchart helps in debugging process.
- **Efficient Program Maintenance:** The maintenance of operating program becomes

easy with the help of flowchart. It helps the programmer to put efforts more efficiently on that part.

Q.18 WRITE THE IMPORTANCE OF FLOW CHART IN SOLVING A PROBLEM.

IMPORTANCE OF FLOWCHART IN SOLVING A PROBLEM:-

Flowchart illustrates the sequence of operations to be performed to solve a problem in the form of a diagram.

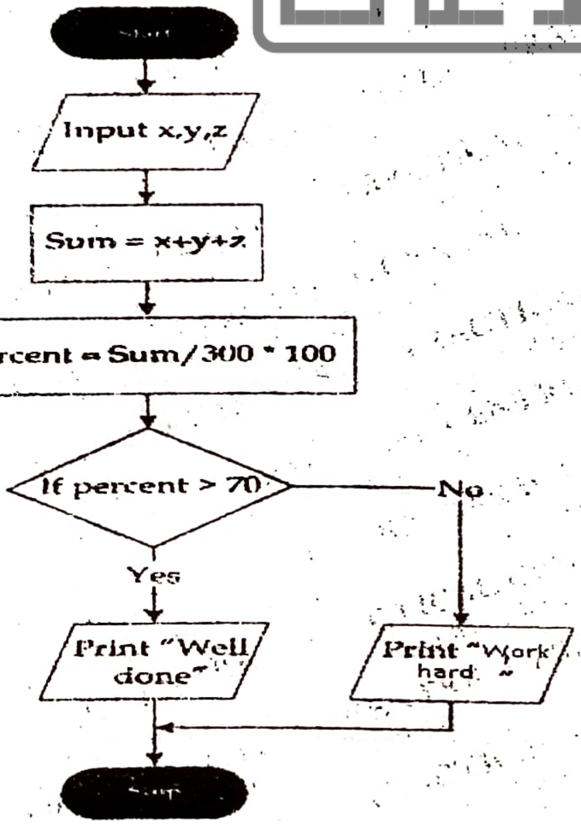
Computer programmers draw flowcharts before writing computer programs. Flowchart provides an easy way to analyze and find the solutions of the problems. Once the flowchart is drawn, it becomes very easy to write the program in any high level language. Flow chart is very helpful in communicating the problem solving method to other people. It also helps in finding and removing errors in computer programs.

Q.19 WRITE THE THREE DISADVANTAGES OF FLOW CHART.

DISADVANTAGES OF USING FLOWCHARTS:

- **Complex logic:** Sometimes, the program logic is quite complicated. In that case, flowchart becomes complex and clumsy. This will become a pain for the user, resulting in a waste of time and money, trying to correct the problem.
- **Alterations and Modifications:** If alterations are required, the flowchart may require re-drawing completely. This will usually waste valuable time.
- **Reproduction:** As the flowchart symbols cannot be typed, reproduction of flowchart becomes a problem.

Q.20 DRAW A FLOWCHART WHICH TAKES THREE NUMBERS AS INPUT, CALCULATES SUM AND PERCENTAGE. IF PERCENTAGE IS ABOVE 70, THEN PRINTS "WELL DONE", OTHERWISE "WORK HARD"



Q.21 DIFFERENTIATE BETWEEN ALGORITHM AND FLOWCHART.

DIFFERENCE BETWEEN ALGORITHM AND FLOWCHART.

S.No.	ALGORITHM	FLOWCHART
1.	Algorithm is step by step solution of a problem.	Flowchart is a diagram of different shapes which shows the flow of data through processing system.
2.	In algorithm, text is used.	In flowchart, symbols or shapes are used.
3.	Algorithm is easy to debug.	Flowchart is difficult to debug.
4.	Algorithm is difficult to write and understand.	Flowchart is easy to construct and understand.
5.	Algorithm does not follow any rules.	Flowchart follows rules for its construction.
6.	Algorithm is the pseudo code of the program.	Flowchart is just graphical or visual representation of program logic.

Q.22 WHAT ARE DATA STRUCTURE AND ITS TYPES?

DATA STRUCTURE:-

Data structure is a way of organizing and storing the data in a computer so that it can be accessed and modified efficiently. The main idea is to reduce the space and time complexities of different tasks.

TYPES OF DATA STRUCTURE

Basically, data structures are divided into two categories:

- Linear Data Structure
- Non-linear Data Structure

LINEAR DATA STRUCTURE:-

In Linear Data Structure data elements are arranged in sequential order and each of the elements is connected to its previous and next element. This structure helps to convert a linear data structure in a single level and in single run. They are easy to implement as computer memory is also in a sequential form. Linear data structures are not efficient in memory utilization.

EXAMPLE:-

Examples of linear data structures are:

- Stack
- Queue.
- Array etc.

NON-LINEAR DATA STRUCTURE:-

The elements of a non-linear data structure are not connected in a sequence. Each element can have multiple paths to connect to other elements. They support multi-level storage and often cannot be traversed in single run. Such data structures are difficult to implement but are more efficient in utilizing computer memory.

EXAMPLE:-

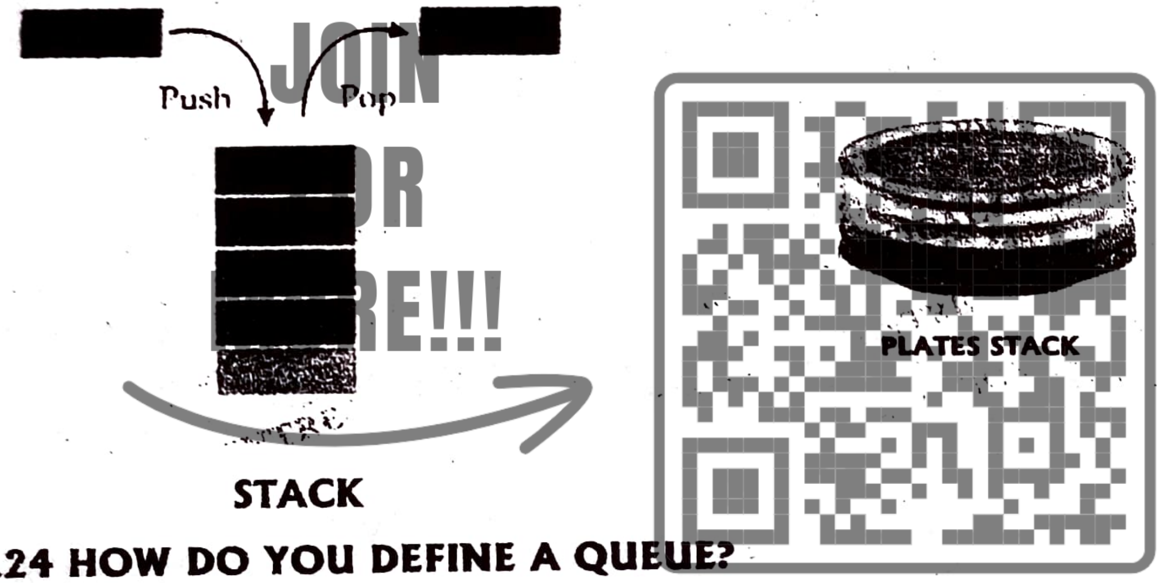
Examples of non-linear data structures are:

- Tree
- Graphs etc

Q.23 WHAT IS STACK IN DATA STRUCTURE?

STACK:-

Stack is a linear data structure which follows a particular order in which the operations are performed. The order may be LIFO(Last In First Out) or FILO(First In Last Out). There are many real-life examples of a stack. Consider an example of plates stacked over one another in the canteen. The plate which is at the top is the first one to be removed, i.e. the plate which has been placed at the bottommost position remains in the stack for the longest period of time. So, it can be simply seen to follow LIFO(Last In First Out)/FILO(First In Last Out) order.

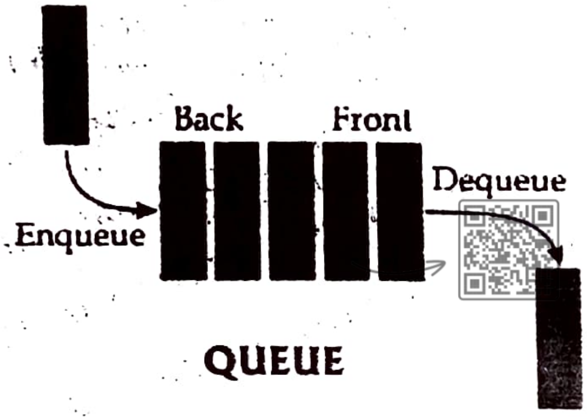


STACK

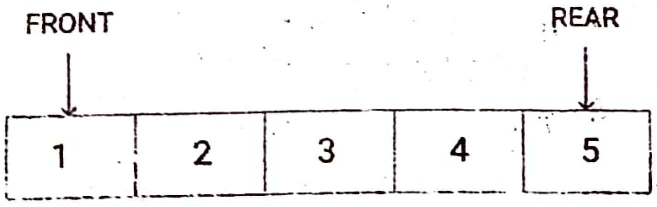
Q.24 HOW DO YOU DEFINE A QUEUE?

QUEUE:

A Queue is a FIFO (First In First Out) data structure where the element that is added first will be deleted first. The basic queue operations are enqueue (insertion) and dequeue (deletion). Enqueue is done at the front of the queue and dequeue is done at the end of the queue. The elements in a queue are arranged sequentially and hence, queues are said to be linear data structures.



QUEUE

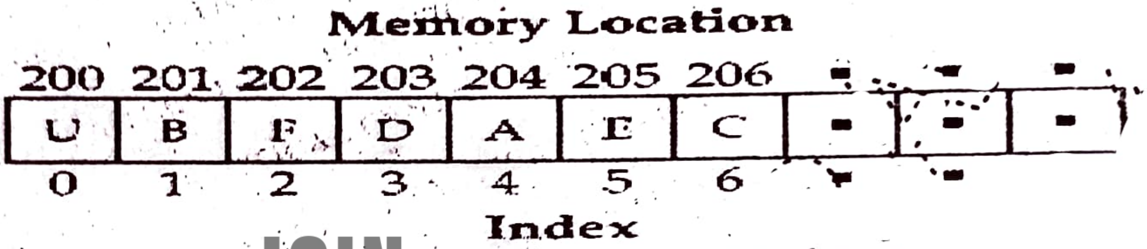


The practical examples of queues are:
- The consumer who comes first to a shop will be served first.
- CPU task scheduling and disk scheduling.
- Waiting list of tickets in case of bus and train tickets.

Q.25 DEFINE AN ARRAY IN DATA STRUCTURE.

ARRAY:

Array is a linear data structure, which holds a list of finite data elements of same data types. Each element of array is referenced by a set of index of consecutive numbers. The elements of an array are stored in successive memory locations. Most of the data



ARRAY (MEMORY LOCATIONS)

structures make use of arrays to implement their algorithms.

Two terms are necessary to understand array.

Element: Each item stored in an array is called an element.

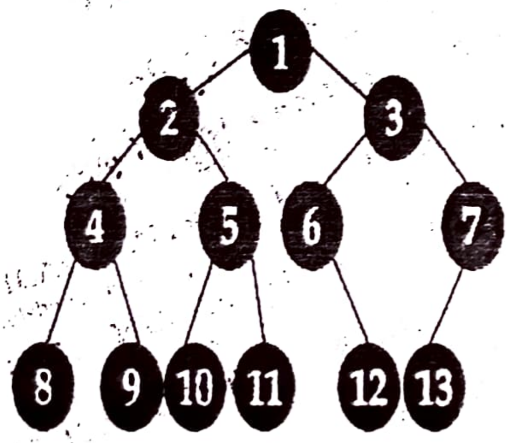
Index: Each location of an element in an array has a numerical value called index, which is used to identify the element. Set of indexes in an array are consecutive numbers.

Operations like Traversal, Search, Insertion, Deletion and Sorting can be performed on arrays.

Q.26 EXPLAIN TREE IN DATA STRUCTURE WITH EXAMPLE.

TREE:

Tree is a non-linear data structure which organizes data in hierarchical structure .Tree represents its elements as the nodes connected to each other by edges. In each tree collection, we have one root node, which is the very first node in our tree. If a node is connected to another node element, it is called a parent node and the connected node is called its child node. There is also a binary tree or binary search tree. A binary tree is a special data structure, used to store data in which each node can have a maximum of two children. Each node element may or may not have child nodes.

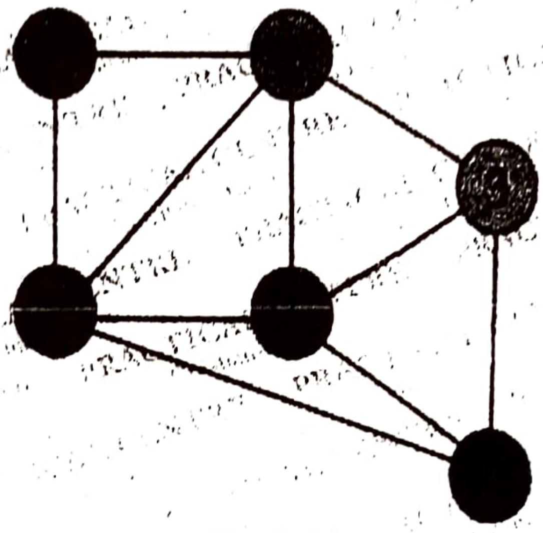


TREE DATA STRUCTURE

Q.27 WHAT IS GRAPH AND ITS TYPES IN DATA STRUCTURE?

GRAPH:

A graph is a non-linear data structure. It can be termed as a data structure consisting of data that is stored among many groups of edges (paths) and vertices (nodes), which are interconnected. Graph data structure (N, E) is structured with a collection of Nodes and Edges. Both nodes and vertices need to be finite. Graphs are used to solve network problems. Examples of networks, include telephone networks, social networks like Facebook, etc.


GRAPH

For example, a single mobile phone is represented as a node (vertex) where as its connection with other phones can be shown as an edge between nodes.

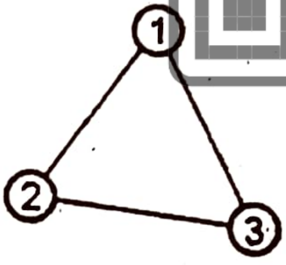
TYPES OF GRAPHS

There are two types of graphs:

- ➢ Undirected Graph
- ➢ Directed Graph

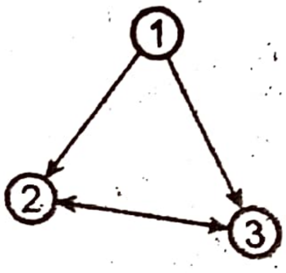
UNDIRECTED GRAPH:

In an undirected graph, nodes are connected by edges that are all bidirectional. For example, if an edge connects node 1 and 2, we can traverse from node 1 to node 2, and from node 2 to 1.



DIRECTED GRAPH:

In a directed graph, nodes are connected by directed edges – they only go in one direction. For example, if an edge connects node 1 and 2, but the arrow head points towards 2, we can only traverse from node 1 to node 2- not in the opposite direction.



Q28. WHAT IS THE DIFFERENCE BETWEEN TREE AND GRAPH DATA STRUCTURE.

S.NO	TREE	GRAPH
1.	Tree is a collection of nodes and edges, T={node, edges}	Graph is a collection of vertices/nodes and edges. G={node, edges}
2.	There is a unique node called root in the tree.	There is no unique node.
3.	There will not be any Cycle /Loops.	There can be loops/cycle.
4.	Represents data in the form of a tree structure, in a hierarchical manner..	Represents data similar to a network.
5.	In tree, only one path exists between two nodes.	In Graph one or more than one path between two nodes.
6.	In this, Preorder, Inorder and Postorder traversal is used.	In this, BFS and DFS traversal is used

Q29. WHAT IS THE DIFFERENCE BETWEEN QUEUE AND STACK DATA STRUCTURE.

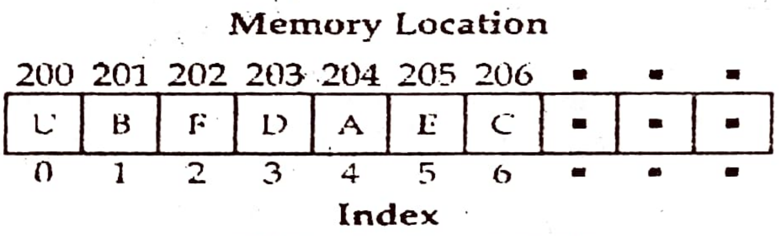
S.NO	STACK	QUEUE
1.	Stack is an abstract data type, represented by a physical stack of entities where insertion and deletion takes place at the same end.	Queue to also an abstract data type, similar to stack except it is open at both the ends.
2.	It is based on the working principle of LIFO meaning last in first out.	It is based on the working principle of FIFO meaning first in first out.
3.	Object can be added in the stack one at a time, meaning the object can be removed which is added last.	The object, which is added first, can only be removed from the queue.
4.	Push adds items to the stack and pop removes recently added item from the stack.	Enqueue adds items to the rear and dequeue removes items from the front of the queue.
5.	Only one pointer is used in a stack.	Two pointers are used in a simple queue case.

Q.30 WHAT IS STACK UNDERFLOW?

STACK UNDERFLOW: This is the situation when the stack contains no element. At this point, the stack top is present at the bottom of the stack.

Q31. WHAT IS THE NEED OF INDEX IN AN ARRAY?

INDEX: Each location of an element in an array has a numerical value called index, which i used to identify the eleme Set of indexes in an arra are consecutive numb



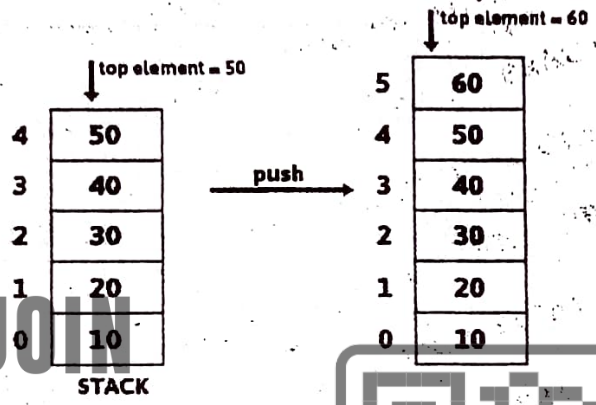
ARRAY (MEMORY LOCATION)

Q32. WITH THE HELP OF A SKETCH, DEFINE: PUS, POP, OVERFLOW AND ENQUEUE, DEQUEUE.

PUSH OPERATION:

Push operation refers to inserting or adding an element in the stack. Since there is only one position at which the new element can be inserted - Top of the stack, the new element is inserted at the top of the stack.

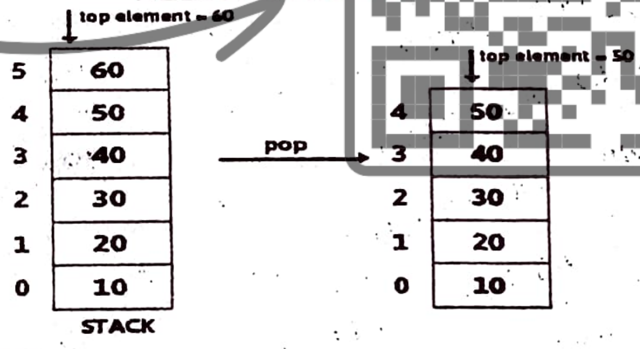
INSERTION IN STACK



POP OPERATION:

Pop operation refers to the deletion or removal of an element from the stack. Since there is only position from where we can remove the element –Top of the stack, the element is removed from the top of the stack.

DELETION IN STACK

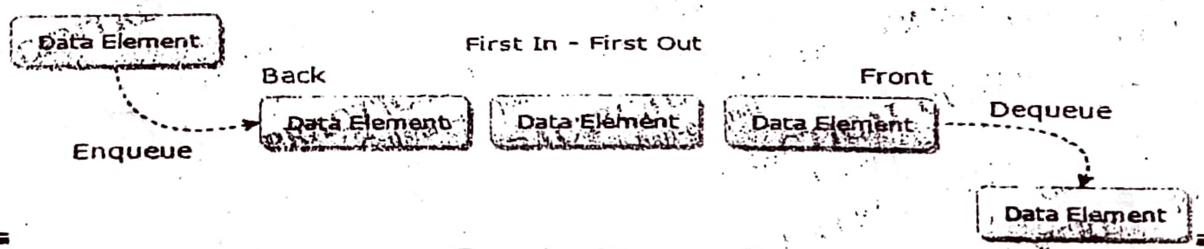


STACK OVERFLOW:

This is the situation when the stack becomes full, and no more elements can be pushed onto the stack. At this point, the stack top is present at the highest location of th stack.

Enqueue: Adds an item to the queue. Addition of an item to the queue is always done a the rear of the queue. If the queue is full, then it is said to be an Overflow condition.

Dequeue: Removes an item from the queue. An item is removed or de-queued alway from the front of the queue. The items are popped in the same order in which they ar pushed. If the queue is empty, then it is said to be an Underflow condition.



EXERCISE

A. ENCIRCLE/CHOOSE THE CORRECT ANSWER:

1. To "bake a cake" is an example of:
   a) Problem b) Strategy c) Algorithm d) Solution
2. To find a feasible solution to a problem, the first step is to:
   a) Establish starting point b) Find available solutions
   c) Create a strategy **d) Identify and analyze the problem**
3. Step by step solution of a problem in simple language is called:
   a) Problem Solving **b) Algorithm** c) Flowchart d) Data Structure
4. ____________ shows the logic of program graphically.
   a) Data Structure b) Graph c) Algorithm **d) Flowchart**
5. ____________ symbol is used for input/output in flowchart.
   a) Triangle b) Square **c) Parallelogram** d) Rectangle
6. Elements of ____________ data structure are not connected sequentially.
   a) Array **b) Graph** c) Queue d) Stack
7. ____________ stores data in hierarchal manner.
   a) Stack b) Queue c) Array **d) Tree**
8. When the data is pushed in stack, it means that data is:
   a) Inserted b) Deleted c) Sorted d) Edited
9. In binary tree, each child can have maximum:
   a) One node **b) Two nodes** c) Three nodes d) Four nodes
10. Traversing an array means accessing:
    a) First element b) Last element
    c) Any specific element **d) Each and every element of the array**

MCQS

Q1. CHOOSE THE RIGHT ANSWER:

1. ________ provide pictorial representation of given problem.
   a) Algorithm **b) Flowchart** c) Pseudocode d) All of these
2. ________ are a procedures or step by step process for solving a problem.
   a) Algorithm b) Flowchart c) Pseudocode d) All of these
3. The ______ symbol is used at the beginning of a flow chart.
   a) Terminal b) Rectangle c) Diamond d) None of these
4. The ________ symbol is used to represent decision in flowchart.
   a) Circle b) Rectangle **c) Diamond** d) None of these
5. The ________ symbol is used to represent process in flowchart.
   a) Circle **b) Rectangle** c) Diamond d) None of these
6. __________ symbol is used to represent input and output operation in flowchart.
   a) Circle b) Rectangle c) Diamond **d) Parallelogram**
7. _______ is a symbol used to connects two symbols of a flowchart.
   a) Circle b) Rectangle c) Diamond **d) Arrow**
8. The last step in problem solving is to:
   a) Define the problem **b) Implement and follow up on the solution**
   c) Generate alternative solutions d) Evaluate and select an alternative
9. Thing to keep in mind, while solving a problem, is:
   a) Input data b) Output data c) Stored data **d) All of these**
10. Before writing an Algorithm of a problem this step is more important:
    a) Collecting requirements **b) Identification of a problem**
    c) Start testing d) Analyzing the problem
11. Programming problems are similar to:
    a) Daily life problems b) Network Problems
    c) Hardware Problems d) Mobile Application Problems

12. The third step in process of problem solving is to:
a) Define the problem b) Implement and follow up on the solution
c) Generate alternative solutions **d) Evaluate and select an alternative**

13. To "bake a cake" is an example of:
a) **Problem** b) Strategy c) Algorithm d) Solution

14. To find a feasible solution to a problem, the first step is to:
a) Establish starting point b) Find available solutions
c) Create a strategy **d) Identify and analyze the problem**

15. Step by step solution of a problem in simple language is called:
a) Problem Solving **b) Algorithm** c) Flowchart d) Data Structure

16. This is a sequence of instructions written in a computer language to solve a problem:
a) Algorithm b) Flowchart **c) Program** d) Problem Analysis

17. Second step in problem solving is to:
a) Define the problem b) Implement and follow up on the solution
c) Generate alternative solutions d) Evaluate and select an alternative

18. The correct formula to calculate the sum of 10 numbers is; where sum=0
a) sum = sum + number b) sum = number + number
c) number = sum + number d) sum + number = number

19. Convert an Algorithm statement "Multiply x with y and stored in product" into programming statement:
a) x * y = product b) x * y **c) product= x * y** d) x * y * =product

20. The following code is an example of: x = 1, y = 2, x = x + y, Print z:
a) Programming code b) Pseudo code
c) Flowchart d) Algorithm

21. You are chasing score in cricket match and you need 40 runs in 14 balls with 1 wicket remaining; you will solve that problem thus:
a) You will listen to your partner **b) Design a step by step strategy on each ball**
c) Hit every ball without thinking d) Wait for easy over to start hitting

22. Selecting a best solution among different solution is:
a) Testing a problem b) Part of planning
c) Analysis of a problem d) All of these

23. When an algorithm is written in the form of programming language, it becomes:
a) Flow chart **b) Computer program**
c) Pseudo code d) Source code

24. This is incorrect:
Algorithms can be represented as:
a) as pseudo codes **b) as syntax**
c) as programs d) as flowcharts

25. A procedure for solving a problem in terms of actions and their order, is called:
a) Program instructions **b) Algorithm**
c) Template d) All of them

26. The first step in the process of problem solving is to:
a) Define the problem b) Implement and follow up on the solution
c) Generate alternative solutions d) Evaluate and select an alternative

27. An artificial and informal language that helps to develop algorithm, is called:
a) Instruction code b) Algocode **c) Pseudocode** d) Control code

28) This one is the process of inserting an element in the stack:
a) Insert b) Add **c) Push** d) None of these

29) When the user tries to delete the element from the empty stack then the condition is said to be a/an:
a) Underflow b) Garbage collection
c) Overflow d) None of the above

30) *If the size of the stack is 10, and we try to add the 11th element in the stack, then the condition is known as________.*
a) Underflow b) Garbage collection **c) Overflow** d) None of the above

31) This is not the correct statement for a stack data structure:
a) Arrays can be used to implement the stack. **b) Stack follows FIFO.**
c) Elements are stored in a sequential manner.
d) Top of the stack contains the last inserted element.

32) If the elements '1', '2', '3' and '4' are added in a stack, the order for the removal would be;
a) 1234 b) 2134 **c) 4321** d) None of these

33) A list of elements in this senqueue operation takes place from one end, and dequeue operation takes place from one end is:
a) Binary tree b) Stack **c) Queue** d) Linked list.

34) This is one of the following principles, Queue uses
a) LIFO principle **b) FIFO principle** c) Linear tree d) Ordered array

35) The necessary condition to be checked before deletion from the Queue is:
a) Overflow **b) Underflow** c) Rear value d) Front value

36) A linear data structure in which insertion and deletion operations can be performed from both the ends is
a) Queue b) Deque c) Priority queue d) Circular queue

37) We can describe an array in the best possible way as:
a) The Array shows a hierarchical structure. b) Arrays are immutable.
c) Container that stores the elements of similar types d) The Array is not a data structure

38) This is the advantage of the array data structure:
a) Elements of mixed data types can be stored. **b) Easier to access the elements in an array**
c) Index of the first element starts from 1. d) Elements of an array cannot be sorted

39) This is the disadvantage of the array:
a) Stack and Queue data structures can be implemented through an array.
b) Index of the first element in an array can be negative
c) Wastage of memory if the elements inserted in an array are lesser than the allocated size.
d) Elements can be accessed sequentially.

40) This is very useful in situation when data have to stored and then retrieved in reverse order:
a) Stack b) Queue c) List d) Linked List

41) Graph traversal is different from a tree traversal, because
a) trees are not connected. b) graphs may have loops.
c) trees have roots. d) None of these.

42) Stack is used for:
a) CPU Resource Allocation b) Breadth First Traversal
c) Recursion d) None of these

43) Push() and Pop() functions are found in
a) queue b) lists **c) stacks** d) trees

44) This illustrates a sequence of operations to be performed to solve a problem in the form of a diagram:
a) Algorithm **b) Flowchart** c) Program d) Problem Analysis

45) This is represented by a small circle in a flowchart:
a) Start/Stop b) Decision c) Processing **d) Connector**

46) The major components for measuring the performance of problem solving is/are:
a) Completeness b) Optimality
c) Time and Space complexity **d) All of the mentioned**

47) From the name of a Persian mathematician Abu Jaffar, Mohammad ibni Musa al Khowarizmi, this word has been coined:
a) Flowchart b) Flow **c) Algorithm** d) Syntax

48) When an algorithm is written in the form of a programming language, it becomes a:
a) Flowchart **b) Program** c) Pseudo code d) Syntax

49) This is a step by step recipe for solving an instance of a problem.
a) Algorithm b) Complexity c) Pseudo code d) Analysis

50) This is used to describe the algorithm, in less formal language.
a) Cannot be defined b) Natural Language **c) Pseudo code** d) None of these

BASICS OF PROGRAMMING OF C++ 2

Q.1 WHAT IS SYNTAX IN COMPUTER PROGRAMMING?

SYNTAX:

In programming, syntax refers to the rules that specify the correct combined sequence of symbols that can be used to form a correctly structured program using a given programming language. Programmers communicate with computers through the correctly structured syntax, semantics and grammar of a programming language.

Syntax tells the computer how to read a set of code. It is essentially a set of keywords and characters that a computer can read, interpret and convert into the task needed.

Example:

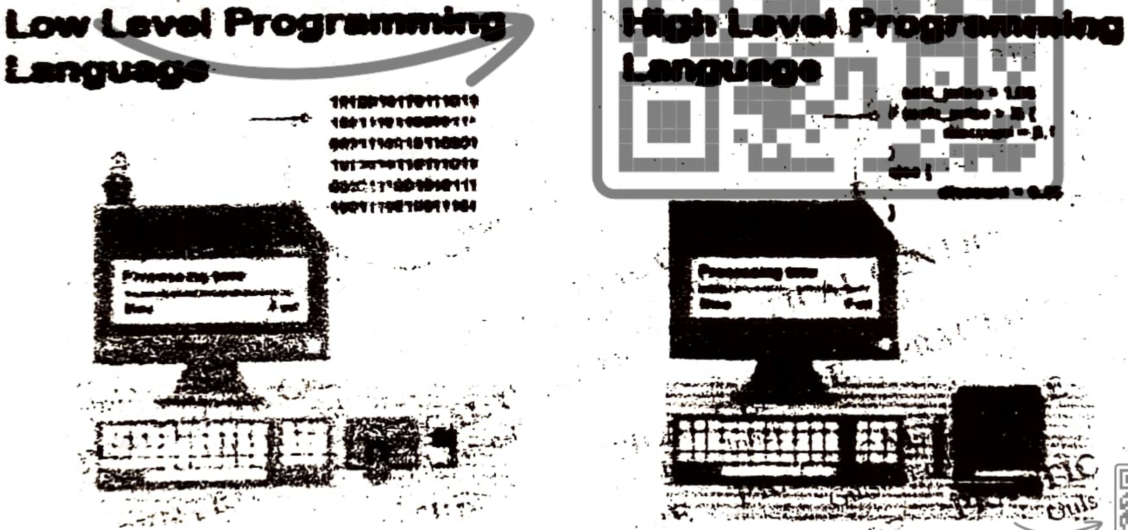
```
Cout << "Hello World";
```

In C++, this syntax displays the message "Hello World" on the screen. Syntax plays an important role in the execution of programs in text-base programming languages, and can even cause syntax errors if a programmer tries to run a program without using proper syntax. It is very common for new programmers to make syntax-based mistakes. Different programming languages use different types of syntax.

Q.2 DEFINE TRANSLATOR AND BRIEFLY DESCRIBE ITS ROLE.

TRANSLATOR:

Computer only understands machine code (binary). This code is difficult to read, write and maintain. Programmers prefer to use a variety of high and low-level programming languages instead. A program written in any language is called source code. To convert the source code into machine code, translators are needed.


Low Level Programming Language
High Level Programming Language

A translator takes a program written in source language as input and converts it into a program in target machine language as output. It also detects and reports the error during translation.

Q.3 DEFINE LANGUAGE TRANSLATOR AND ITS TYPES IN DETAIL

LANGUAGE TRANSLATOR:

Computers cannot execute programming instructions of a programming language. They work electronically and, therefore, must represent data and instructions in binary (1s and 0s) form. Language Translators are programs which convert or translate programming language (high level language) into 1s and 0s, or machine language (low level language). Language translators use two methods to translate high level language. They either compile it or interpret it. These methods give language translators either more common names of compilers and interpreters.), while assembler is used to translate a program written in assembly language into machine language.

TYPES OF LANGUAGE TRANSLATORS:

There are three types of system software used for translating the code that a programmer writes into a form that the computer can execute (i.e. machine code). These are:

1. Assemblers
2. Compilers
3. Interpreters

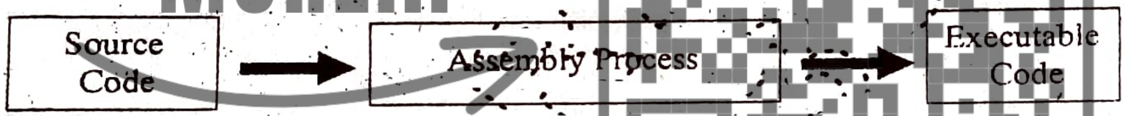
Source Code is the code that is input to a translator.

Executable code is the code that is output from the translator.

1. ASSEMBLER:

An Assembler is a program that converts assembly language program into machine language. Computer does understand only machine language, assembly facilitates the programmers to write the programs easily but still needs to convert the assembly language program into machine understandable form. The assembler is language translator for low level programming language.

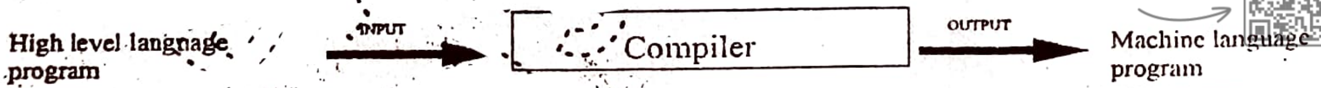
An assembler is like a compiler for the assembly language but interactive like an interpreter. Assembly language is difficult to understand as it is a low-level programming language. An assembler translates a low-level language, an assembly language to an even lower-level language, which is the machine code. The machine code can be directly understood by the CPU.

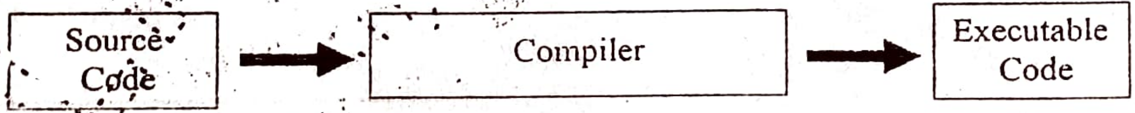


2. COMPILER:

A Compiler is a program that translates a high-level language into machine code. It translates whole high-level language program at once into machine language before it is executed.

A program written by a programmer in a high-level language is called a source program. After this source program has been converted into machine language by a compiler, it is referred to as an object program. The object program is saved as program file ready for execution at any desired time. As shown in the figure the input to a compiler (program) is a source program written in a high-level language and its output is an object program which consists of machine language instructions. Note that the source program and the object program are the same program, but at different stages of development.



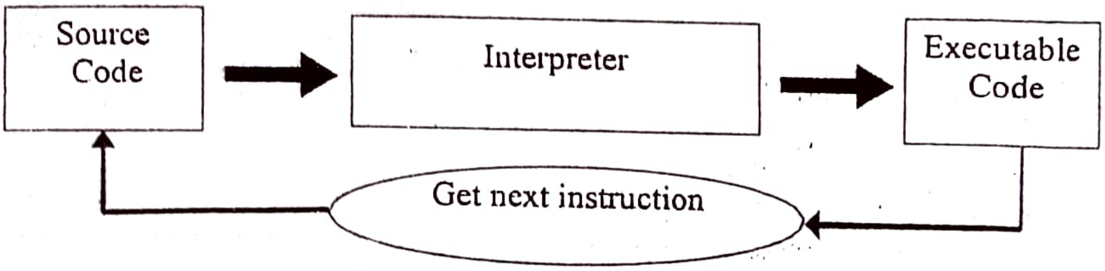


A compiler can translate only those source programs which been have written in the language for which the computer is meant. For example, a C compiler is only capable of translating source programs which have been written in C and, therefore, the computer requires a separate compiler for each high-level language.

While translating a given program, the compiler analyses each statement in the source program and generates a sequence or machine instructions which, when executed, will precisely carry out the computation specified in the statement. As the compiler analyses each statement, it uncovers certain types of errors.

3. INTERPRETER:

An Interpreter is a program that translates high-level source code into executable code. An interpreter translates one line at a time and then executes it: no object code is produced, and so the program has to be interpreted each time it is to be run.



An interpreter is a program that translates a high-level programming language into machine language during the actual step-by-step execution of a program. Translation and execution alternate for each statement encountered in the high-level language program. In other words, an interpreter translates one instruction, and the control unit executes the resulting machine language, the next instruction is translated, and the control unit executes the machine language instruction, and so on. If the program is run seven times a day, the programming language is reinterpreted seven times.

Q.4 WHAT ARE THE ADVANTAGES AND DISADVANTAGES OF A COMPILER?

ADVANTAGES OF A COMPILER:

FAST IN EXECUTION:

The object/executable code produced by a compiler can be distributed or executed without the presence of the compiler.
The object program can be used whenever required, without the need of re- compilation.

DISADVANTAGES OF A COMPILER:

1. Debugging a program is much harder. Hence, it is not very good at finding errors.
2. When an error is found, the whole program has to be recompiled.

Q.5 WHAT ARE THE ADVANTAGES AND DISADVANTAGES OF AN INTERPRETER?

ADVANTAGES OF AN INTERPRETER:

1. Good at locating errors in programs.
2. Debugging is easier since the interpreter stops when it encounters an error.
3. If an error is deducted, there is no need to retranslate the whole program.

DISADVANTAGES OF AN INTERPRETER:

RATHER SLOW

No object code is produced, so a translation has to be done every time the program is running.

Q.6 WHAT ARE THE ADVANTAGES AND DISADVANTAGES OF AN ASSEMBLER?

ADVANTAGES OF A ASSEMBLER:

Very fast in translating assembly language to machine code as 1 to 1 relationship.The symbolic programming is easier to understand, thus, time-saving for the programmer.
1. It is easier to fix errors and alter program instructions.
2. Efficiency in execution just like machine level language.

DISADVANTAGES OF A ASSEMBLER

1. It is machine dependent, so, cannot be used in other architecture.
2. A small change in design can invalidate the whole program.

3. It is difficult to maintain.

Q.7 DIFFERENTIATE BETWEEN COMPILER AND INTERPRETER.

S.NO	COMPILER	INTERPRETER
1.	Compiler translates the whole program at once and give an error list to be removed.	An interpreter translates high-level programming into machine language during the actual step by step execution of the program.
2.	Compiler creates object file.	Interpreter does not create object code file.
3.	Compilation is not necessary for repeated execution of a program.	Whenever an instruction is used, it must once again be interpreted and translated into machine language.
4.	Compiler is a complex program.	Interpreters are easy to write and do not require large space in the computer.

Q.8 WHAT IS ERROR EXPLAIN DIFFERENT TYPES OF ERROR

ERROR:

Errors are the problems or the faults that occur in the program which cause the program to behave abnormally. Programming errors often remain undetected until the program is compiled or executed. Some of the errors prohibit the program from getting compiled or executed. These errors should be removed before compiling and executing.

TYPES OF ERROR:

There are three types of errors:

- Syntax errors
- Logical errors
- Run time errors.

SYNTAX ERRORS:

A syntax error in computer science is an error in the syntax of a coding or programming language, entered by a programmer. Syntax errors are caught by a software program called a compiler, and the programmer must fix them before the program is compiled and then run.

Example:

If you misspelled a keyword or used a colon instead of a semicolon to end a statement.

LOGIC ERRORS:

Logic errors are those errors that prevent your program from doing what you expected it to do. On compilation and execution of a program, the desired output is not obtained when certain input values are given. These are one of the most common errors done by beginner programmers.

Example: If you got a math formula or algorithm correct.

RUNTIME ERRORS:

Errors, which occur during program execution (run-time) after successful compilation, are called run-time errors. A run-time error occurs when a program is asked to do something that it cannot perform, resulting in a 'crash'.

Example:

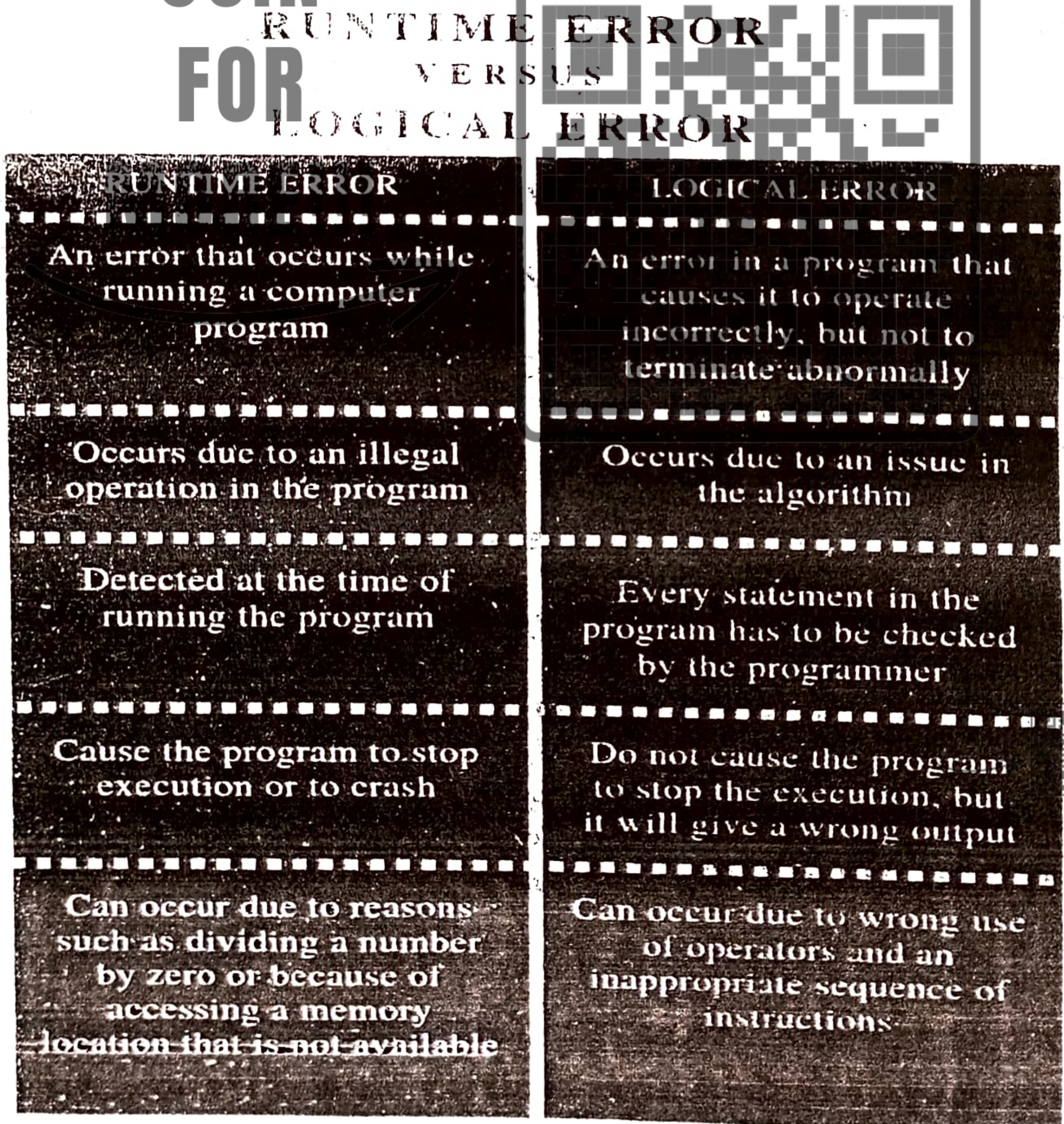
i. Number divided by 0.

ii. Assigning a string to an int.

Q.9 DIFFERENTIATE BETWEEN SYNTAX ERROR AND LOGICAL ERROR.

Syntax Error vs Logical Error	
A syntax error is an error in the syntax of a sequence of characters or tokens that is intended to be written in a particular programming language	A logical error is an error in a program that causes it to operate incorrectly but not to terminate abnormally.
Occurrence	
A syntax error occurs due to fault in the program syntax.	A logical error occurs due to a fault in the algorithm.
Detection	
In compiled languages, the compile indicates the syntax error with the location and what the error is	The programmer has to detect the error by himself.
Simplicity	
It is easier to identify a syntax error	It is comparatively difficult to identify a logical error.

Q.10 WRITE THE DIFFERENCE BETWEEN RUN TIME ERROR AND LOGICAL ERROR.



RUNTIME ERROR VERSUS LOGICAL ERROR

RUNTIME ERROR	LOGICAL ERROR
An error that occurs while running a computer program	An error in a program that causes it to operate incorrectly, but not to terminate abnormally
Occurs due to an illegal operation in the program	Occurs due to an issue in the algorithm
Detected at the time of running the program	Every statement in the program has to be checked by the programmer
Cause the program to stop execution or to crash	Do not cause the program to stop the execution, but it will give a wrong output
Can occur due to reasons such as dividing a number by zero or because of accessing a memory location that is not available	Can occur due to wrong use of operators and an inappropriate sequence of instructions

Q.11 WHAT ARE THE DIFFERENT TYPES OF PROGRAMMING LANGUAGES?

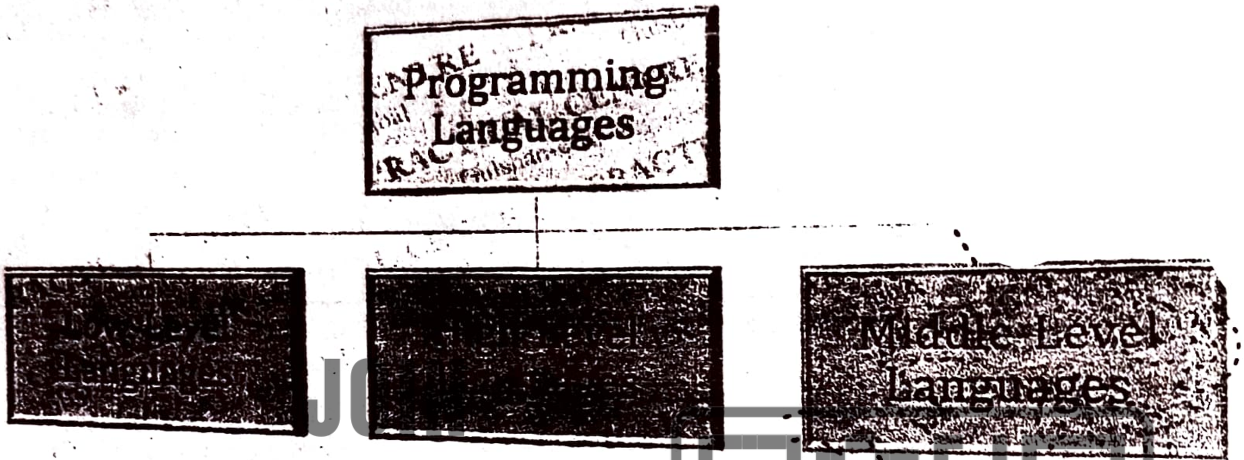
TYPES OF COMPUTER PROGRAMMING LANGUAGES:

PROGRAMMING LANGUAGE:

Programming language is the medium of communication between human and computer systems. It is the set of instructions written in a specific style (coding) to instruct the computer to perform a specific task.

TYPES OF PROGRAMMING LANGUAGE:

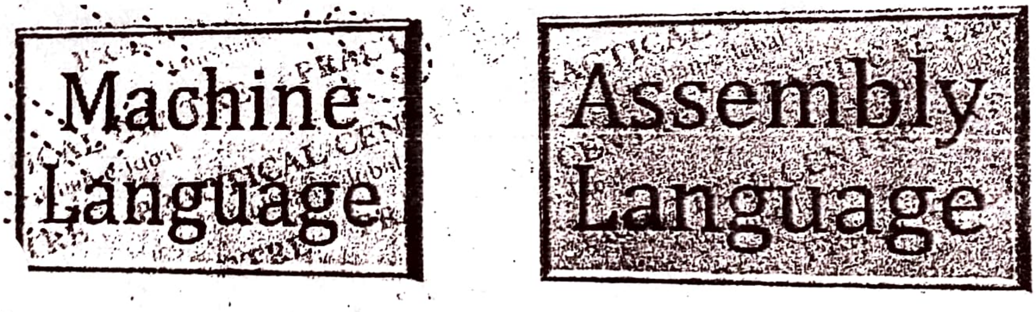
There are three types of programming languages: Low-level language, High level language and Middle level languages.



LOW LEVEL LANGUAGE:

A language which is closer to machine (computer)

Low level language is one which is closer to machine (computer).It is easier for machines to understand and difficult for humans to understand. It is faster in execution as compared to high level and middle level languages.
The low level language is a programming language that can directly access and communicate with the hardware, and it is represented in 0 or 1 forms, which are the machine instructions. The languages that come under this category are the Machine level language and Assembly language.



MACHINE LANGUAGE:

A computer or machine can only understand its machine language which is defined by its hardware architecture. This is one of the low-level programming languages which is the first generation language developed for communicating with a computer. It is written in machine code which represents 0 and 1 binary digits inside the computer string which makes it easy to understand and perform the operations. As we know a computer system can recognize electric signals. So, here 0 stands for turning off electric pulse and 1 stands for turning on electric pulse. It is very easy to understand by the computer and also increases the processing speed.

The main advantage of using Machine language is that there is no need of a translator or interpreter to translate the code, as the computer directly can understand. But there are some disadvantages also. Like you have to remember the operation codes, memory address every time you write a program and also hard to find errors in a written program. It is a machine dependent and can be used by a single type of computer.

ASSEMBLY LANGUAGE:

Assembly language is the second-generation programming language that has almost similar structure and set of commands as machine language. Instead of using numbers like in machine languages, here we use words or names in English forms and also symbols. The programs that have been written using words, names and symbols in assembly language are converted to machine language using an assembler. Because a computer only understands machine code languages that is why we need an assembler that can convert the assembly level language to machine language so the computer gets the instruction and responds quickly.

The main disadvantage of this language is that it is written only for a single type of CPU and does not run on any other CPU. But its speed makes it the most used low level language till today, which is used by many programmers.

HIGH LEVEL LANGUAGE:

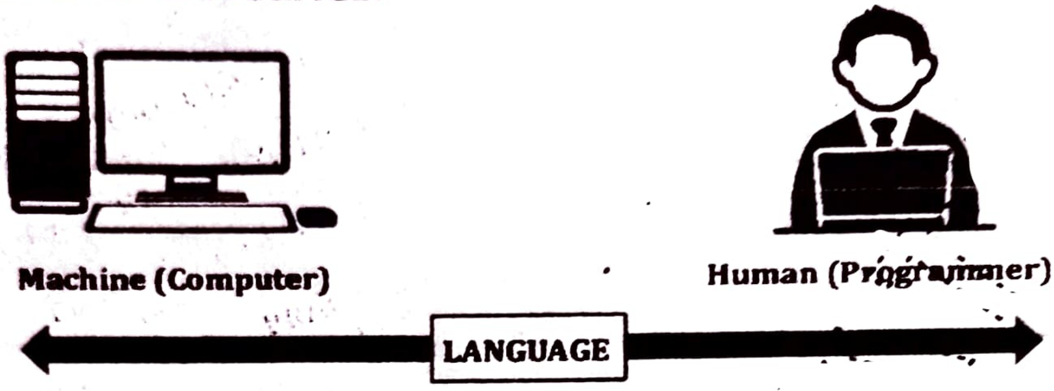
A language which is closer to human (programmer)

A high level language is one which is closer to the human (programmer).It is easier for humans to understand and difficult for machines to understand. It is slower in execution as compared to low level languages.

The high level languages are the most used and also more considered programming languages that help a programmer to read, write and maintain. It is also the third generation language that is used and also running till now by many programmers. They are less independent to a particular type of computer and also require a translator that can convert the high level language to machine language. The translator may be an interpreter and compiler that help to convert into binary code for a computer to understand. There are various high level programming languages like C, FORTRAN or Pascal that are less independent and also enable the programmer to write a program.

The compiler plays an important role on the computer as it can convert to machine language and also checks for errors if any before executing. There are several high level languages that were used earlier and also now like COBOL, FORTRAN, BASIC, C, C++, PASCAL, LISP, Ada, Algol, Prolog and Java. It is user-friendly as the programs are written in English using words, symbols, characters, numbers that need to be converted to machine code for processing.

MIDDLE LEVEL LANGUAGE:



A language which is some how closer to machine as well as human.

A middle level language is one which is closer to machine (computer) as well as to human (programmer). A language that has the feature of both low level and high level languages come under this category. Hence, we can say that the programming languages which have features of Low Level as well as High Level programming languages are known as "Middle Level" programming languages. More formally, a high level language, that allows you to write low level program in it, is called middle level language.

It bridges the gap between traditional machine understandable machine level language and more conventional high level language. This programming helps in writing operation system as well as application programming.

Using a middle level language, we can not only control the hardware but we can also develop application programs. Only few languages are there which contain the characteristics of both a high and a low level language.

C programming language is the best example of low level programming languages as it has the features of both low level and high level programming languages.

Q.12 WRITE THE DIFFERENCES BETWEEN LOW-LEVEL AND HIGH-LEVEL LANGUAGES.

Low-level language	High-level language
It is a machine-friendly language, i.e., the computer understands the machine language, which is represented in 0 or 1.	It is a user-friendly language as this language is written in simple English words, which can be easily understood by humans.
It requires the assembler to convert the assembly code into machine code.	It requires the compiler or interpreter to convert the high-level language instructions into machine code.
One type of machine code cannot run on all machines, so it is not a portable language.	The high-level code can be translated to required machine-code, so it is a portable language.
It has direct access to memory.	It is less memory efficient.
Coding and maintenance are not easy in a low-level language.	Coding and maintenance are easier in a high-level language.

Q.13 DESCRIBE THE PROGRAMMING ENVIRONMENT OF C++.

PROGRAMMING ENVIRONMENT OF C++:

C++ runs on lots of platforms like Windows, Liux, Unix, Mac, etc. Before we start programming with C++ we will need an environment to be set up on our local computer to compile and run our C++ programs successfully.

Q.14 WHAT IS IDE?

INTEGRATED DEVELOPMENT ENVIRONMENT:

An IDE, or Integrated Development Environment, enables the programmers to consolidate the different aspects of writing a computer program.

IDEs increase programmer productivity by combining the common activities of writing software into a single application: editing source code, building executables, and debugging.

Q.15 WRITE THE KEY BENEFITS OF IDE.

KEY BENEFITS OF IDE:

1. It serves as a single environment for most needs of a developer such as compilation, linking, loading, and debugging tools.
2. Code completion capabilities improve programming workflow.
3. It automatically checks for errors to ensure top quality code.
4. Refactoring capabilities allow developers to make comprehensive and mistake-free renaming changes.

Q.16 DEFINE THE COMPONENTS OF IDE.

COMPONENTS OF IDE:

IDEs increase programmer productivity by combining common activities of writing software into single application: editing source code, building executable, and debugging.

EDITING A SOURCE CODE:

This feature is a text editor designed to create, modify, and validate the source code. Writing code is an important part of programming. IDEs facilitate this process with features like syntax highlighting and auto complete.

SYNTAX HIGHLIGHTING:

Syntax Highlighting is a function of editors, which highlight individual components of the source code according to their syntax in colour, with changed fonts or through graphical changes. Syntax highlighting is alternatively called code colouring. Highlighting does not affect the functionality of the code, but it makes life easier for developers.

CODE COMPLETION:

When the IDE knows your programming language, it can anticipate what you are going to type next. Code completion features assist programmers by intelligently identifying and inserting common code components. These features save the developers, time writing code and reduce the chance of errors.

COMPILER:

Compilers are components that translate programming language into a form which the machine can process, such as binary code. IDEs provide automated build process for languages, so the act of compiling and executing code is done automatically.

LINKER:

The linker opens the compiled program file and links it with the referenced library files as needed. Unless all linker items are resolved, the process stops and returns the user to the source code file within the text editor with an error message. If no problems are encountered, it saves the linked objects as executable files.

LOADER:

The IDE directs the operating system's program called the loader to load the executable file into the computer's memory and have the Central Processing Unit (CPU) start processing the instructions.

DEBUGGING:

IDE provides debugging tools that allow programmers to examine different variables and inspect their code step by step. IDEs also provide hints while coding to prevent errors before compilation.

Q.17 DEFINE DEV- C++.

DEV- C++: -

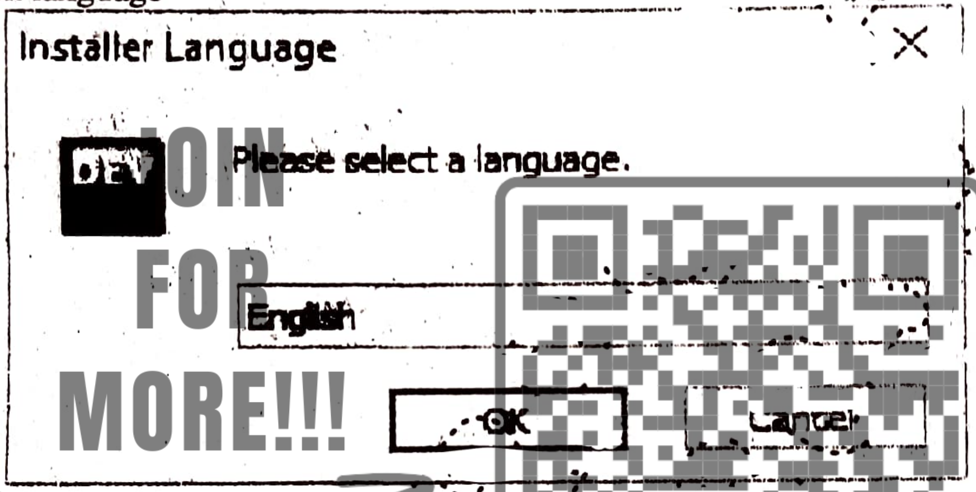
One of the most commonly used IDE for coding programs in C++ is Dev- C++. It is a full-featured integrated development environment for windows platforms. Dev-C++ is a fully feature IDE, supporting features like debugging, auto competition, localization, syntax highlighting, class and variables browsing, project management, package managers and others.

Q.18 WRITE THE STEPS TO DOWNLOAD, INSTALL AND CONFIGURE DEV-C++.

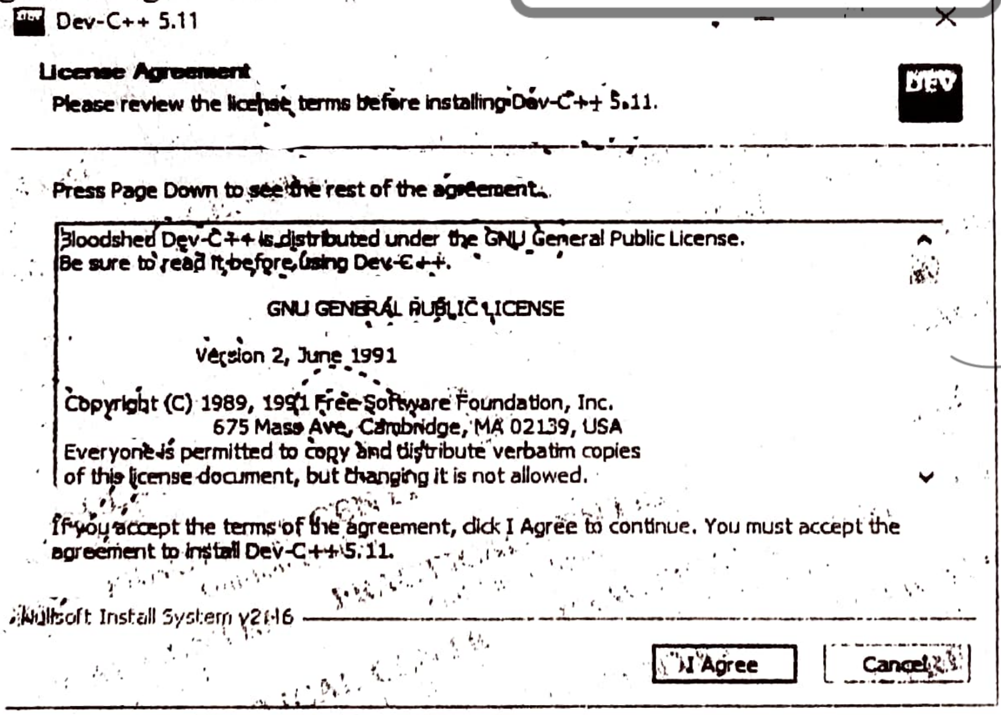
DOWNLOADING AND INSTALLING DEV-C++ IDE: -

Dev C++ is freely available for download from this link:
https://sourceforge.net/projects/orwelldevcpp/
After downloading the installation package, we can begin the installation process.

STEP 1:
Select English language



STEP 2:
Read and Agree the agreement



Please review the license terms before installing Dev-C++ 5.11.

Press Page Down to see the rest of the agreement.

Bloodshed Dev-C++ is distributed under the GNU General Public License.
Be sure to read it before using Dev-C++.

GNU GENERAL PUBLIC LICENSE

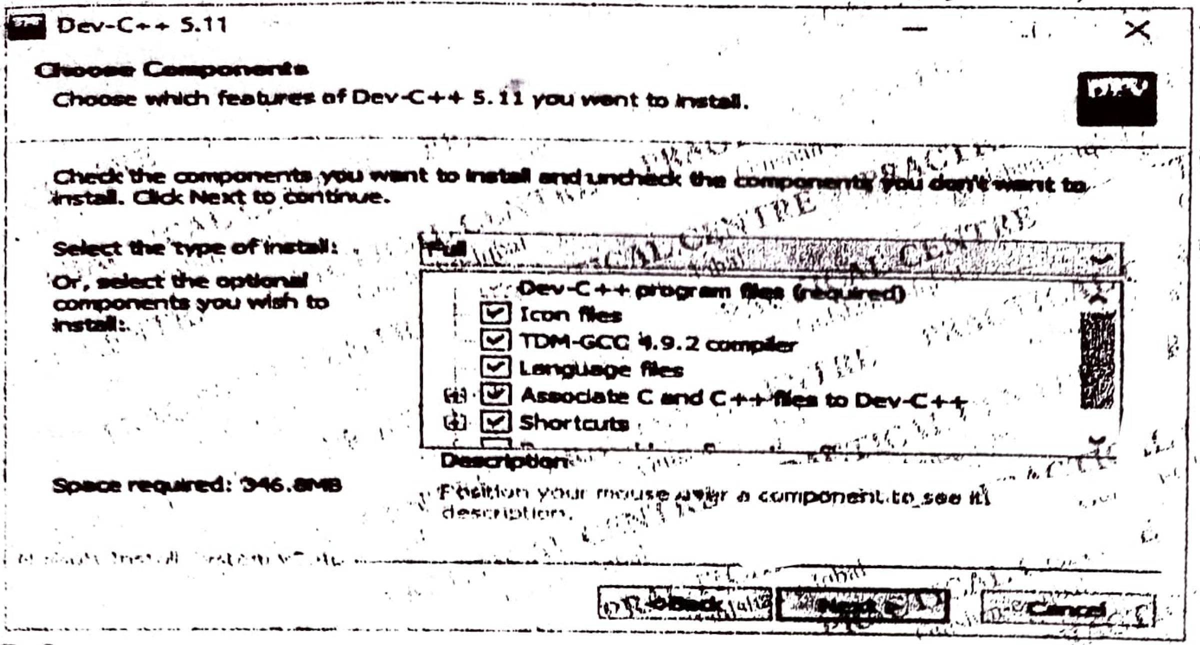
Version 2, June 1991

Copyright (C) 1989, 1991 Free Software Foundation, Inc.
675 Mass Ave, Cambridge, MA 02139, USA
Everyone is permitted to copy and distribute verbatim copies
of this license document, but changing it is not allowed.

If you accept the terms of the agreement, click I Agree to continue. You must accept the agreement to install Dev-C++ 5.11.

STEP-3:

Choose components (Click next, all needed components are selected by default)



Dev-C++ 5.11

Choose Components

Choose which features of Dev-C++ 5.11 you want to install.

Check the components you want to install and uncheck the components you don't want to install. Click Next to continue.

Select the type of install: Full

Or, select the optional components you wish to install:

- Dev-C++ program files (required)
- ✓ Icon files
- ✓ TDM-GCC 4.9.2 compiler
- ✓ Language files
- ✓ Associate C and C++ files to Dev-C++
- ✓ Shortcuts

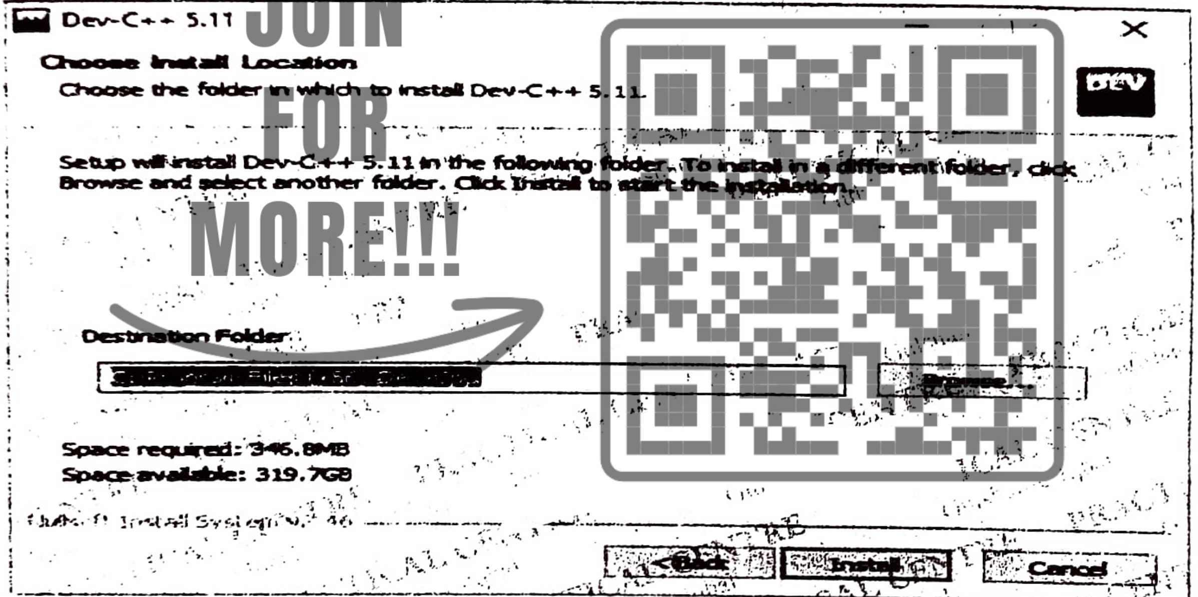
Space required: 346.8MB

Description

Position your mouse over a component to see its description.

STEP 4:

Choose Dev-C++ location



Dev-C++ 5.11

Choose Install Location

Choose the folder in which to install Dev-C++ 5.11.

Setup will install Dev-C++ 5.11 in the following folder. To install in a different folder, click Browse and select another folder. Click Install to start the installation.

Destination Folder

C:\Program Files (x86)\Dev-Cpp

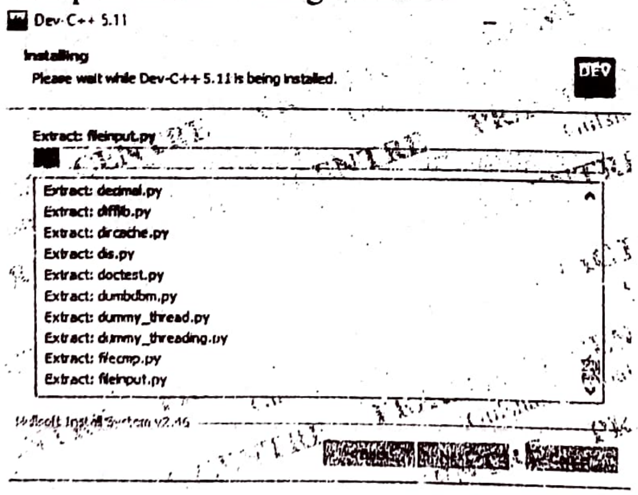
Space required: 346.8MB

Space available: 319.7GB

Nullsoft Install System v2.46

STEP 5:

Wait while the package components are being installed



Dev-C++ 5.11

Installing

Please wait while Dev-C++ 5.11 is being installed.

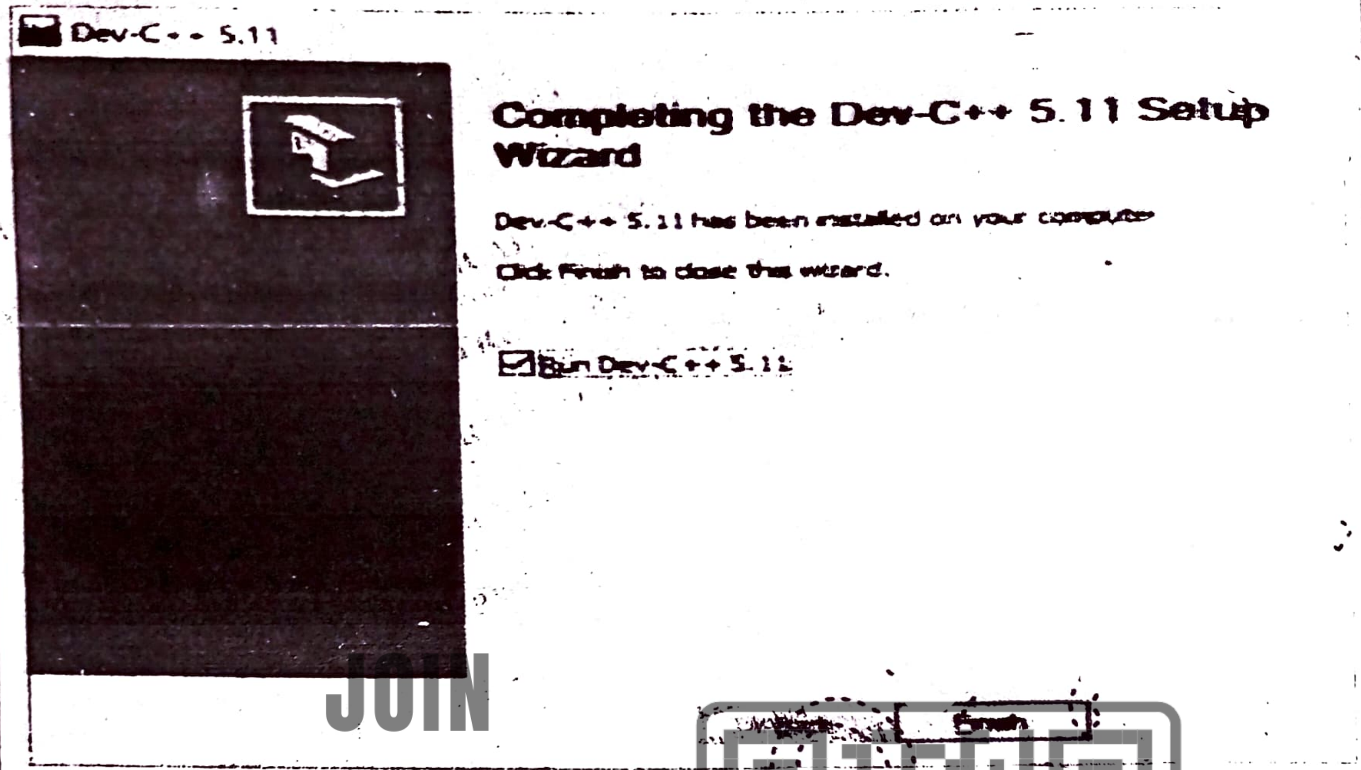
```
Extract: fileinput.py
```

```
Extract: decimal.py
Extract: difflib.py
Extract: dircache.py
Extract: dis.py
Extract: doctest.py
Extract: dumbdbm.py
Extract: dummy_thread.py
Extract: dummy_threading.py
Extract: filecmp.py
Extract: fileinput.py
```

Nullsoft Install System v2.46

STEP 6:

Done


Dev-C++ 5.11

Completing the Dev-C++ 5.11 Setup Wizard

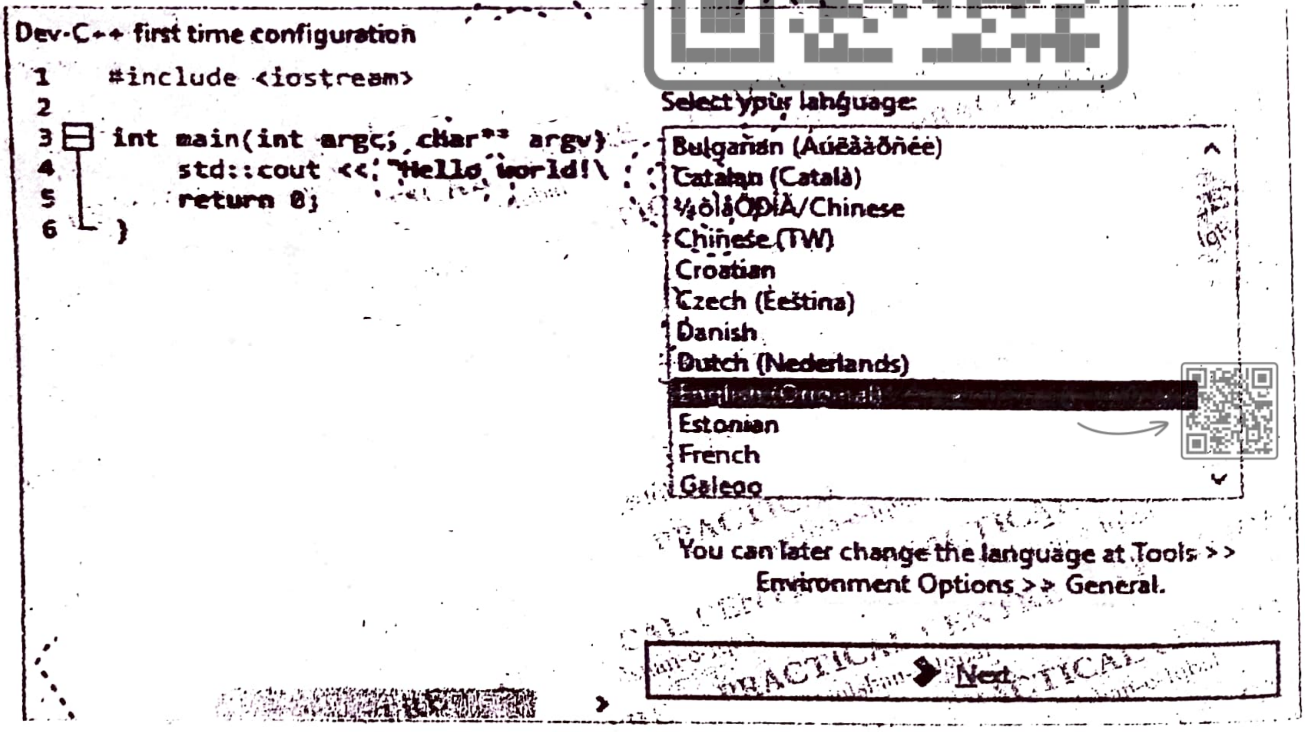
Dev-C++ 5.11 has been installed on your computer.

Click Finish to close this wizard.

CONFIGURE DEV-C++ IDE

STEP 1:

After installing Dev-C++ IDE, the following window will appear, select "English" and "Next" to continue.


Dev-C++ first time configuration

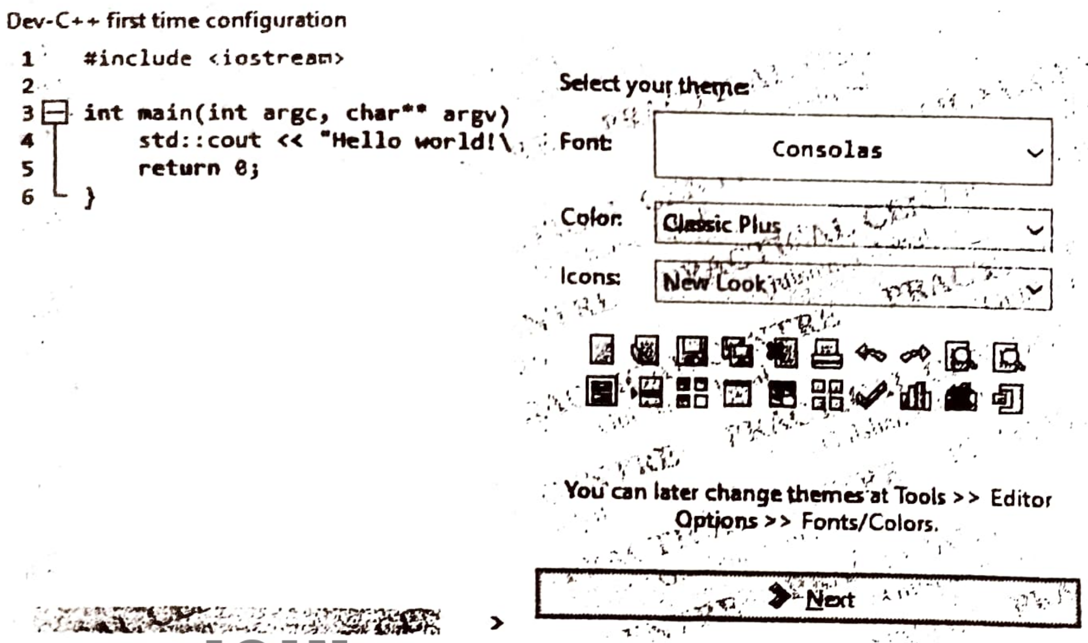
```
#include <iostream>

int main(int argc, char** argv) {
    std::cout << "Hello world!\
    return 0;
}
```

Select your language:

You can later change the language at Tools >> Environment Options >> General.

STEP 2:

On the "theme" selection dialog, leave the default setting and click on "Next" to continue.



Dev-C++ first time configuration

```
#include <iostream>

int main(int argc, char** argv)
    std::cout << "Hello world!\
    return 0;
}
```

Select your theme:

Font: Consolas

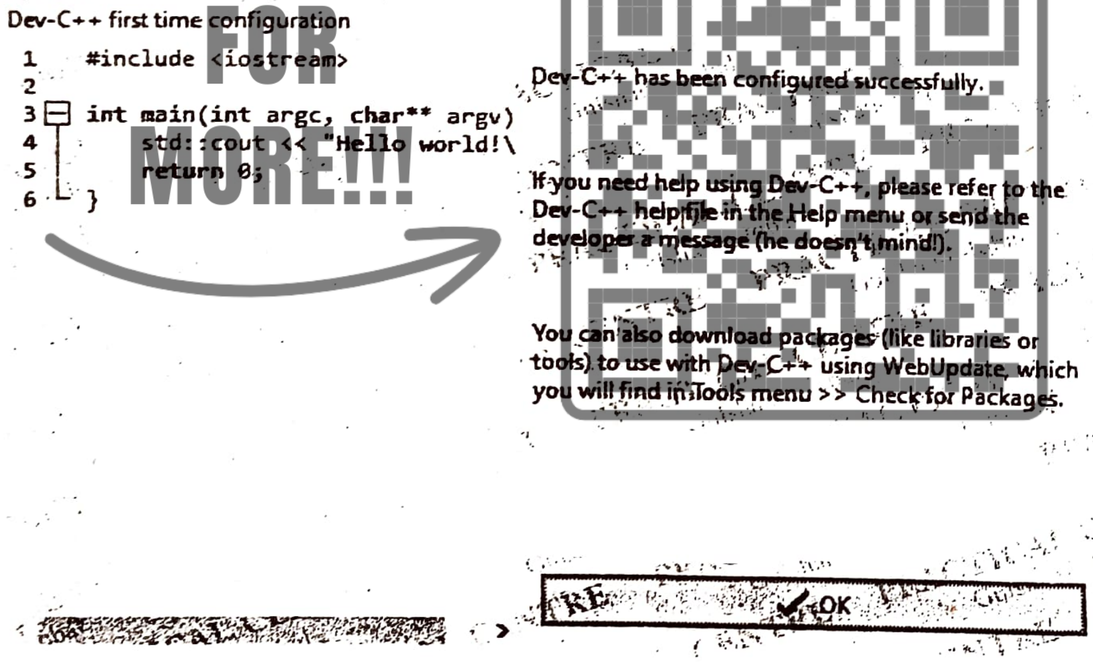
Color: Classic Plus

Icons: New Look

You can later change themes at Tools >> Editor Options >> Fonts/Colors.

STEP 3:

Click On "OK".



Dev-C++ first time configuration

```
#include <iostream>

int main(int argc, char** argv)
    std::cout << "Hello world!\
    return 0;
}
```

Dev-C++ has been configured successfully.

If you need help using Dev-C++, please refer to the Dev-C++ help file in the Help menu or send the developer a message (he doesn't mind!).

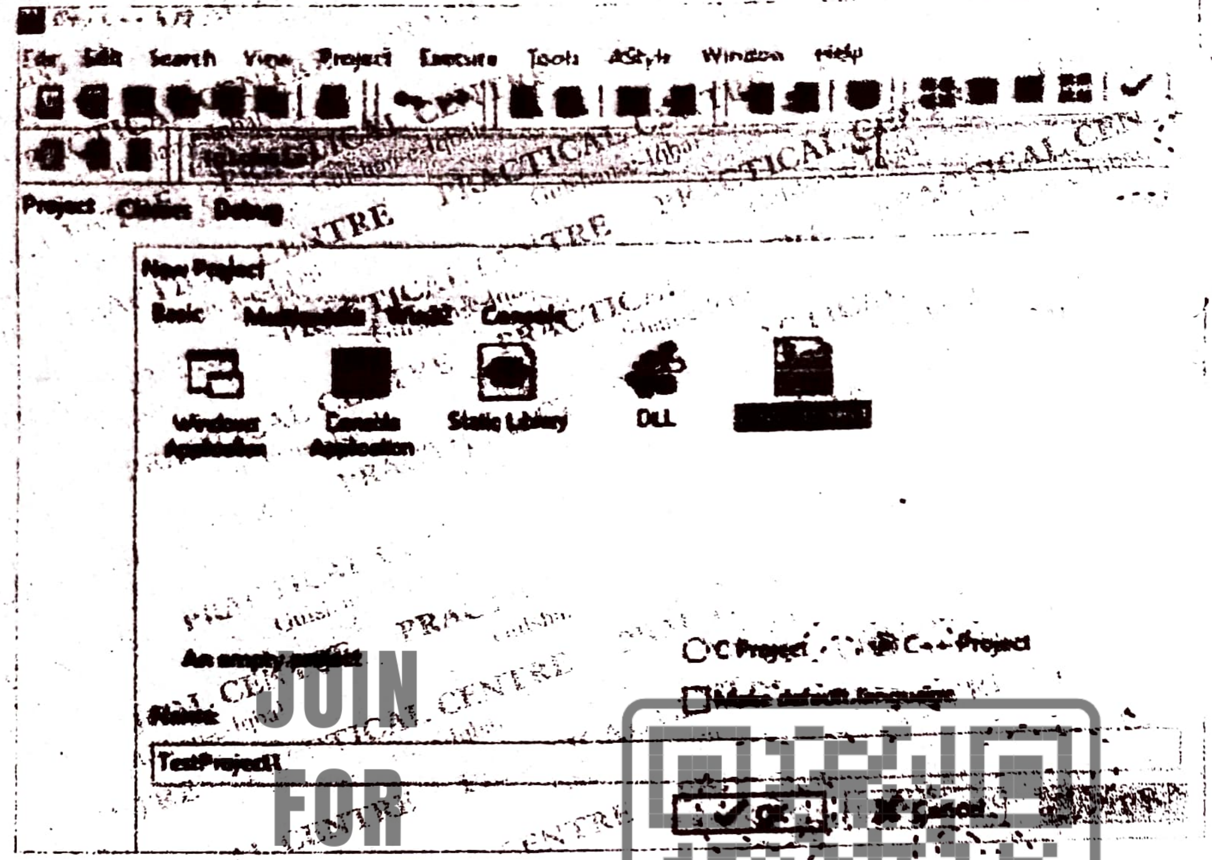
You can also download packages (like libraries or tools) to use with Dev-C++ using WebUpdate, which you will find in Tools menu >> Check for Packages.

Q.19 WRITE THE STEPS TO CREATE A NEW PROJECT IN DEV-C++.

STEPS TO CREATE A NEW PROJECT IN DEV-C++:

The steps to create a new project in Dev C++ are:

1. Click on File -> New -> Project.
2. From the New Project dialog, make sure Empty Project is selected. From language options, select C++ Project. Then enter a Name for your project.
3. Click on Oded-C++ will ask for the path where you want the new project to be stored. Once it is done, Dev-C++ will open a work space. It will show Project Explorer on the left side that shows the project we just created.



Q.20 WRITE THE STEPS TO ADD FILES TO PROJECT:

ADD FILES TO PROJECT:

A project requires source file which will contain codes for your program. The stops to create a file are:

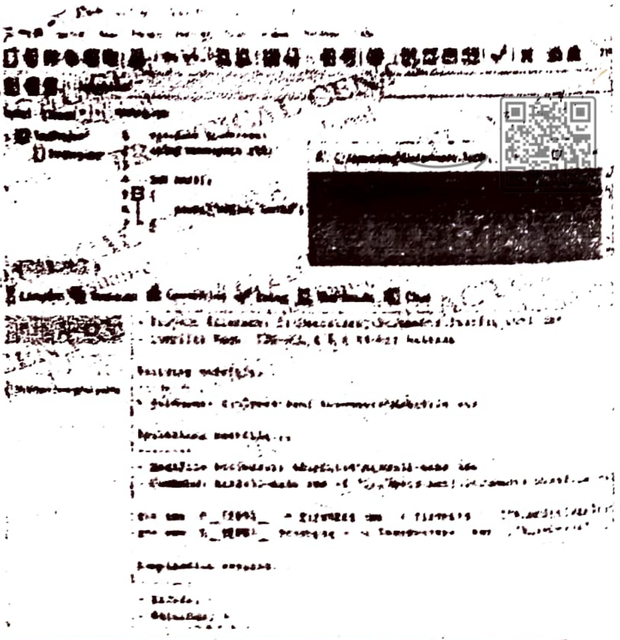
1. Click on Project ->New File. Alternatively, you can also right-click on the Project Name in the Project Explorer and click on New File
2. Click on Yes on the Confirm dialog to add a file. This file is not stored until it is deliberately saved.
3. To save newly added file, click on File ->Save. Enter a path where you want to save the field and provide its name. Click on Save to store the file.

Q.21 HOW DO YOU COMPILE AND EXECUTE C++ PROJECT?

COMPILE AND EXECUTE PROJECT:

Follow these steps to compile and run a project:

1. The project needs to be compiled before execution. To compiler, click Execute ->Compiler or pass F9 key. Compile Log tab shows the compilation status. Compiler tab will show if there are any syntax errors.
2. After successfully compiling the project, run it by clicking on Execute -> Run or by pressing F10 key.
3. A console window will open and show the output of the program.



Q.22 DEFINE C++ PROGRAMMING LANGUAGES.

C++ PROGRAMMING LANGUAGE: -

C++ is a middle-level programming language developed by Bjarne Stroustrup starting in 1979 at Bell Labs. C++ runs on a variety of platforms, such as Windows, Mac OS, and the various versions of UNIX.

C++ is a language with an advantage of programming low-level (drivers, kernels) and even higher-level applications (games, GUI, desktop apps etc.). C++ was developed as an enhancement of the C language to include object-oriented concept. The basic syntax and code structure of both C and C++ are the same.

Q.23 WHAT IS RESERVE WORDS IN C++?

RESERVED WORDS: -

A reserved word is a word that cannot be used as identifier, such as the name of a variable, function, or label. It is "reserved from use". This is syntactic definition, and a reserved word may have no meaning.

There is a total of 95 reserved words in C++.The reserved words of C++ may be conveniently placed into several groups. In the first group, we put those that were also present into C++.There are 32 of these.

There are another 30 reserved words that were not in C, so they are new to C++ programming language. Some of the commonly used C++ reserved are:

and	break	case
auto	char	do
bool	class	else
catch	const	export
default	continue	float
double	delete	goto
enum	explicit	inline
extern	false	module
for	friend	new
if	import	or
int	long	protected
nullptr	namespace	short
public	not	static
requires	operator	struct
signed	private	template
switch	register	throw
this	return	typedef
true	sizeof	union
unsigned	try	virtual
void	using	while

Q.24 WHAT ARE C++ DATA TYPES?

C++ DATA TYPES: -

A data type determines the type and the operations that can be performed on the data. C++ provides various data types and each data type is represented differently within the computer's memory.

Following table lists some of the basic C++ data types.

Type	Keyword	Size	Range
Boolean	bool	1 byte	0 (false), 1 (true)
Character	char	1 byte	-127 to 127 or 0 to 255
Integer	int	4 bytes	-2147483648 to 2147483647
Floating point	float	4 bytes	1.5×10^{-45} to 3.4×10^{38}. Stores fractional numbers. Sufficient for storing 7 decimal digits
Double floating point	double	8 bytes	5.0×10^{-345} to 1.7×10^{308}. Stores fractional numbers, Sufficient for storing 15 decimal digits

Q.25 DEFINE CONSTANT.

CONSTANT:

A **constant** is a data item whose value cannot change during the program's execution. Thus, as its name implies, - the value is constant.

Q.26 DEFINE VARIABLE.

VARIABLE:

A **variable** is a data item whose value can change during the program's execution. Thus, as its name implies, - the value can vary.

Q.27 WHAT ARE THE BASIC TYPES OF CONSTANT?

TYPES OF CONSTANT:

Constants are used in two ways. They are:

1. Literal constants
2. Defined constants

LITERAL CONSTANTS:

Literal constants are actual values fixed into the source code. Examples include the constants used for initializing a variable and constants used in lines of code:

Example:

```
21
12.34
'A'
"Hello world!"
false
null
```

NAMED CONSTANT:

A *Named Constant* is an identifier that represents a permanent value. The value of a variable may change during the execution of a program; but a *Named Constant* (or just *Constant*) represents permanent data that never changes (for example, π is a constant).

Example:

```
#define PI 3.14159
```
or
```
const double PI = 3.14159;
```

Q.28 DIFFERENTIATE BETWEEN CONSTANT AND VARIABLE.

Constant	Variable
A constant does not change its value during program execution.	A variable, on the other hand, changes its value depending on instructions.
Constants are usually written in numbers and may be defined in identifiers.	Variables are always written in letters or symbols.
Constants usually represent the known values in an equation, expression or in line of programming.	Variables, on the other hand, represent the unknown values.

Q.29 WHAT ARE THE RULES OF NAMING VARIABLES IN C++?

RULES FOR NAMING VARIABLES:

The rules and conventions for naming your variables can be summarized as follows:

1. Names can contain letters, digits and underscores.
2. Names must begin with a letter or an underscore (_).
3. Names are case sensitive (myVar and myvar are different variables).
4. Names cannot contain whitespaces or special characters like!,#,%,etc.
5. Reserved words (like C++ keywords, such as int) cannot be used as names.

Names cannot be longer than 32 characters in C++ by default.

Q.30 WHAT IS DECLARATION OF VARIABLE IN C++?

VARIABLE DECLARATION:

Declaration of a variable in a computer programming language, is a statement used to specify the variable name and its data type. Declaration tells the compiler about the existence of an entity in the program and its location. When you declare a variable, you should also initialize it.

SYNTAX:

DATA_TYPE VARIABLE _ NAME;

Where type is one of the C++ data types (such as int),and variable_name is the name of the variable (such as x or my Name).

VARIABLE INITIALIZATION:

Initialization is the process of assigning a value to the variable. Every programming language has its own method of initializing the variable. If the value is not assigned to the variable, then the process in is only called a declaration only.

SYNTAX: -

DATA_TYPE VARIABLE _ NAME = VALUE;

The equal sign is used to assign value to the variable.

EXERCISE

A. ENCIRCLE OR CHOOSE THE CORRECT ANSWER:

1. A computer program is a collection of:
a) Task **b) Instructions** c) Computer d) Programmers

2. High-level languages have syntax that is:
a) Easily readable by humans b) Easily readable by machines
c) Easily readable by both d) None of these

3. Low-level language have syntax that is:
a) Easily readable by humans **b) Easily readable by machines**
c) Easily readable by both d) None of these.

4. The primary characteristic of a compiler is to:
a) Translate codes line-by-line b) Translate low-level code to machine language.
c) Detect logic errors **d) Translate codes all at once**

5. The primary characteristics of an interpreter is to:
a) Translate codes line-by-line b) Translate low-level code to machine language
c) Detect logic errors d) Translate codes all at once

6. An Integrated Development Environment facilitates a programmer to:
a) Edit source code b) Complete and highlight syntaxes
c) Debug and compile codes **d) All of these**

7. All errors, detected by the user are typically:
a) Syntax Errors b) Semantic Errors c) Run-Time Errors **d) Logical Errors**

8. Allowed names for declaring a variable:
a) Can contain white spaces b) Can be one of the reserved words
c) Can contain letters, digits and underscores d) Can be the same as its data type

9. A bool data can store the following type of value:
a) Numbers c) Fractional numbers b) Strings **d) True or False**

10. This data type occupies the most space in memory:
a) Character **b) Double floating point** c) Floating point d) Integer

B. RESPOND THE FOLLOWING:

Q.1 What is computer program?

COMPUTER PROGRAM:-

A computer program is a sequence of instructions written using a Computer Programming Language to perform a specified task by the computer.

Q.2 List five common high-level languages used and describe their purpose.

BASIC (BEGINNERS ALL PURPOSE SYMBOLIC INSTRCUTION CODE):-

Beginner's All-Purpose Symbolic Instruction Code (BASIC) is a high-level and simple programming language that was developed by Dr. Thomas Kurtz and DR. John Kemeny at Dartmouth College in 1965 for teaching university students to program and provided with every self-respecting personal computer in the 1980s.BASIC has been the first programming language for many programmers. It also the foundation for Visual Basic.

PASCAL:-

Pascal is a general purpose language, which encourages good structure. As there are no line numbers, programs must be written in separate modules or procedures. An international standard, ISO Pascal, ensures portability between different machines. Pascal is a compiled language. So, execution of programs is fast. Pascal is used in colleges and universities and by professional programmers for writing commercial software.

COBOL(COMMON BUSINESS ORIENTED LANGUAGE):-

This is one of the oldest languages and uses statements very similar to English sentences. It is intended primarily for handling very large files of data and is used on mainframe computers, e.g. in banks, building societies and large commercial/industrial organizations.

FORTRAN (FORMULA TRANSLATION):-

FORTRAN is a high level language used for mathematical work in science and engineering, e.g. aircraft design. FORTRAN was one of the first languages used in mainframe computers and as it is compiled, is also used on powerful microcomputers. Being biased towards mathematics, FORTRAN is weak in text files handling.

C - LANGUAGE:-

C is a procedural programming language. It was initially developed by Dennis Ritchie in the year 1972. It was mainly developed as a system programming language to write an operating system. The main features of the C language include low-level memory access, a simple set of keywords, and a clean style; these features make C language suitable for system programming like an operating system or compiler development.

Q.3 Using the rules of naming variable, develop ten meaningful and valid variable names.

VALID VARIABLE NAMES:-

```
int age;
int Age;
int AGE;
int _age;
int age_ ;
int age1;
int my_age1;
int age, my_age1;
int age=10;
int MyAge;
```

Q.4 Write two differences between machine and assembly language.

DIFFERENCES BETWEEN MACHINE-LEVEL LANGUAGE AND ASSEMBLY LANGUAGE

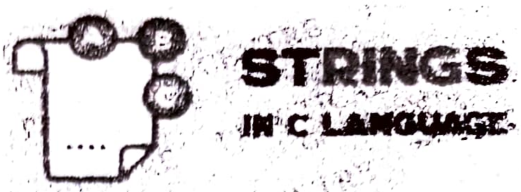
The following are the differences between machine-level language and assembly language:

Machine-level language	Assembly language
The machine-level language comes at the lowest level in the hierarchy, so it has zero abstraction level from the hardware.	The assembly language comes above the machine language. This means that it has less abstraction level from the hardware.
It cannot be easily understood by humans.	It is easy to read, write, and maintain.
The machine-level language is written in binary digits, i.e., 0 and 1.	The assembly language is written in simple English language, so it is easily understandable by the users.
It does not require any translator as the machine code is directly executed by the computer.	In assembly language, the assembler is used to convert the assembly code into machine code.
It is a first-generation programming language.	It is a second-generation programming language.

Q.5 What are strings in C + +?

Strings in C++

Variables that can store non-numerical values that are longer than one single character are known as strings.



The C++ language library provides support for strings through the standard string class. This is not a fundamental type, but it behaves in a similar way as fundamental types do in its most basic usage. Strings can be declared without an initial value and can be assigned values during execution.

Q.6 What is the difference between declaring and initializing a variable?

DECLARATION IN C++	DEFINITION IN C++
A statement that assures the compiler of the existing variable so that the compiler can proceed for further compilation without requiring the complete details about the variable	A statement that explains the compiler on where and how much storage to create for the variable
Declaration indicates the compiler of the existence of a variable	Definition indicates the compiler where and how much storage to create for a variable

Q.7 What is the difference between source code and object code?

S.NO	SOURCE CODE	OBJECT CODE
1.	Source code is in the form of text.	Object code is in the form of binary numbers.
2.	Source code is human readable code.	Object code is in machine readable format.
3.	Source code is generated by human or programmer.	Object code is generated by compiler.
4.	Source code is received by Compiler as an input.	Object code is generated by compiler as an output.

Q.8 List any four advantages of using an IDE.

ADVANTAGES OF USING AN IDE:-

1. The IDE can save the lot of effort writing a code.
2. Look and feel customizations such as themes, fonts, colors etc.
3. Folder structure for the project.
4. Auto compiles and runs code.
5. Provide syntax highlighting
6. Automatic indentation for code blocks.

MCQS

Q.1 Choose the right answer:

1. C++ was invented by:

a) Dennis Ritchie b) Ken Thompson

c) Brian Kernighan **d) Bjarne Stroustrup**

2. C++ is:

a) an object-oriented programming language b) a procedural programming language

c) both a procedural and an object-oriented programming language

d) a functional programming language

3. Translator, used to convert codes of assembly language into machine language, is termed as a/an:

a)Assembler b)Attempter c)Compiler d)Debugger

4. This Language is developed for business applications:

a) PASCAL b) COBOL c)C d)C++

5. Types of computer language translators are:

a) Compilers b) Interpreter c)Assembler **d) All of these**

6. A computer translator is best described as:

a) Application Software **b) System Software**

c)Hardware d) Window

7. Low level language is:

a) Human readable like language b) Language with big program size

c)Language with small program size

d) Difficult to understand and readability questionable

8. High level language is:

a) Human readable like language b) Language with small program size

c)Language with big program size

d) Language which is difficult to understand and not human readable

9. This computer program translates one program instruction at a time into machine language:

a) Interpreter b) CPU c)Compiler d) Simulator

10. This computer program that converts program into machine language all at once:

a) Interpreter **b) Compiler** c)Simulator d) Converter

11. Computer can understand this language directly:

a) High level language b) Assembly language

c) Machine language d) System program

12. Syntax errors are called:

a) Run time errors b) Compilation errors

c) Compile time errors **d) Both B and C**

13. Omitting semicolon at the end of a statement is called:

a) Run time error b) Execution time error

c) Syntax error d) Logical error

14. Error in a program is called:

a) bug b) debug c) virus d) noise

15. This refers to the process of locating and correcting errors:

a) Testing **b) Debugging** c) Retransforming d) Correcting

16 This these occur while forgetting to enter an instruction, or entering the instruction in wrong order:

a) Logical errors b) Syntax errors c) Exceptions d) Application shut down

17. This these occur when you break the rules of the language:

a) Logical error **b) Syntax errors** c) Exceptions d) Application shut down

18. This type of error cannot be detected by a language processor:

a) Syntax error **b) Logical error** c) Inception error d) Conception error

19. The full form of OOP is:

a) Object Oriented Programming b) Oriented Object Programming

c) Office Oriented Programming d) Office Objective Programming

20. These errors occur during program execution:

a) Logical errors b) Syntax errors

c) Run time errors d) Application shut down

21. An IDE allows:

a) Editing the program b) Compiling the program

c) Both editing and compiling **d) Editing, Building, Debugging, Compile**

22. The acronym IDE stands for:

a) Individually Determined Environment b) Integrated Development Entity

c) Integrated Development Environment d) Integrated Dynamic Entity

23. Eclipse is:

a) IDE b) Software c) GNU d) Code base

24. This element of IDE performs the debugging integration as well as tracking of code execution:

a) Compiler
b) Project management
c) Source Code Editor
d) IDE debugger

25. This one is not a correct variable type in C++:

a) float
b) real
c) int
d) double

26. This one is not a C++ keyword:

a) for
b) class
c) void
d) none of these

27. In C++, the words that is already defined and is reserved for a single special purpose, are called:

a) Keywords
b) Statement
c) Functions
d) Queries

28. Keywords are also called:

a) Preprocessors
b) Reserved words
c) Punctuation marks
d) Operators

29. Total number of keywords in C++ is:

a) 20
b) 35
c) 48
d) 52

30. C++ is a:

a) Case-sensitive language
b) Case-insensitive language
c) Only lowercase letter identifier
d) none of these

31. This one of the following is a reserved word in C++:

a) CHAR
b) char
c) character
d) Char

32. This one of the following variable name is valid:

a) _abcd
b) $abcd
c) long
d) -abcd

33. The constants are also called:

a) const
b) preprocessor
c) literals
d) variables

34. This one of the following, is the correct identified:

a) $var_name
b) VAR_123
c) varname@
d) none of these

35. Constants are declared as:

a) const keyword
b) #define preprocessor
c) both const keyword and # defined preprocessor
d) $define

36. The parts of the literal constants are:

a) integer numerals
b) floating-point numerals
c) strings and Boolean values
d) all of the mentioned

37. Constant variables can be created in CPP by using:

a) const b) enum c) #define **d) all of these**

38. This/These of the following is/are required to write and run CPP program:

a) Compiler b) Text Editor c) Operating System **d) all of these**

39. The following statement is used to __________ a variable:

int a;

a)Declare b)Initialize c)Debug d)Link

40. The following statement is used to __________ a variable:

int a=30;

a) Declare **b) Initialize** c)Debug d)Link

41. The value of these data items cannot be changed:

a) Class b) Return c) Constants d)Variable

42. These literals are treated as array of characters:

a) Graphic b) Character **c) String** d)Non-graphic

43. A variable is a symbol that represents:

a) A number b) A string

c) A storage location in the computer's memory d)Nothing

44. The information that is stored in a variable is called as:

a) String b) Literal **c) Value** d)Operator

45. This allocates memory, based on the data type of the variable:

a) Interpreter b) Link c) Converter **d) Compiler**

46. Pick the right option.

Statement 1: A definition is also a declaration.

Statement 2: An identifier can be declared just once.

a) Statement 1 is true, Statement 2 is false **b) Statement 2 is true, Statement 1 is false**

c) Both are false d) Both are true

47. Reusability of code can be achieved in CPP through:

a) Object **b) Inheritance** c) Encapsulation d) Class

48. Overflow is a one kind of

a) Syntax error b) Logical error **c) Run time error** d) none of these

49. This constant contains a single character enclosed within single quotes:

a) Character b) Numeric c) Fixed d) Floating point

50. Every variable should be separated by this separator:

a) Dot b) Colon **c) Comma** d) Semicolon

INPUT/OUTPUT HANDLING IN C++ 3

Q.1 EXPLAIN THE BASIC STRUCTURE OF C++ PROGRAM.

BASIC STRUCTURE OF C++: -

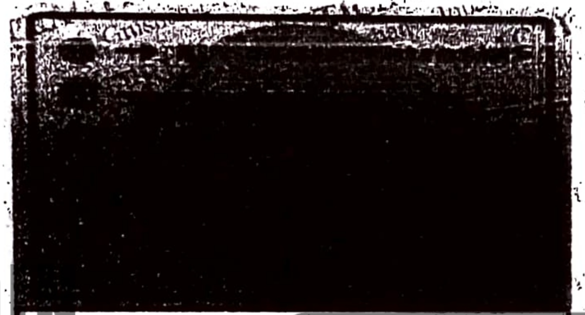
In C++ program is divided into three parts:

1. Preprocessor Directives

2. Main Function Header

3. Body of Program/Function

The basic structure of the C++ program is given in the Figure.



Basic Structure of C++ Program

S.NO.	Codes and symbols	Description
1.	#include <iostream>	The # symbol is called preprocessor Directives. **#include** is used to link the external header files/libraries which may be required in the program. **#define** is used to define constants in the program.
2.	Using namespace std;	This instruction tells the compiler to use standard namespace. The Namespace is the collection of identifiers. It is used for variables, class, function, and objects. All these elements of the standard library of C++ are declared within standard within standard **"std"**.
3.	In main (void)	**int main (void)** function is used for execution of C++ program. Void main (void) nothing to return value.
4.	{	This symbol indicates the beginning of the main function. It is also known as opening curly braces.
5.	Statement;	Instruction that perform a particular task is called a statement. Statement terminator (;) is used to end of the statement. This symbol also known as semi colon. For example: **cout << "Pakistan zindabad";** The output of the given example is that "Pakistan Zindabad" will be printed on the screen.
6.	Return value;	The return value is the exit code of your program. By default, **main ( )** in C++ returns in **int** integar data type value to the operating system.
7.	}	This symbol indicates the ending of the main function. It is also known as closing Curly braces

Q.2 HOW DO WE HANDLE INPUT-OUTPUT OPERATIONS INC++?

OR

WHAT IS THE FUNCTION OF IOSTREAM IN C++?

INPUT/OUTPUT HANDLING IN C++: -

In C++ Input and Output streams, perform Input/output (I/O) operation and these I/O streams are stored in header file. Such as <iostream>. These header files must be mentioned at the beginning of the program.

Q.3 DESCRIBE OUTPUT FUNCTIONS WITH EXAMPLES:

OUTPUT FUNCTIONS IN C++:-

Output means to display data on the screen or write the data to a printer or a file.
The C++ programming language provides standard library functions to read any input and to display data on the console.

S.no.	Functions	Description with examples
1.	cout statement	cout is predefined object in C++. It is used to display the output to the standard output device i.e monitor. "cout" insertion operator (<<). Syntax: cout <<variable or cout<<exp./string <<variable For example: **cout << "MY FIRST PROGRAM";**

2.	puts( )	This function is used to print the string to the output stream. The new line is automatically inserted after printing the string.
		Syntax: int puts(const chart char* str);
		For example.
		#incude<iostream>
		Using namespace std;
		int main (void)
		{
		Puts("MY FIRST PROGRAM");
		Return 0;
		}
		OUTPUT:
		MY FIRST PROGRAM

Q.4 DEFINE INPUT FUNCTIONS AND WRITE SOME INPUT FUNCTIONS WITH EXAMPLES.

INPUT FUNCTIONS IN C++: -

Input means to provide the program with some data to be used in the program. The C++ programming language provides standard library functions to read any given input and to display data on the console.

Q.5 What is statement terminator? Why do we use it in C++?

STATEMENT TERMINATOR (;) IN C++

Every statement in C++ must be terminated with semicolon (;). It indicates the end of statement. The semicolon (;) at the end of each statement in C++ specifies the distinction between statements. It does not matter if you put many statements on a single line, but it does matter if you do not use **semicolons** to separate them. The usage of each statement on its line is only to add clarity to the program. It is also called Statement terminator. If the terminator is missing, an error message will occur.

Q.6 What are escape sequence characters? List the escape characters mostly used in C++.

THE ESCAPE SEQUENCE:

The escape sequences are special non-printing characters that are used to control the printing behavior of the output stream objects (such as 'cout'). These characters are not displayed in the output. An escape sequence is prefixed with a backslash (\) and a coded character is used to control the printing behavior. The backslash (\) is called an escape character. So, the escape sequence looks like two characters.

The escape sequence is used inside a string constant or independently. These are written in single or double-quotes. The escape sequence can be inserted in any position of the string such as:

- At the beginning of the string.
- In the middle of the string.
- At the end of the string etc.

For example, the escape sequence '\n' is used to insert a new line. The cursor moves from the current position on the output device to the beginning of the next line. If escape sequence '\n' is inserted at the beginning of the string, then the string will be printed after printing a blank line e.g.

cout<<"\nWelcome";

LIST OF ESCAPE SEQUENCES IN C++

The commonly used escape sequences are given below:

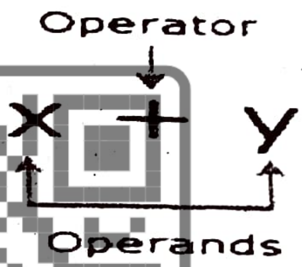
Escape Sequence	Explanation with example
\a	"a" means Alert or alarm. It caused a beep sound in the computer. **Example: cout << " \a";**
\b	"b" stands for backspace. It moves the cursor backspace. **Example: cout << "\b";**
	"t" stands for Horizontal tab. It is used to shift the cursor to a couple of spaces to the right in the same line. **Example: cout << " \b";**
	"n" stands for New line or line feed. It inserts a new line and cursor moves to the beginning of the next line **Example: cout << "\n" ;**

\r	Carriage Return "r". It is used to position the cursor to the beginning of the current line. **Example: cout << "\r";**
\\	" \ backslash It is used to print the backslash character. **Example: cout << "\ \t";**
\'	"Single quotation". It is used to print the apostrophe (') sign or character. **Example: cout << "\'";**
\"	"Double quotation" It is used to print the quotation mark. **Example: cout << \";**

Q.7 What are operator and its types in C++?

OPERATORS: -

An operator is a symbol that tells the compiler to perform specific mathematical or logical manipulations.



For example: x + y. Here, "x" and "y" are operand and "+" operators.

TYPES OF OPERATORS: -

There are seven types of operators that are used in C++ programming.

1. Arithmetic Operators 2.Increment Operators
2. Decrement Operators 4.Relational Operators
3. Logical Operators 6. Assignment Operators 7.Arithmetic Assignment Operators

Q.8 What are arithmetic operators. Explain with example?

ARITHMETIC OPERATORS:-

An arithmetic operator is a mathematical function that takes two operands and performs a calculation on them. They are used in common arithmetic and most computer languages contain a set of such operators that can be used within equations to perform a number of types of sequential calculation.

In arithmetic, operator five different operators are used to perform an arithmetic operation:

- Addition (+)
- Subtraction (-)
- Multiplication (*)
- Division (/)
- Remainder/Modulus (%)

All operators, except Remainder or Modulus operator, can be used in integer and float data type.

- **Addition (+):** It is used to perform arithmetic addition.
  Example: a+b;
- **Subtraction (-):** It is used to perform arithmetic subtraction.
  Example: a-b;

- **Multiplication (*):** It is used to perform arithmetic multiplication.
  Example: a*b;
- **Division (/):** It performs the arithmetic division of two numbers.
  Example: a/b;
- **Remainder / Modulus (%):** It is used to find the remainder of a division. It returns the remainder of an integer value. Remainder operator is also known as Modulus operator. This operator is used only with integral data types.

Example:5/2=2 and 1 is remainder. We can write in this form 5%2.

Q.9 Develop a simple calculator in C++ by using arithmetic operators.

```
#include<iostream>
#include<cstdio.h>
using namespace std;
int main(void)
{
        int a,b,add,sub,multi,remd;
        float div;
        cout <<"\n \t CALCULATOR";
        cout <<"\n \t ============";
        cout << "\n \t Enter the Value of a.....";
        cin >> a;
        cout << "\n \t Enter the value of b.....";
        cin >> b;
        add = a+b;
        cout << "\n \t Addition of "<< a <<"and.. "<< b <<"is...."<<
add;
        sub=a-b;
        cout  <<  "\n  \t  Subtraction  of  "<<  a <<"and..."<<  b  <<
"is...."<< sub;
        multi=a*b;
        cout  <<  "\n  \t  Multiplication  of  "<<  a <<"and..."<<  b  <<
"is...."<< multi;
        div=a/b;
        cout << "\n \t Division of "<< a <<"and..."<< b << "is...."<<
div;
        remd=a%b;
        cout  <<  "\n  \t  Remainder  of  Modulus  division  of  "<<  a
<<"and..."<< b << "is...."<< remd
return 0;
}
```

```
OUTPUT
CALCULATOR
======
Enter the value of a ..... 15
Enter the value of b ..... 10
Addition of 15 and 10 is..... 25
Subtraction of 15 and 10 is.....5
Multiplication of 15 and 10 is .....150
Division of 15 and 10 is .....1
Remainder of Modulus division of 15 and 10 is .....5
```

Q.10 Describe Increment operator with example.

INCREMENT OPERATOR (++): -

The increment operator can be used with any type of variable. It is used to add 1 to the value of a variable. Increment operator represented by ++ (double plus sign). This operator can be applied only to a single variable.

There are two ways to use increment operator:

PREFIX INCREMENT OPERATOR: -

The prefix increment operator (++) is used to increment the value before it is getting used. It means this operator increments the value and that incremented value is then used. You can apply this operator before the variable name. It can be written like ++a. i.e. x=++a;

Example:

```
#include < iostream >
Using namespace std;
Int main (void)
{
Int a = 10;
Cout << "\n value of is. . . . ." << ++a; ➡ Prefix Increment Operator return 0;
  }
```

In prefix increment, operation value of a is printed as 11 because 1 is added in 'a' before printing.

POSTFIX INCREMENT OPERATOR: -

If the increment operator is applied after the variable name, it is known as Postfix Increment operator. It is used to increment the value of the variable after it is used. It means this operator postpones the increment of the value. It is written like a++.

Example:

```
#include < iostream >
Using namespace std;
Int main (void)
{
Int a =10;
Cout << "/n value of a is . . . . ."  << a++ ; ➡ postfix Increment operator return 0;
}
```

In postfix increment, operation value of a is printed as 10 because 1 is added in 'a' after printing. So, the value of 'a' will change to 11 after printing.

Q.11 Explain Decrement operator with example.

DECREMENT OPERATOR (--):-

The decrement operator can be used with any type of variable. It is used to subtract 1 to the value of a variable. Decrement operator represented by -- (double minus sign). This operator can be applied only to a single variable.

There are two ways to use decrement operator:

PREFIX DECREMENT OPERATOR: -

The prefix decrement operator (--) is used to decrement the value before it is getting used. It means that this the decreased value is then used. You can apply this operator before the variable name. It can be written like --a. i.e. x=--a;

Example

```
#include<iostream.h>
#include<conio.h>

void main()
{
int x,i;
i=10;
x=--i;
cout<<"x: "<<x;
cout<<"i: "<<i;
getch();
}
```

Output

```
x: 9
i: 9
```

POSTFIX DECREMENT OPERATOR:-

The postfix decrement operator (--) is used to decrease the value of the variable after it is used. It means this operator postpones the decrement of the value. You can apply this operator after the variable name. It can be written like a--. i.e. x=a--;

Example

```
#include<iostream.h>
#include<conio.h>

void main()
{
int x,i;
i=10;
x=i--;
cout<<"x: "<<x;
cout<<"i: "<<i;
getch();
}
```

Output

```
x: 10
i: 9
```

Q.12 Explain Relational operators with example.

RELATIONAL OPERATOR:-

The relational operators are used to test the relation between two values. All relational operators are binary operators. These operators must require two operands. The result of the comparison is True (1) or False (0). The relational operators are also known as Comparison Operators. The following are the relational operators and their operations.

Operators	Meaning	Purpose	Expression
= =	Equal to	Its checks the equality of two operands values.	a = = b
! =	Not equal to	It checks whether the value of the left operand is not equal to the value of the right operand.	a ! = b
>	Greater than	This operator checks the value of left operand is greater than the value of right operand.	a > b
<	Less than	It checks the value of the left operand is less than the value of the right operand.	a < b
>=	Greater than or equal to	It checks that the value of the left operand is greater than or equal to that of the right operand.	A >= b
<=	Less than or equal to	It checks that the value of the left operand is less than or equal to the value of right operand.	a <= b

Q.13 Differentiate between Arithmetic operators and Relational operators.

Arithmetical Operator	Relational Operator
Arithmetic operators are used to perform mathematical operations.	Relational operators are used to compare the values of two expressions. Relational operators are binary operators because they require two operands to operate.
+, -, *, /, etc. are a few examples of Arithmetic operators.	<,<=,>,>=,==,!= are a few examples of Relational Operators

Q.14 Define Logical operators with example.

LOGICAL OPERATOR: -

Logical operators are used to evaluate two or more conditions. In general, Logical operators are used to combine relational expressions, but they are not limited to just relational expression. You can use any kind of expression, even constants. If the result of the logical operator is true, then 1 is returned, otherwise 0 is returned.

Logical operators are very important in any programming language and they help us take decisions based on certain conditions. There are three logical operators in use in C++ programming.

Operators	Description	Expression
&&	This operator is called AND. The condition will be true if both expressions are true.	x = 10, y = 5, z = 12 x > y && x < z
| |	It is known as OR operator. The condition will be true if any one of the expressions is true.	x = 10, y = 5 z = 12 x > y | | x > z
!	This operator is called NOT. The condition will be inverted, false becomes true, while true becomes false.	x = 10, y = 5; ! (x < y) ;

Q.15 Draw the truth table of logical operators.

1. Logical AND Operator (&&):

The Logical AND Operator works on the following Truth Table of AND Logic:

Inputs		Output
A	B	A && B
0	0	0
0	1	0
1	0	0
1	1	1

2. Logical OR Operator (||):

The Logical OR Operator works on the following Truth Table of OR Logic:

Inputs		Output
A	B	A || B
0	0	0
0	1	1
1	0	1
1	1	1

3. Logical NOT Operator (!):

Logical NOT Operator is a Unary Operator i.e., it requires only one operand to work on. Following is the Truth Table of NOT Logic:

Input	Output
A	!A
0	1
1	0

Q.16 Differentiate Relational and Logical operators.

Relational Operator	Logical Operator
Relational operators are used to compare the values of two expressions. Relational operators are binary operators because they require two operands to operate.	Logical operators operate on boolean expressions to combine the results of this boolean expression into a single boolean value.
These operators are used to perform logical operations on two given variables.	These operators are used to compare the two relational statements.
The relational operators used to compare any two values.	The logical operators are used to combine one or more than one relational expression
<,<=,>,>=,==,!= are a few examples of Relational Operators	&&, ||, ! are a few examples of Logical Operators

Q.17 Develop a program by using Relational and Logical operators.

```
    #include <iostream>
    using namespace std;
    int main(void)
    {
    int x = 10;
    int y = 5;
    int z = 12;
    cout << "\n \t LOGICAL OPERATOR";
    cout << "\n \t ================";
    cout << "\n \t" << ((x > y) && (x < z)) << "\n \t ADD
       OPERATOR"<< endl;
    cout << "\n \t" << ((x > y) || (x > z)) << "\n \t OR
OPERATOR"<< endl;
    cout << "\n \t" << !(x < y) << "\n \t NOT OPERATOR" <<
endl;
    return 0;
    }
```

Q.18

Define Assignment operator with examples.

ARITHMETIC ASSIGNMENT OPERATORS: -

In arithmetic assignment operators, the arithmetic operator is combined with the assignment operator. The assignment operator comes to the right of an arithmetic operator. This operator is also known as Compound Assignment operator.

Operators	description
+ = (Addition-assignment)	It adds the right operand to the left operand and assigns the result to the left operand. **Example:** a + = 2 means a = a + 2
- = (Subtraction-assignment)	It subtracts the right operand from the left operand and assigns the result to the left operand. **Example:** a - = 3 means a = a – 3
* = (Multiplication-assignment)	It multiples the right operand with the left operand and assigns the result to the left operand. **Example:** a * = 4 means a = a * 4
/ = (Division-assignment)	It divides the left operand with the right operand and assigns the result to the left operand. **Example:** a / = 4 means a = a / 4

Q.18 Write a program using Assignment operator in initialization of variable and equal to operator in order to compare two variables.

```
    #include<iostream>
    using namespace std;
    int main(void)
    {
          int x=20, y=10;
          cout << "\n \t Assignment vs Equal to Operator";

          cout << "\n \t =================;
          cout << "\n \t x - 20 assignment opt....." << x;
          cout << "\n \t y = 10 assignment opt....." << y;
          cout << "\n \t Equal to opt. result is....." <<
    (x==y);
    return 0;
    }
```

OUTPUT

```
X = 20 assignment opt .... 20
Y = 10 assignment opt..... 10
Equal to opt is ..... 0
```

MCQS

Q.1 CHOOSE THE RIGHT ANSWER:

1. The arithmetic operators are:
   a) Ternary operators b) Unary operators
   c) Binary operators d) none of these
2. The escape sequence for carriage return is:
   a) /t b) /f c) /n **d) /r**
3. The relational operators in C++ are:
   a) 2 b) 4 c) 6 d) 3
4. This one of the following is a logical operator:
   a) = b) != c) == d) !
5. The correct value to return to the operating system upon the successful completion of a program:
   a) -1 b) ! **c) 0** d) Programs do not return a value.
6. The only function all C++ programs must contain:
   a) start() b) system() **c) main()** d) program()
7. This punctuation is used to signal the beginning and end of code blocks:
   a){ } b) -> and <- c) BEGIN and END d) (and)
8. This punctuation ends most lines of C++ code:
   a) . (dot) **b) ; (semi-colon)** c) : (colon) d) ' (single quote)
9. This one of the following is a correct comment:
   a) */ Comments */ b) ** Comment **
   c) /* Comment */ d) { Comment }
10. This of the following is the correct operator to compare two variables:
    a) := b) = c) equal d) ==
11. This of the following is the boolean operator for logical-and:
    a)& **b) &&** c) | d) |&
12. By default, the standard output device for C++ programs is:
    a) Printer **b) Monitor** c) Modem d) Disk
13. By default, the standard input device for C++ program is:
    a) Keyboard b) Mouse c) Scanner d) none of these
14. This of the following escape sequence represents tab:
    a) \t b) \t\r c)\b d)\a
15. This of the following is called insertion/put to operator:
    a) << b) >> c)> d)<
16. This of the following is called extraction/getfrom operator:
    a) << **b) >>** c)> d)<
17. This function is used to read a single character from the console in C++:
    a) getch() b) getline() c) read() d) cin()
18. These are the escape sequences:
    a) Set of characters that convey special meaning in a program
    b) Set of characters that whose use are avoided in C++ programs
    c) Set of characters that are used in the name of the main function of the program
    d) Set of characters that are avoided in cout statements
19. This of the following escape sequence represents carriage return:
    a) \r b) \n c) \n\r d) \c
20. This of the following is C++ equivalent for scanf():
    a) cin b) cout c) print d) input
21. This of the following is C++ equivalent for printf():
    a) cin **b) cout** c) print d) input

22. **A relational operator:**
a) Compares two operands b) assign one operand to another
c) logically combines two operands d) adds two operands.

23. **This logical operator is unary operator:**
a) // **b) !** c) && d) both a and b

24. **~~This~~ ~~of~~ the following is not a valid escape sequence:**
a) \n b) \t c) \b **d) None of these**

25. **Operator have precedence. A precedence determines which operator is:**
a) faster b) take less memory **c) evaluated first** d) takes no arguments

26. **Escape sequence character \0 occupies __________ amount of memory.**
a) 0 **b) 1** c) 2 d) 4

27. **A cin stops during extraction of data:**
a) By seeing a blank space b) By seeing ()
c) By swaping () d) None of these

28. **Assignment operation will take place in this direction:**
a) left to right **b) right to left** c) top to bottom d) bottom to top

29. **Pick out the compound assignment statement.**
a) a = a – 5 b) a = a / b **c) a -= 5** d) a = a + 5

30. **The name of | | operator is:**
a) size of **b) or** c) and d) modulus

31. **This/these arithmetic operators is offered by C++:**
a) Negate - b) Divide / c) Remainder % **d) all of them**

32. **This C++ operator associativity starts from the right side:**
a) Add Operator + **b) Simple assignment operator =**
c) Subtraction operator - d) Bit shift operator <<

33. **Unary operator means that:**
a) Operator takes only one operand b) Operator takes two operands
c) Operator takes no value d) none of them

34. **This from the following is not a relational operator:**
a) == b) >= **c) !=** **d) none of them**

35. **This from the following is a relational operator:**
a) + b) - **c) <** d) ÷

36. **Operators which specify the relation between two variables by comparing them are known as:**
a) Logical Operators b) Assignment Operators
c) Arithmetic Operators **d) Relational Operators**

37. **'==' Operator is known as:**
a) Equal to operator b) Assignment Operators
c) Arithmetic Operators d) Equal Equal Operators

38. **In CPP, cin and cout are the predefined stream:**
a) Operator b) Functions **c) Objects** d) Data types

39. **Not equal to' operator is represented by this in C++:**
a) != b) /= c) Not = d) \=

40. **'Less than or equal to' is represented by this in C++:**
a) <= b) =< c) => d) >=

41. **This value is returned if the comparison of two variable is True:**
a) 1 b) 4 c) 0 d) 2

42. **This is how the value of variable B is equated to a variable A:**
a) A=B **b) A==B** c) A=C d) A==D

43. This one of the following is the correct syntax to print the message in C++ Language:
a) cout<<"Hello world!"; b) Cout<< Hello world!
c) Out <<"Hello world!; c) none of these

44. This function is used to read character as you type:
a) getchar() b) getch() c) getche() c) Both (b) and (c)

45. cin stands for:
a) Console input b) Console output c) Console stream c) none of these

46. Preprocessor Directives are used for:
a) Macro Expansion **b) File Inclusion**
c) Conditional Compilation c) all of these

47. By default, a function returns a value of type:
a) int b) char c) void c) none of these

48. Pick the odd one out from the following:
a) gets() **b) getline()** c) getch() d) getchar()

49. ____________ does not print the character entered.
a) getche() b) gets() **c) getch()** d) getchar()

50. The __________ string function can be used to read multiple words from the user.
a)getc() **b) gets()** c)getchar() d)getche()

EXERCISE

A. ENCIRCLE OR CHOOSE THE CORRECT ANSWER:

1. The C++ header file ___________ contains the function prototype of the standard Input and Output functions
a) <iomain.h> **b) <iostream>** c) <iostream.h> d) <cstdio.h>

2. This operator is used for input stream:-
a) > b) << c) >> d) <

3. gets stands for:
a) get stream **b) get string** c) get str d) get std

4. getch() and getche() are included in ___________ header file.
a) <csdtio.h> **b) <conio.h>** c) <stdlib.h> d) <stdio.h>

5. This operator is used for logical AND operations:
a) & **b) &&** c) || d) !

6. This one of the following operators is correct to compare two value of variables:
a) = b) <= c) == **d) both b and c**

7. This one of the following needs pressing Enter from the keyboard:
a) getch() b) getche() **c) getchar()** d) gets()

8. != operator belong to which type of operator.
a) Relational b) Logical c) Arithmetic d) none of these

9. This operator adds the first operand to the second operand and gives the result to the first operand:
a) *= **b) +=** c) ++ d) +

10. cout<< 12-6/2; the result on the screen will be:
a) 3haracter b) 6 **c) 9** d) 12

B. RESPOND THE FOLLOWING:

Q.1 Use \a and \r both escape sequence in a program.

Ans. \a
"a" means Alert or alarm. It causes a beep sound in the computer.
Example: -
cout<< "\a";
\r: -

Carriage Return "r". It is used to position the cursor to the beginning of the current line.

Example :-

cout<< "\r";

Q.2 How many types of comments statement are used in C++?

Ans. **Comment Statement in C++:-**

A **comment** is an explanation or description of the source code of the program. It helps a developer explain the logic of the code and improves the program readability. At run-time, a comment is ignored by the compiler.

In C++, there are two types of comment statements:

1. Single Line Comment
2. Multi Line Comment

Single-line Comments

Single-line comments start with two forward slashes (//).

Any text between // and the end of the line is ignored by the compiler (will not be executed).

This example uses a single-line comment before a line of code:

Example

```
// This is a comment
cout<< "Hello World!";
```

This example uses a single-line comment at the end of a line of code:

Example

```
cout<< "Hello World!"; // This is a comment.
```

C++ Multi-line Comments

Multi-line comments start with /* and ends with */.

Any text between /* and */ will be ignored by the compiler:

Example

```
/* The code below will print the words Hello World!
to the screen, and it is amazing */
cout<< "Hello World!";
```

Q.3 Differentiate between Arithmetic operators and Relational operators.

Arithmetical Operator	Relational Operator
Arithmetic operators are used to perform mathematical operations.	Relational operators are used to compare the values of two expressions. Relational operators are binary operators because they require two operands to operate.
+, -, *, /, %,are examples of Arithmetic operators.	<,<= >,> =,== ,!= are examples of Relational Operators

Q.4 Write a program in C++ and use all arithmetic assignment operators.

Ans.
```
#include <iostream>
using namespace std;
int main()
{
int x = 5;
cout<< "Initial value of x is " << x << "\n";
x += 5;
cout<< "x += 5 gives :" << x << "\n";
x -= 5;
cout<< "x -= 5 gives : " << x << "\n";
```

```
    x *= 5;
    cout<< "x *= 5 gives :" << x << "\n";
    x /= 5;
    cout<< "x /= 5 gives : " << x << "\n";
    return 0;
}
```

Q.5 What is the difference between Assignment operator and Equal to operator?

S.NO	Assignment Operator (=)	Equal Operator (==)
1.	= is an Assignment Operator.	== is a comparison operator.
2.	= operator is used to assign value to a variable.	== operator is used to compare two variables or constant.
3.	The left side of = operator cannot be a constant.	In == operator, both sides can be operators.
4.	The operator assigns the value of right-side expression to the left-side variable. Such as x=10;	The operator compares the value of the left side and right-side expression. Such as x=10 and y=10 than x==y If the condition true otherwise false.

Q.6 What is the basic difference between \n and \t?

S.NO	\n	\t
1.	\n is the new line character.	\t is the tab character
2.	It takes the cursor to next line.	It moves the cursor to a couple spaces to the right in the same line.
3.	\n is for vertical tab.	\t is for horizontal tab.
4.	Example: cout<< "\n";	Example: cout<< "\t";

Q.7 Get the output of the following program

Ans.
```
#include <iostream>
using namespace std;
int main(void)
{
        int a=27;
         cout<< "a is " << a <<endl;
         cout<< "a is now" << a++ <<endl
         cout<< "a is now" << a <<endl;
         cout<< "a is now" << --a <<endl
         cout<< "a is now" << a <<endl;
retrun 0
}
```

Output of the Program:

```
a is     27
a is now  27
a is now 28
a is now 27
a is now 27
```

CONTROL STRUCTURE

Q.1 WHAT IS CONTROL STATEMENT?

Control Statements in c + + :-

In C++, **Control Statements** are usually jumped from one part of the C++ code to another depending on whether a particular condition is satisfied or not. There are three types of C++ Control Statements. These are as follows:

1. Sequence Statement
2. Selection Statement
3. Loop Statement

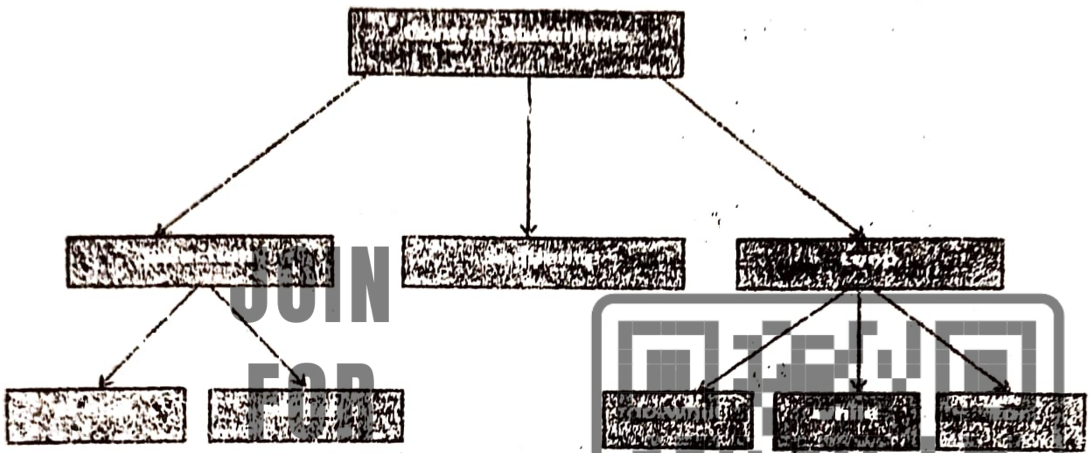


Fig: Control Statements in C++

Q.2 WHAT IS THE FUNCTION OF IF STATEMENT EXPLAIN WITH EXAMPLE.

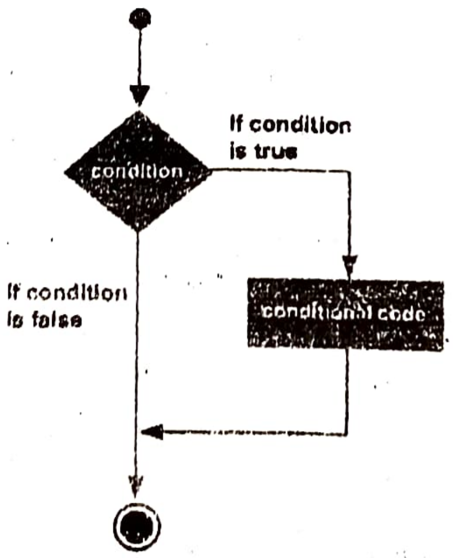
"if" statement:-

It is the basic decision statement. The structure contains "if" keyword followed by a conditional expression in parenthesis and then its body of statements(s),also called if block. If there are more than one statement in "if" block or body we enclose all of them in braces. If statement checks a condition, if the condition is true, the statement in "if" block are executed and if the condition is false, it leaves the statement in "if" block and starts executing statements after "if" block.

Syntax:-

```
If (test condition)
{
Statement(s);
}
```

Flow Diagram:-



Example:-

```
// Program to print positive number entered by the user
// If the user enters a negative number, it is skipped
#include <iostream>
usingnamespacestd;
intmain() {
int number;
cout<<"Enter an integer: ";
cin>> number;
// checks if the number is positive
if (number >0) {
cout<<"You entered a positive integer: "<< number <<endl;
   }
cout<<"This statement is always executed.";
return0;
}
```

Output:-

```
Enter an integer: 5
You entered a positive number: 5
This statement is always executed.
```

Q.3 WHAT IS SELECTION /DECISION MAKING STRUCTURE?

Selection/Decision Making Structure:-

Decision making structures require that the programmer specify one or more conditions to be evaluated or tested by the program, along with a statement or statements to be executed if the condition is determined to be true, and optionally, other statements to be executed if the condition is determined to be false.

They are used to decide whether a certain part of code is executed or not.

C++ has three decision making structure;

1. "if" statement
2. "if-else" statement
3. "switch" statement

Q.4 DEFINE NESTED IF STATEMENT AND HOW IT WORKS?

Nested "if" Statement:-

A nested if statement is an if statement placed inside another if statement. The inner "if" statement will only be tested if the outer "if" is true. Nested if statements are often used when you must test a combination of conditions before deciding on the proper action.

Q.5 WRITE AN EXAMPLE PROGRAM FOR NESTED IF STATEMENT.

Example Of Nested "If" Statement:-

In following program, marks and age of a person are taken as input. First condition checks marks, if they are greater than or equal to 60 then next condition checks age. If age is also greater than 18 then it prints message "you got the job". "good luck" is always printed.

```
/* Program  .    Nested "if" statement example  */
// job given message on basis of marks and age
#include<stdio.h>
#include <iostream.h>
using namespace std;
int main(void)
{
      int marks,age;

      cout<<"\nEnter your Marks ";
      cin>>marks;
      cout<<"\nEnter your age ";
      cin>>age;

      if(marks>=60)
            {
            if(age>=18)
                  {
                  cout<<"you got the job";
                  }
            }
      cout<<" Good Luck";
return 0;
}
```

Q.6 WHAT IS IF-ELSE STATEMENT? HOW DOES IT WORK?

"if-else" statement:-

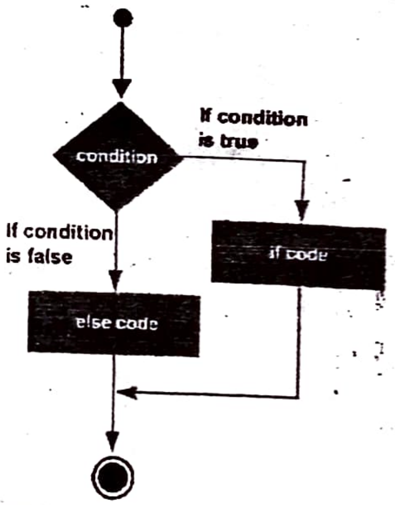
Purpose:-

The if-else statement is used to execute both the true part and the false part of a given condition. If the condition is true, the if block code is executed and if the condition is false, the else block code is executed.

Syntax:-

```
If (test condition)
{
Statement(s);
}
else
{
Statement(s);
}
```

Flow Diagram:-



Q.7 WRITE AN EXAMPLE PROGRAM FOR "IF-ELSE" STATEMENT.

Example of "if-else" statement:-

The following program takes marks as input and decides pass or fail on the basis of marks. If greater than or equal to 40 then pass otherwise fail.

```
/* Program    "if-else" statement example */
// shows pass or fail on the basis of marks
#include<stdio.h>
#include <iostream.h>
using namespace std;
int main(void)
{
        int marks;
        cout<<"\nEnter your Marks ";
        cin>>marks;

        if(marks>=40)
                cout<<"You are pass";
        else
                cout<<" You are fail";
return 0;
}
```

Q.8 WHAT IS THE DIFFERENT BETWEEN IF, IF-ELSE AND SWITCH STATEMENTS?

"if"	**"if"** statement checks a condition using relational and other operators, if the condition is false, it leaves the statements in "if" block execute. If condition is false, it leaves the statements after "if" block.
"if-else"	**"if-else"** statement checks a condition using relational and other operators, if the condition is true, the statements in "if" block are executed and in case of false condition it executes statements in 'else' block. It means either "if" block statements will execute or "else" block statements.
"switch"	**"switch"** statement checks 'switch' variable value equal to constant that follows case statement. If it matches any of them, then control is transferred to that case and all statement after colon are executed and switch is broken with **"break"** statement. Otherwise, control is transferred to **"dafault"** statement. It has some limitations. It only matches character and integer data type variables, it checks switch variable value only with case constant not with any variable and it also cannot use relational operators like less than (<) or greater than (>) and exactly matches value with case constant.

Q.9 HIGHLIGHT THE FUNCTION OF ITERATION/LOOP.

Function Of Iteration/Loop:-

Normally, statements are executed sequentially, one after the other. In some situations, we need to execute a block of statement several number of times. Loops allow us to execute a statement or a group of statement several numbers of times. A group or block is made by enclosing statements in braces. Repeating a block of statement is a very common and useful task in programming. Loops make the task of writing programs easier and efficient.

Three types of loops are used in C++ programs. These are for, while, and do while loops.

Q.10 WHAT IS LOOP? LIST THE ESSENTIAL ELEMENTS OF A LOOP. MAY THE WHY PROGRAMMER WANT TO EXECUTE A LOOP?

LOOP:-

A loop is a statement in a programming language that allows one or more statements to be repeatedly executed as many times as required.

ESSENTIAL ELEMENTS OF A LOOP:-

There are two essential elements of a loop. The block of statements forms the body of loop that is to be executed repeatedly until loop condition is true. Loops terminate based on test conditions.

PROGRAMMER MIGHT WANT TO EXECUTE A LOOP:-

i. A given number of times.
ii. Until a given value exceeds another value.
iii. Until a particular character is entered.

Q.11 WRITE THE SYNTAX, PURPOSE AND EXAMPLE OF "FOR" LOOP.

PURPOSE:-

A for loop is a repetition control structure which allows us to write a loop that is executed a specific number of times. The loop enables us to perform n number of steps together in one line. For loop executes the statements of program several times repeatedly until a given condition returns false.

SYNTAX:-

```
for (initialization expression; test expression; update expression)
{
// statements to execute in the loop body
}
```

EXAMPLE:-

The following program shows even numbers from two to twenty. It initializes loop variable count with two and then prints numbers by adding two each time to this variable until count variable remain less than twenty.

```
/* Program . "for" loop example */
// shows even numbers from 1 to 20
#include<stdio.h>
#include <iostream.h>
using namespace std;
int main(void)
{
      int count;


      for(count=2;count<=20;count=count+2)
            {
            cout<<"\n Number= ";
            cout<<count;
            }
return 0;
}
```

Q.12 DEFINE THE "WHILE" LOOP.

"while" loop:-

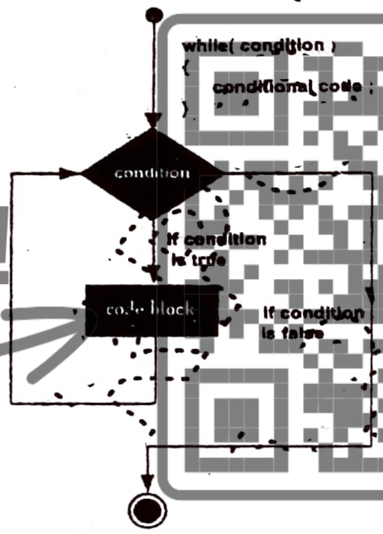
It is a pre-test loop. It tests the condition at the start of loop before executing the body of loop. While loops statement allows to repeatedly run the same block of code until a condition is met.

While loop has one control condition, and executes as long the condition is true. The condition of the loop is tested before the body of the loop is executed; hence it is called entry-controlled loop.

Syntax:-

```
While (condition)
{
  statement(s);
  Incrementation;
}
```

FLOW DIAGRAM:-



Q.13 WRITE AN EXAMPLE PROGRAM OF "WHILE" LOOP.

Example Of "While" Loop:-

```
#include<iostream>
usingnamespace std;

int main (){
// Local variable declaration:
int a =10;

// while loop execution
while( a<20){
cout<<"value or a: "<< a <<endl;
    a++;
}
return0;
}
```

OUTPUT:-

When the above code is compiled and executed, it produces the following result –

```
value of a: 10
value of a: 11
value of a: 12
value of a: 13
value of a: 14
value of a: 15
value of a: 16
value of a: 17
value of a: 18
value of a: 19
```

Q.14 DEFINE "DO-WHILE" LOOP.

"do-while" loop:-

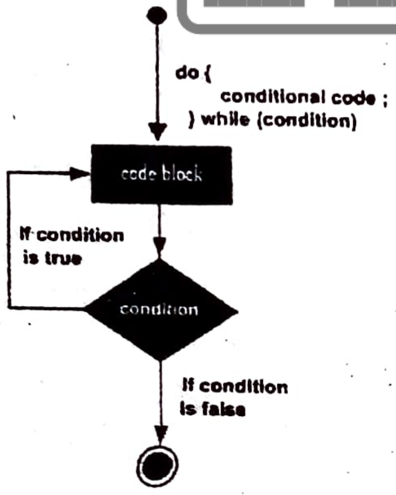
The do-while loop iterates a section of the C++ program several times. In the do-while loop, test expression is added at the bottom of the loop. The loop body comes before the test expression. That is why the loop body must execute for once, even when test expression evaluates too false in the first test.

Syntax:-

The syntax of a do...while loop in C++ is –

```
do {
  statement(s);
}
while(condition);
```

FLOW DIAGRAM:-



Q.15 WRITE AN EXAMPLE PROGRAM OF "DO-WHILE" LOOP.

Example of "do-while" loop:-

Following program takes salaries of employees as input in a loop. We press y to take more salaries as input, any other character to end. Finally, it shows total salary paid to all employees.

```
/* Program  . "do-while" loop example */
// calculates total salary paid to all employees
#include<stdio.h>
#include <constream.h>
using namespace std;
int main(void)
{
        long count=1;
        float tot_sal=0,salary;
        char ch;

        clrscr();
        do
        {
        cout<<"\n Enter salary of employee e"<<count<<[illegible]
        cin>>salary;
        tot_sal=tot_sal+salary;
        cout<<" Press Y to enter more salaries
        ch=getche();
        count++;
        }
        while(ch=='y');

        cout<<"\n Total salary paid is."<<tot_sal
return 0;
}
```

Q.16 EXPLAIN THE CONCEPT NESTED LOOP

Nested Loop:-

A nested loop is a loop within a loop, an inner loop within the body of an outer one. The inner nested loop is completely executed every time for each repetition of outer loop.

Syntax of Nested loop

```
Outer_loop
{
   Inner_loop
   {
      // inner loop statements.
   }
    // outer loop statements.
}
```

Outer loop and Inner loop are the valid loops that can be a 'for' loop, 'while' loop or 'do-while' loop.

Q.17 WRITE AN EXAMPLE PROGRAM FOR "NESTED" LOOP.

Example of "Nested" loop:-

The following program is nested for loop. It prints numbers from 1 t0 10 in five rows. The outer loop shows row number, then inner loop prints numbers from 1 to 10 in one row, then outer loop changes the row using '\n'. The outer loop runs five times, so inner loop which points 1 to 10 numbers will run five times and print numbers in five rows.

```
/* Program  .   nested loops example */
// prints numbers from 1 to 10 in five rows
#include<stdio.h>
#include <iostream.h>
using namespace std;
int main(void)
{
      int i,j;

      clrscr();
      for(i=1;i<=5;i++)
            {
            cout <<" Row no. "<<i<<" -> ";
            for(j=1;j<=10;j++)
                  cout<<j<<"  " ;
            cout<<"\n";
            }
return 0;
}
```

Q.18 DIFFERENTIATE BETWEEN FOR, WHILE AND DO-WHILE LOOP STRUCTURES.

"for" Loop	**"while" Loop**	**"do while" Loop**
"for" Loop is usually used in situations where we know at the start of Loop how many times loop block will execute. It is also called definite repetition loop.	It is usually used in situations where we do not know at the start of loop how many times loop block will execute. It is also called indefinite repetition loop.	It is usually used in situations where we do not know at the start of loop how many times loop block will execute. It is also called indefinite repetition loop.
It is called counter controlled loop as loop is controlled by a counter value, at each iteration counter value will increase or decrease.	It does not need counter value for its execution.	It does not need a counter value for its execution.
It has three parts: first initialization, second condition testing and third increment of decrement.	It has only one part which is condition testing. If needed, initialization is done before loop and increment or decrement is done in loop body.	It has only one part which is condition testing. If needed, initialization is done before loop and increment or decrement is done in loop body.
It is pre-test loop as condition is tested at the start of loop.	It is pre-test loop as condition is tested at the start of loop.	It is post-test loop as condition is tested at the end of loop.

Q.19 DIFFERENTIATE BETWEEN WHILE AND DO-WHILE LOOP.

S. No	While	Do-while
1.	While loop is entry control loop.	do-while loop is exit control loop.
2.	A while loop is a pre-test loop.	It tests the condition at the end of the loop body.
3.	It tests the condition before executing the loop body.	It tests the condition at the end of the loop body.

4.	There is no semicolon at the end of the loop.
	Syntax:
	while(condition)
	{
	statement(s);
	}
		The semicolon is compulsory at the end of the loop.
		Syntax:
		do
		{
		statement(s);
		}
		while(condition);

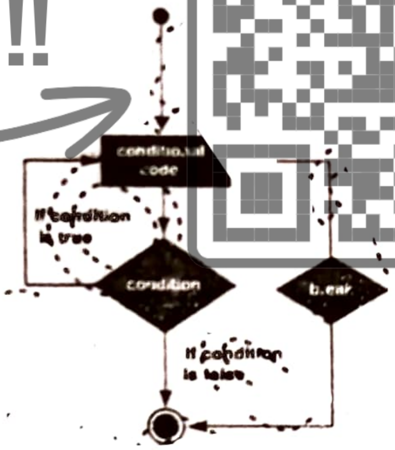
Q.20 DEFINE "BREAK" STATEMENT.

"break"statement :-

Break Statement is a loop control statement that is used to terminate the loop. As soon as the break statement is encountered from within a loop, the loop iterations stop there, and control returns from the loop immediately to the first statement after the loop.

Syntax:- break;

Flow Diagram:-



Q.21 WRITE AN EXAMPLE PROGRAM FOR "BREAK" STATEMENT.

Example of "break" statement :-

In the example below, we have a while loop running from 10 to 200 but since w have a break statement that gets encountered when the loop counter variable value reache 12, the loop gets terminated and the control jumps to the next statement in program afte the loop body.

```
#include<iostream>
usingnamespace std
intmain(){
int num =10,
while(num<=200) {
cout<<"Value of num is: "<<num<<endl;
if (num==12) {
break;
  }
```

```
    num++;
  }
cout<<"Hey, I'm out of the loop";
return0;
}
```

Output:

```
Value of num is: 10
Value of num is: 11
Value of num is: 12
Hey, I'm out of the loop
```

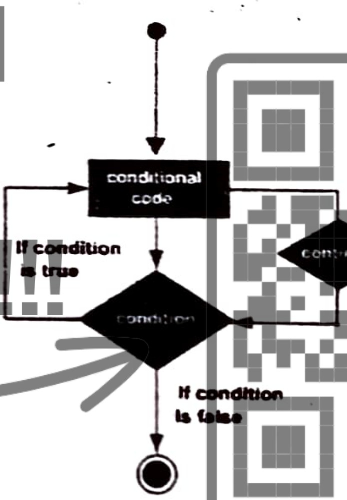
Q.22 Define "continue" statement.

"continue"statement :-

Continue statement is used inside loops. Whenever a continue statement is encountered inside a loop, control directly jumps to the beginning of the loop for next iteration, skipping the execution of statements inside loop's body for the current iteration.

Syntax :- continue;

Flow Diagram :-



Q.23 WRITE AN EXAMPLE PROGRAM FOR "CONTINUE" STATEMENT.

Example of "continue" statement :-

Consider the situation when you need to write a program which prints numbers from 10 to 19 and but not 15. It is specified that you have to do this using loop and only one loop is allowed to be used.

Here comes the usage of continue statement. What we can do here is we can run a loop from 10 to 19 and every time we have to compare the value of iterator with 15. If it is equal to 15, we will use the *continue* statement to continue to next iteration without printing anything otherwise we will print the value.

Below is the implementation of the above idea:

```
#include<iostream>
usingnamespace std;

int main (){
// Local variable declaration:
int a =10;

// do loop execution
do{
if( a==15){
```

```
// skip the iteration.
      a = a +1;
continue;
}
cout<<"value of a: "<< a <<endl;
    a = a +1;
}
while( a<20);

return0;
}
```

Output:

When the above code is compiled and executed, it produces the following result –

```
value of a: 10
value of a: 11
value of a: 12
value of a: 13
value of a: 14
value of a: 16
value of a: 17
value of a: 18
value of a: 19
```

Q.24 Define "goto" statement.

"goto" statement :-

A goto statement provides an unconditional jump from the goto to a labeled statement in the same function.

Syntax :-

The syntax of goto statement looks like this;

goto label_name;

Q.25 Write an example program for "goto" statement.

Example of "goto" statement :-

```
#include<iostream>
usingnamespace std;
intmain(){
int num;cout<<"Enter a number:";cin>>num;
if(num %2==0){
gotoprint;
}
else{
cout<<"Odd Number";
}
print:
cout<<"Even Number";
return0;
}
```

Output:-

```
Enter a number:42
EvenNumber
```

Q.26 Define "return" statement.

"return"statement:-

Terminates the execution of a function and returns control to the calling function (or to the operating system if you transfer control from the main function). Execution resumes in the calling function at the point immediately following the call.

Syntax:-

```
return [expression];
```

Q.27 WRITE AN EXAMPLE PROGRAM FOR "RETURN" STATEMENT.

Example of "return" statement :-

```
#include <iostream>
using namespace std;
int myFunction(int x) {
  return 5 + x;
}
int main() {
cout<<myFunction(3);
 return 0;
}
```

Q.28 Define "exit" function.

"exit()"function :-

Theoretically, the exit() function in C++ causes the respective program to terminate as soon as the function is encountered, no matter where it appears in the program listing. The function has been defined under the "stdlib.h" header file, which must be included while using the exit() function.

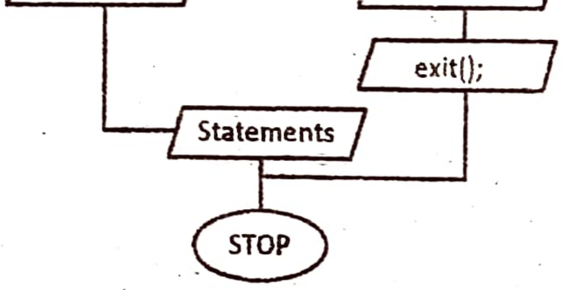
Syntax for the exit() function

The syntax for using the exit() function is given below,

```
exit( exit_value );
```

Here, exit_value is the value passed to the Operating system after the successful termination of the program. This value can be tested in batch files where ERROR LEVEL gives us the return value provided by the exit() function. Generally, the value 0 indicates a successful termination and any other number indicates some error.

Flow Diagram:-



Q.29 WRITE AN EXAMPLE PROGRAM FOR "EXIT()" FUNCTION.

Example of "exit()" function:-

```
include<iostream>
isingnamespacestd;
ntmain()

  inti;
  cout<<"Enter a non-zero value: ";  //user input
  cin>>i;
  if(i)   // checks whether the user input is non-zero or not
  {
     cout<<"Valid input.\n";
  }
  else
  {
     cout<<"ERROR!";  //the program exists if the value is 0
     exit(0);
  }
  cout<<"The input was : "<<i;
```

Output:

```
Enter a non-zero value: 0
ERROR!
```

MULTIPLE CHOICE QUESTIONS

Q.30 Choose The Right Answer:

1. **This of the following can replace a simple if-else construct:**
   a) Ternary operator b) while loop c) do-while loop d) for loop
2. **This/These of the following is an entry-controlled loops:**
   a) do-while loop b) while loop c) for loop d) Both (b) and (c)
3. **This/These of the following is/are most suitable for a menu-driven program:**
   a) do-while loop b) while loop c) for loop d) All of these
4. **A switch construct can be used with this type of variable:**
   a) int b) int, char c) int, float, char d) Any basic data type
5. **This/These of the following must be present in switch construct:**
   a) Expression in ( ) after switch b) default
   c) case followed by value d) All of these
6. **The effect of writing a break statement inside a loop:**
   a) It cancels remaining iterations. b) It skips a particular iteration.
   c) The program terminates immediately. d) Loop counter is reset.
7. **The effect of writing a continue statement inside a loop?**
   a) The program terminates immediately. b) It cancels remaining iterations.
   c) It skips execution of statements which are written below it. d) Loop counter is reset
8. **If the variable count exceeds 100, a single statement that prints "Too many" is:**
   a) if (count<< "Too many"; b) if (count>100) cout>> "Too many";
   c) if (count>100) cout<< "Too many"; d) none of these.
9. **The break statement causes an exit:**
   a) from the innermost loop only. b) only from the innermost switch.
   c) from all loops & switches. d) from the innermost loop or switch.

10. for (; ;)
a) means the test which is done using some expression is always true
b) is not valid c) will loop forever d) should be written as for( )

11. **The switch statement is also called as:**
a) choosing structure b) selective structure c) certain structure d) none of these

12. **The destination statement for the goto label is identified by this label:**
a) $ b) @ c) * d) :

13. **The break statement in repetition structure causes a program to:**
a) immediately exit b) terminate
c) again start the loop from 1 d) all of these

14. **Specifying the order in which statements are to be executed in a program is called:**
a) Program Control b) Program Order c) Program Counter d) Program Flow

15. **if you have to make decision based on multiple choices, which of the following is/are best suited:**
a) if b) if-else c) if-else-if d) all of these

16. **In situations where we need to execute the body of the loop before testing the condition, we should use:**
a) For loop b) while loop c) do-while loop d) nested for loop

17. **Loops in C++ Language are implemented using:**
a) While loop b) For loop c) Do while loop d) All of these

18. **This loop is faster in C++ Language, for, while or do-while:**
a) For b) while c) do-while d) All work at the same speed

19. **This is the way to suddenly come out of or quit any loop in C++:**
a) continue; statement b) break; statement
c) leave; statement d) quit; statement

20. **The correct syntax of for loop is:**
a) for (initialization;condition; increment/decrement)
b for(increment/decrement; initialization; condition)
c) for(initialization, condition, increment/decrement
d) none of these

21. **This looping process is best used when the number of iterations is known:**
a) While loop b) For loop c) Do while loop
d) all looping processes require that the iterations be known

22. **The purpose of 'default' keyword in C++:**
a) It instruct the compiler to set default value to any variable
b) It instruct the compiler to set default return value to any function
c) It instruct the compiler to set default argument values for any function
d) It instruct the compiler to generate the default implementation

23. **The statement structure that is NOT a looping is:**
a) For b) Do…While c) If d) While

24. **A process of repeating the workflow is called:**
a) Branching b) Looping c) Return d) Algorithm

25. **Each pass through a loop is called a/an:**
a) enumeration b) iteration c) culmination d) pass through

26. **If there is more than one statement in the block of a for loop, this/these of the following must be placed at the beginning and the ending of the loop block:**
a) parentheses ( ) b) braces { } c) brackets [ ] d) arrows <>

27. **The statement i++; is equivalent to**
a)i = i + i; b)i = i + 1; c)i = i - 1; d)i --;

28. In a group of nested loops, this loop is executed the most number of times:
a) the outermost loop b) the innermost loop
c) all loops are executed the same number of times
d) cannot be determined without knowing the size of the loops

29. This/These is/are true about the loop structure:
a) body of the loop b) condition
c) loop control variable d) all of these

30. ____________ is not a loop structure.
a) while *b) do while* *c) switch* *d) for*

31. Statements executed at least once in this structure:
a) while b) do while c) for d) none of these

32. This of the following loop inside another loop is called:
a) bounded loop b) nested loop c) internal loop d) inner loop

33. Statements can only be used inside the body of loop called:
a) if b) break c) continue d) switch

34. This/These of the following option is exit due to library function exit():
a) the function in which it occurs b) the loop in which it occurs
c) the program in which it occurs d) all of these

35. Switch statement accepts:
a) int b) char c) long d) all of these

36. do-while loop terminates when conditional expression returns?
a) One b) Zero c) Non zero d) none of these

37. The for loop is controlled by this/these many separate parts.
a) 1 b) 2 c) 3 d) 4

38. This from the following is not a part of the for statement:
a) Initialization b) Statement checker
c) Continuation condition d) Update

39. If a single statement is written after if condition, braces {} are:
a) Mandatory b) Optional
c) Depends on compiler d) If code is inside, then optional

40. These brackets are used for the body of "if" statement:
a) () b) {} c) <> d) []

41. These of the following is valid case in switch:
a) case 1: b) case "true" c) case "false" d) case 1.5:

42. Choose correct statements about C++ switch construct:
a) default case is optional b) use of break is optional d) all of these
c) break; causes the control to exit the switch immediately and valid fall down to other CASE statements

43. The break statement in reputation structure causes a program to:
a) immediately exit b) terminate
c) again start the loop form 1 d) all of these

44. Within a switch statement:
a) Continue can be used but Break cannot be used
b) Continue cannot be used but Break can be used
c) Both Continue and Break can be used
d) Neither Continue nor Break can be used

45. This/These is/are the alternative to SWITCH in C++ language:
a) If b) Else c) If-Else d) all of these

46. In the switch case block this/these keywords is/are used to cover unhandled possibilities:
a) break b) default c) continue d) all

47. An IF-ELSE statement is also called:
a) Branching statement b) Control statement
c) Block statements d) all of these

48) An IF or ELSE IF statement accepts ______________ as input before branching.
a) Boolean b) int c) float d) char

49) An IF statement in Java is also a/an ___ statement.
a) Boolean b) conditional c) iterative d) optional

50) In which loop condition is placed at the end?
a) for loop b) do while loop c) while loop d) nested loop

51) This structure enables the programmer to execute a set of instructions repeatedly until a particular condition is met:
a) selection b) sequence c) choice d) loop

52) This of the following is also called counter loop:
a) do while b) while c) for d) if else

53) A pass through a loop is called:
a) execution b) iteration c) enumeration d) communication

EXERCISE

A. Encircle or choose the correct answer:

1. **Loop within a loop is called ______________ loop.**
a) inner b) outer c) enclosed d) nested
2. **"case" and __________ are also parts of "switch"statement.**
a) have b) default c) for d) if
3. **"for" Loop expression has ______________ part.**
a) one b) two c) three d) four
4. **exit() function is used to ___________ in header file.**
a) close function b) close loop c) close program d) close switch
5. **"continue" statement takes control to the:**
a) top of loop b) end of loop c) top of function d) end of function
6. **In "goto" statement, label is followed by________ character.**
a) colon(:) b) semi colon (;) c) single quote (') d) double quote(")
7. **To send value to the calling function, we use __________ statement.**
a) throw b) return c) send d) back
8. **"break" statement is used with:**
a) if b) switch c) for d) while
9. **Using "else" is __________ with "if" statement.**
a) prohibited b) advised c) compulsory d) optional
10. **"if" and loop expression use _________ operators to test condition.**
a) arithmetic b) relational c) insertion d) bitwise

B. Answer the following questions:

Q.1 What is the purpose of "default" statement in C++?
"default" Statement in C++: -

The switch and case keywords evaluate expression and execute any statement associated with constant-expression whose value matches the initial expression.

If there is no match with a constant expression, the statement associated with the default keyword is executed. If the default keyword is not used, control passes to the statement following the switch block.

Q.2 Can we use "while" loop in place of "for" loop, if yes then how?

The answer is **YES**. But in most of the cases it is advisable to use the loops appropriately. All for loops can be written as while loops, and vice-versa. Just use whichever loop seems more appropriate to the task at hand.

In general, you should use a for loop when you know how many times the loop should run. If you want the loop to break based on a condition other than the number of times it runs, you should use a while loop.

Q.3 What is the main difference while and do while loops?

The main difference between a while loop and do while loop is that while loop checks the condition before iteration of the loop, the do-while loop verifies the condition after the execution of the statement inside the loop.

Q.4 Write the function of for loop.

For Loop :-

for loop executes the statements of program several times repeatedly until a given conditions return false.

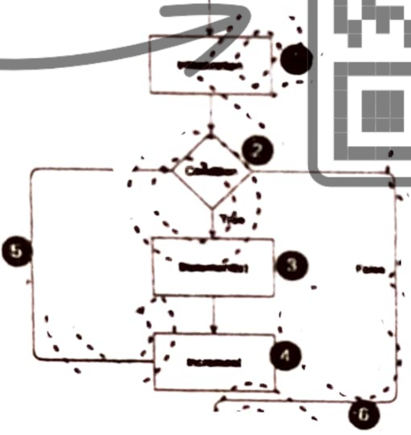
Syntax :-

```
for (initialization; condition; incr/decr)
{
Statement (s);
}
```

How For Loop Works :-

1. The initialization statement is executed only once at the beginning.
2. Then, the test expression is evaluated.
3. If the test expression is false, for loop is terminated. But if the test expression is true, codes inside the body of for loop is executed and update expression is updated.
4. Again, the test expression is evaluated and this process repeats until the test expression is false.

Flow Diagram :-



Q.5 Why do we make block of statements using braces?

In order to make more than one statement be controlled by a decision, we use a block statement. Curly braces {} are used to define a block. All statement within the braces are treated as a unit. A block can be used anywhere a single executable statement is expected. This permits us to "nest" our statement in any combination. Curly braces make the block less error-prone to future modifications by others. Curly braces make the code more organized in case of multiple if-else statements.

Q.6 Which data type variables can be used in "switch" statement?

The Variable used in a switch statement can only be a byte, short, int, or char. The values for each case must be of the same data type as the variable type.

Q.7 What is the purpose of jump statements?

Jump statement :-

Jump statements are a type of Control Statements used to interrupt the normal flow of the program. It makes the program jump to another section of the program unconditionally when encountered. It can also be used to terminate any loop. The jump statements defined in C++ are break, continue, goto and return. return and goto used anywhere in the program whereas break and continue are used inside smallest enclosing like loops etc. In addition to these jump statements, a standard library function exit () is used to jump out of an entire program.

Q.8 Write the purpose of the following statements:

a. else if

Purpose :- The if...else statement is used to execute a block of code among two alternatives. However, if we need to make a choice between more than two alternatives, we use the else if statement. elseif statement is used when we need to check multiple conditions. In this control structure we have only one "if" and one "else", however we can have multiple "else if" blocks. This is how it looks:

Syntax :-

```
if(condition_1) {
/*if condition_1 is true execute this*/
  statement(s);
}
elseif(condition_2) {
/* execute this if condition_1 is not met and
   * condition_2 is met
   */
  statement(s);
}
elseif(condition_3) {
/* execute this if condition_1 & condition_2 are
   * not met and condition_3 is met
   */
  statement(s);
}
.
.
.
else {
/* if none of the condition is true
   * then these statements get executed
   */
  statement(s);
}
```

Example of if-else-if

```
#include<iostream>
usingnamespace std;
intmain(){
int num;
cout<<"Enter an integer number between 1 & 99999: ";
cin>>num;
if(num <100&& num>=1){
cout<<"Its a two digit number";
}
elseif(num <1000&& num>=100){
cout<<"Its a three digit number";
}
elseif(num <10000&& num>=1000){
cout<<"Its a four digit number";
}
elseif(num <100000&& num>=10000){
cout<<"Its a five digit number";
}
else{
cout<<"number is not between 1 & 99999";
}
return0;
}
```

Output:

```
Enter an integer number between 1&99999:8976
Its a four digit number
```

b. switch

Purpose :-

C++ provides a multiple-branch selection statement known as switch.

Switch case statement is used when we have multiple conditions and we need to perform different action based on the condition. When we have multiple conditions and we need to execute a block of statements when a particular condition is satisfied.

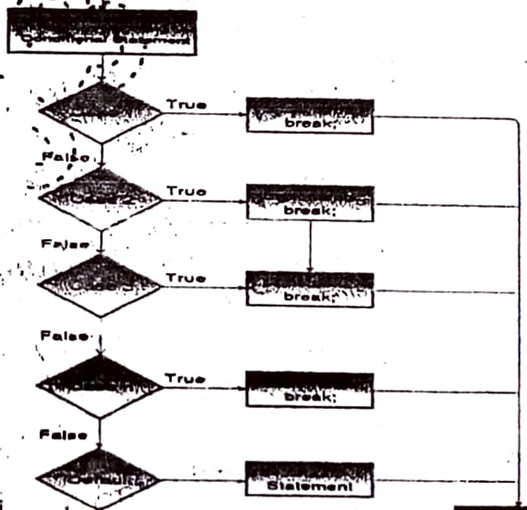
This selection statement successively tests the value of an expression against a list of integer or character constants. When a match is found, the statements associated with that constant are executed.

Syntax :-

The syntax of switch statement is as follows:

```
switch(expression)
{
  case constant1 : statement sequence
      break;
  case constant2 : statement sequence 2;
      break;
  case constant3 : statement sequence 3;
      break;
  .
  .
  .
  case constant n-1 : statement sequence,
      break;
default : statement sequence n;
}
```

Flow Diagram: -

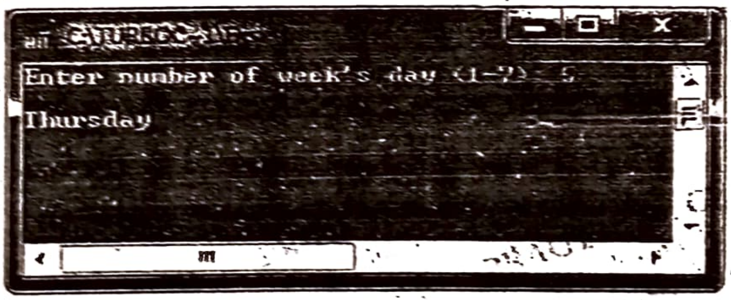


Example :-

```
/* C++ Selection Statement - C++ switch Statement */

#include<iostream.h>
#include<conio.h>

intmain()
{
clrscr();
  int dow;
cout<<"Enter number of week's day (1-7): ";
cin>>dow;
  switch(dow)
  {
    case 1 :cout<<"\nSunday";
      break;
    case 2 :cout<<"\nMonday";
      break;
    case 3 :cout<<"\nTuesday";
      break;
    case 4 :cout<<"\nWednesday";
      break;
    case 5 :cout<<"\nThursday";
      break;
    case 6 :cout<<"\nFriday";
      break;
    case 7 :cout<<"\nSaturday";
      break;
default :cout<<"\nWrong number of day";
      break;
  }
return 0;
}
```

Output :-

When the above program is compiled and executed, it will produce the following output:



```
Enter number of week's day (1-7): 5
Thursday
```

c. goto

Purpose :-

A goto statement provides an unconditional jump from the goto to a labeled statement in the same function.

Syntax

The syntax of a goto statement in C++ is :

goto label;

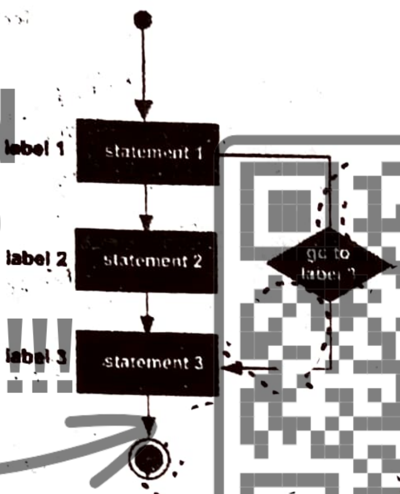
..

.

label: statement;

Where label is an identifier that identifies a labeled statement. A labeled statement is any statement that is preceded by an identifier followed by a colon (:).

Flow Diagram



Example

```
#include<iostream>

usingnamespace std;


int main (){
// Local variable declaration:
int a =10;


// do loop execution
LOOP:do{
if( a==15){
// skip the iteration.
      a = a+1;
goto LOOP
}
cout<<"value of a: "<< a <<endl;
   a = a +1;
}
while( a<20);


return0;
}
```

Output :-

When the above code is compiled and executed, it produces the following result –

```
value of a: 10
value of a: 11
value of a: 12
value of a: 13
value of a: 14
value of a: 16
value of a: 17
value of a: 18
value of a: 19
```

d. exit

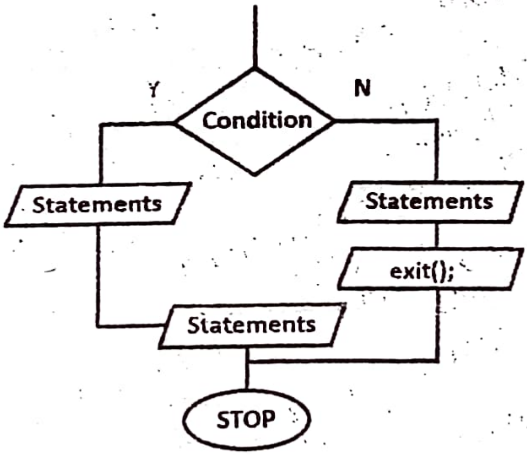
Purpose :-

The exit () is used to terminate a C++ program and close program whenever executed. It is defined under "stdlib.h" header file.

Syntax :-

```
void exit (int);
```

Flow Diagram



Example

```
#include<iostream>
using namespace std;
int main()
{
  int i;
  cout<<"Enter a non-zero value: ";  //user input
  cin>>i;
  if(i)   // checks whether the user input is non-zero or not
  {
    cout<<"Valid input.\n";
  }
  else
  {
    cout<<"ERROR!";  //the program exists if the value is 0
    exit(0);
  }
  cout<<"The input was : "<<i;
}
```

Output

```
Enter a non-zero value: 0
ERROR!
```

Q.3 Match the column:

s. no	A	s. no	B
(a)	If	(i)	Relational operators (c)
(b)	Loop	(ii)	break (d)
(c)	Conditional expression	(iii)	switch (g)
(d)	Loops and switch	(iv)	operator (f)
(e)	do	(v)	iteration (b)
(f)	>>	(vi)	else (a)
(g)	case	(vii)	while (e)

FUNCTIONS 5

Q1. Define function.

Function:

A function in C++ refers to a group of statements that takes input, processes it, and returns an output. The idea behind a function is to combine common tasks that are done repeatedly. If you have different inputs, you will not write the same code again. You will simply call the function with a different set of data called parameters.

Each C++ program has at least one function, the main() function. You can divide your code into different functions. This division should be such that every function does a specific task.

Q2. What are the benefits of functions?

Advantages of Functions

The advantages of using functions are:

- Avoids repetition of codes.
- Increases program readability.
- Divides a complex problem into simpler ones.
- Reduces chances of error.
- Modifying a program becomes easier by using functions.

Q3. What are the types of functions in C++?

Types of Functions

Functions are created by the programmers according to their requirement; C++ compiler has some built-in functions that can be used anytime by the programmer. C++ Programming Language has two types of functions:

- Built-in Functions/library Functions
- User-defined Functions

Q4. Define predefined functions.

Predefined Functions

Predefined Functions or Library functions are the built-in functions that are already defined in the C++ library. These functions do not need to be declared and defined. Theprototype of these functions is written in header files. So, we need to include respective header files before using a library function. For example, the prototype of math functions like *pow()*, *sqrt()*, etc is present in math.h, the prototype of *exit()*, *malloc()*, *calloc()* etc is in stdlib.h and so on.

Q5. Define user-defined function.

User-defined Functions

A function created by the user is known as user defined function. Programmer has to write the coding for such functions and test them properly before using them. The syntax of the function is given by the user so there is no need to include any header files. These functions need declaration and definition. When the user-defined function is called from any part of the program, it will execute the code defined inside the body of the function.

For example, multiply (), sum (), average (), etc., are some of the user defined functions.

A user-defined function is based on two parts:

1. Function declaration or prototype

2. Function definition

Q6. Differentiate between predefined and user defined function.

S.NO.	Predefined function	User defined function
1.	A function whose prototype is already defined in library is called predefined function. **Example:** pow (), strcpy (), exit(), etc.	A function which is created by user for its own programming requirement called user defined function. **Example:** multiply (), sum (), average (), etc.
2.	We do not need to write a code for predefine function.	We have to write a code for user defined function.
3.	The name of predefined function cannot be changed because its prototype predefined in the compiler.	The name of user defined function can be changed.
4.	In every program the predefined function must be compulsory.	User defined function is not compulsory.
5.	It is already present inside header file. **Example:** *pow(), sqrt()*, etc is present in math.h, the prototype of *exit(), malloc(), calloc()* etc is in stdlib.h and so on.	For user defined we does not need to write header file. **Example:** multiply (), sum (), average ()→ No header file.
6.	Predefined functions are the part of header file.	User defined functions are the part of program.
7.	Predefined functions is given by the developer.	User defined functions are decided by user.

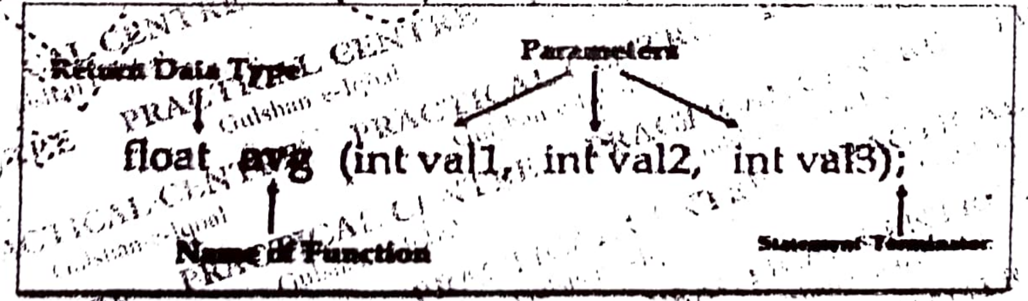
Q7. State the function declaration or function prototype.

Function Declaration:-

A function without its definition (code block) is known as a function declaration or function prototype is declared before the main () function. A function declaration tells the compiler about the number of parameters function takes, data-types of parameters, and return type of function and it ends with statement terminator. Putting parameter names in function declaration is optional in the function declaration, but it is necessary to put them in the definition.

Below is an example of function declaration.

Syntax:



Function Declaration

a. Return Data Type:-

It shows the data type of value returned by function. It may be int, float, double and char data types. If no value is returned by the function in that case keyword "void" is used.

b. Function Name:-

It specifies the name of function. It is recommended that meaningful and understandable names are given to the function.

c. Parameters:-

It is defined the list of data types of function parameters that are to be passed to the function. Parameters are separated by commas. Parameters are also known as arguments. If there is no parameter in function, programmer uses keyword "void". Variable names are optional in prototype parameters /arguments.

d. Statement Terminator:-

In function declaration, statement terminator must be used at the end.

Q8. What does a function definition contain?

Function Definition:-

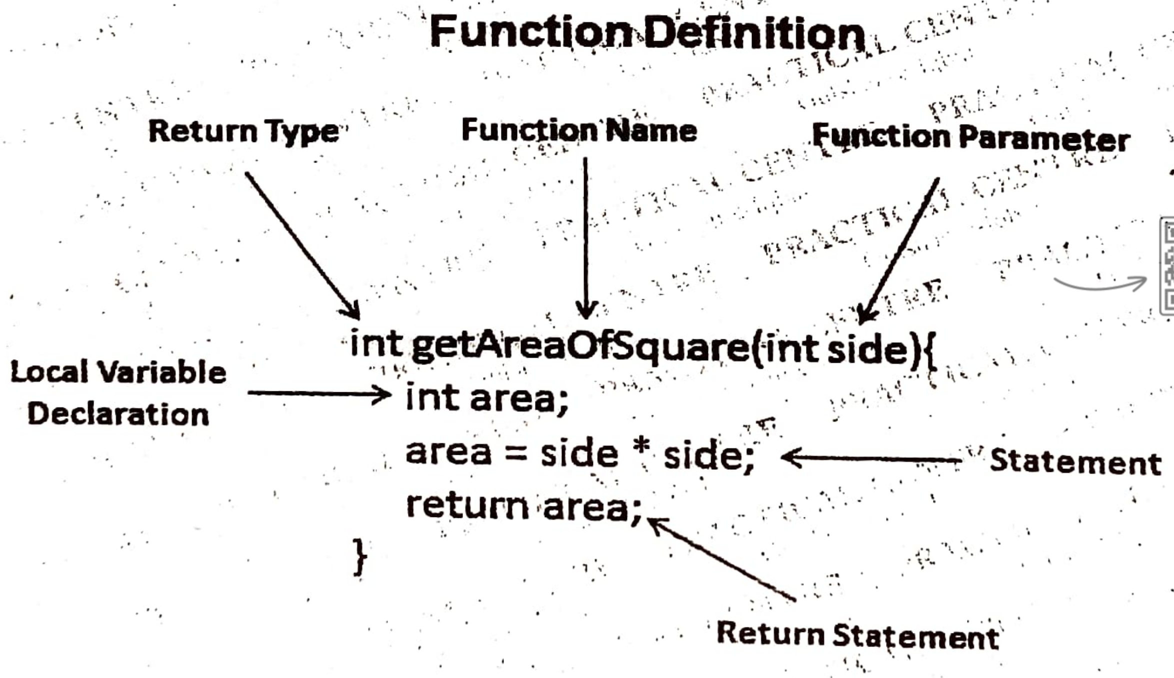
Function definition is function itself. It consists of function name, function parameters, return value and function's body.

- First line is called Function Header and it should be identical to function Declaration/Prototype except semicolon.
- Name of arguments are compulsory here unlike function declaration.
- Function's definition contains the code to be executed when this function is called.

  Syntax of Function Definition

```
return type function name (type arg1, type arg2 .....)
{
   body of a function
   return (expression);
}
```

Example:-



```
int getAreaOfSquare(int side){
   int area;
   area = side * side;
   return area;
}
```

Q9. What is meant by call a function?

Calling a Function

To use the function code, we have to call or invoke that function with its name. It i called function call. When a program calls a function, program control is transferred to the called function. A called function performs defined task and when its return statement is executed or when its function-ending closing brace is reached, it return program control back to the main program.

If the function is without return value and no argument, then its called by its name. I means that function's braces will be empty.

Syntax: function_name ();

Example:add ();

If the function returns a value, then we can store return value to a variable in the calling function.

Example: x=add (y,z);

Q10. Describe Function passing arguments or parameters.

Function passing or parameters:-

An argument is a part of data passed to the function. when function is called fo execution, the actual values as parameters are also given with function call statement Passing actual values to function as argument with function call statement are known as actual parameters. Actual parameters may be variables or constants. They are placed in parentheses after function name. These values are received in variables of the header of function definition. These receiving variables are called the formal parameters of the function. It acts as a local variable inside the function in which they are used.

Q11. What is "Return value"?

Returning value from Function:-

In C++, the return keyword allows a function to return a value. When a function completes its execution, it can return a single value to calling function. Return data type must be specified with the function header in the function definition as well as function declaration. It is written before function name.

Syntax: int function name ();

Q12. What is the difference between Function definition and Function call?

Differentiate between function definition and function call

Function definition and function call can be differentiated on the basis of following criteria.

The function definition is function itself. It has header and body with statements. Function definition may appears before or after the main ()function.	**Function call is to invoke the code of function by its name. As the function is called, control is transferred to the called function.**

Syntax:
`data_type function_name (parameters list)`
`statements;`
`}`
Example
`int sum(int p, int q)`
`{int z;`
`z= p + q;`
`return z;`
`}`
	Syntax:
	`variable_name= function_name (parameter list);`
	
	Example
	`a=sum (x, y);`

Q13. I how many ways can we use user-define function, based on argument /parameters and return type?

Different Ways to Use User-Define Function based on argument/parameters and return type:-

There are four different methods in C++ based on passing parameters to the function and return values from the functions.

- No return vale and No passing arguments/parameters
  void function name(void);
- Return vale but No passing arguments/parameters
  int/float/char function name(void);
- No return vale but passing arguments/parameters
  void function name (int, float, char);
- Return vale and Passing arguments/parameters
  int/float/char function name (int, float, char);

Q14.Write a program to find square root by using predefined function in C++.

Using the Pre - defined Function in C++ Program

```
//Using Square Root formula in program
#include<iostream.h>
#include<cmath.h>
using namespace std;
int main(void)
{

   int a;
   cout << "\n Enter the value of a.......";
   cin >> a;
   cout << "\n The square root of a is......" << sqrt(a); //
Pre-defined function
   return 0;
}
```

Q15. Write a program to find the sum of two numbers by using user defined function in C++.

```
#include <iostream>
usingnamespacestd;

//function declaration
int addition(int a,int b);

intmain()
{
        int num1;        //to store first number
        int num2;        //to store second number
        int add;//to store addition

        //read numbers
        cout<<"Enter first number: ";
        cin>>num1;
        cout<<"Enter second number: ";
        cin>>num2;

        //call function
        add=addition(num1,num2);

        //print addition
        cout<<"Addition is: "<<add<<endl;

        return0;
}

//function definition
int addition(int a,int b)
{
        return(a+b);
}
```

Q16. Write a program to find the average of two numbers by using user defined function in C++.

```
/* Program of an average of two numbers by using ser-define
function */
#include<iostream.h>
using namespace std;

flaot average(int x, int y);
int main(void)
{
    int x,y;
    average(x,y);

return 0;

}

float average(int x, int y)
{
  cout << "\n \t Enter the value of x.........:";
  cin  >> x;

  cout << "\n \t Enter the value of y.........:";
  cin  >> y;

  avg = (x+y) / 2;

  cout << "\n The average of two numbers is....” << avg;
  return avg;
```

Q17. Write a program using Local variables.

```
#include<iostream.h>
using namespace std;
int main(void)
{
int a=10;  // Local Variable
cout << "\n The value of a....." << a;
return 0;
}
```

Q18. Describe Local and Global variables.

Local Variable:-

Variables that are declared within or inside a function block are known as Local variables. These variables can only be accessed within the function in which they are declared. The lifetime of the local variable is within its function only, which means the variable exists till the function executes. Once function execution is completed, local variables are destroyed and no longer exist outside the function. The reason for the limited scope of local variables is that local variables are stored in the stack, which is dynamic in nature and automatically cleans up the data stored within it. But by making the variable static with "static" keyword, we can retain the value of local variable.

Global Variable:-

Global variables are those variables which are declared outside of all the functions or block and can be accessed globally in a program. It can be accessed by any function present in the program. Once we declare a global variable, its value can be varied as used with different functions. The lifetime of the global variable exists till the program executes. These variables are stored in fixed memory locations given by the compiler and do not automatically clean up. Global variables are mostly used in programming and useful for cases where all the functions need to access the same data.

Q19. Write a program using Global Variables.

```
#include<iostream.h>
using namespace std;
void add(int c);
int a=20;  // Global Variable
int main(void)
{
     int x;    //Local Variable
     cin>>x;
     add(x);
     cout << "\n the value of a is....." << a;

 return 0;
 }

     void add (int c)
     {
          int b;   //Local Variable
          b=a+c;
          cout << "\n the addition of two numbers is....."
     << b;
     }
```

MCQs

Q20. Choose the right answer:

1. **The variable that is listed in the functions are called:**
   a) Actual parameter b) Declared parameter
   c) Passes parameter d) none of these
2. **A program can create custom header files that must end with:**
   a) .h extension b) .l extension c) .ios extension d) .a extension
3. **To make large programs more manageable, programmers modularize them into subprograms that are called:**
   a) Operators b) Classes **c) Functions** d) none of these
4. **This one of the following is the default return value of functions in C++?**
   a) int b) char c) float d) void
5. **The execution of the program starts at:**
   a) user-defined function **b) main function** c) void function d) else functio
6. **The mandatory parts in the function declaration:**
   a) return type, function name b) return type, function name, parameters
   c) parameters, function name d) parameters, variables
7. **This one of the following is used to terminate the function declaration:**
   a) : b)) **c) ;** d)].
8. **The scope of the variable declared in the user defined function is:**
   a) whole program **b) only inside the {} block**
   c) the main function d) header section
9. **The minimum numbers of functions that should be present in a C++ program for its execution is:**
   a) 0 **b) 1** c) 2 d) 3
10. **We define the default values for a function when:**
   a) a function is defined **b) a function is declared**
   c) the scope of the function is over d) a function is called
11. **The default parameters should appear in a function prototype:**
   a) To the rightmost side of the parameter list
   b) To the leftmost side of the parameter list
   c) Anywhere inside the parameter list
   d) In the middle of the parameter list
12. **If an argument from the parameter list of a function is defined constant, then:**
   a) It can be modified inside the function
   b) It cannot be modified inside the function
   c) Error occurs
   d) Segmentation fault
13. **Default values for a function are specified when:**
   a) Function is defined **b) Function is declared**
   c) Caller function d) none of these
14. **The return statement returns the execution of the program to:**
   a) main function **b) Caller function** c) same function d) None of these
15. **When our function does not need to return anything, it means we will pass this as parameter in function:**
   a) void **b) blank space** c) both d) None of these

16. If an argument to a function is declared as const, then _____________.

a) function can modify the argument **b) function cannot modify the argument**
c) const argument to a function is not possible d) none of these

17. This one of the following statement is correct:

a) Only one parameter of a function can be a default parameter
b) Minimum one parameter of a function must be a default parameter
c) All the parameters of a function can be default parameters
d) No parameter of a function can be default

18. This/These of the following function or types of function cannot have default parameters:

a) Members function class **b) Main()**
c) Member function of structure d) none of these

19. The () parenthesis in a function call

a) is actually an operator in C++ b) causes the function to be called
c) causes syntax error **d) Both a and b**

20. The are mandatory parts in the function declaration:

a) return type, function name **b) return type, function name, parameters**
c) parameters, function name d) parameters, variables

21. This will happen when we use void in argument passing:

a) It will not return values to its caller b) it will return value to its caller
c) both a & b are correct d) none of these

22. A called function can return control to the caller if:

a) The function does not return a value
b) If that function has no value in expression
c) If the program ends successfully
d) all of these

23. This function is known as standard library function:

a) maths **b) main** c) sqrt d) void

24. A function in C++ that does not return a value is declared with:

a) empty return type **b) void return type** c) endl return type d) getchar return type

25. Use of functions

a) helps to avoid repeating set of statements many times
b) enhance the logical clarity of the program
c) makes the debugging task easier
d) all of these

26. The default parameter passing mechanism is:

a) call by value b) call by reference c) call by value result d) none of these

27. The types of functions in C++ Language are:

a) Library Functions b) User Defined Functions
c) Both Library and User Defined d) none of these

28. The limit for number of functions in a C++ Program is:

a) 16 b) 31 c) 32 **d) No Limit**

29. This function calls itself:

a) Self Function b) Auto Function
c) Recursive Function d) Static Function

30. Choose correct statements about C++ Language Pass by Value.
a) Pass By Value copies the variable value in one more memory location
b) Pass By Value does not use Pointers
c) Pass By Value protects your source or original variables from changes in outside functions or called functions
d) All of these

31. These characters are allowed in a C++ function name identifier:
a) Alphabets, Numbers, %, $, _ **b) Alphabets, Numbers, Underscore (**
c) Alphabets, Numbers, dollar $ d) Alphabets, Numbers, %

32. Arguments received by a function in C++ language are called
a) Definite arguments **b) Formal arguments**
c) Actual arguments d) Ideal arguments

33. Choose the correct statement about C++ language function arguments:
a) Number of arguments should be same when sending and receiving
b) Type of each argument should match exactly
c) Order of each argument should be same
d) all of these

34. The local variables are known as:
a) external variables
b) static variables
c) dynamic variables
d) automatic variables

35. When a program is terminated these variable are destroyed:
a) auto variables **b) global variables** c) register d) local variables

36. The keyword used to transfer control from a function back to the calling function is:
a) switch b) goto **c) return** d) exit

37. All keywords in C++ are in:
a) LowerCase letters b) UpperCase letters
c) CamelCase letters d) none of these

38. Functions in C++ Language are always:
a) Internal
b) External
c) Both Internal and External
d) External and Internal are not valid terms for functions

39. The max number of arguments present in function in the C++ compiler can be:
a) 99 b) 90 c) 102 **d) 127**

40. In C++, a parameter may be passed in this no. of ways:
a) 1 b) 2 c) 3 d) 4

41. If the function returns no value, it is called:
a) Data type functionb) Calling function c) Main function d) Void function

42. A function:
a) May or may not need input data b) May or may not return a value
c) Both a and b d) none of these

43. Of the following input functions this cannot be used to input multiword string in a single function call:
a) getche() b) gets() c) cin() d) None of these

44. This of the following functions can be used to write an entire array or a structure in a single call:
a) writeblock() b) write() **c) fwrite()** d) WRITE()

45. Library function getch() belongs to this header file:
a) stdio.h **b) conio.h** c) stdlib.h d) stdlibio.h

46. Library function pow() belongs to this header file:
a) mathio.h **b) math.h** c) square.h d) stdio.h

47. The other name for external variables in C++ is:
a) Global variables b) Static variables c) Register variables d) none of these

48. A variable is a symbol that represents:
a) A number b) Nothing
c) A storage location in the computer's memory d) A string

49. In ______________ ,a local variable is defined within a block.
a) File scope **b) Local scope** c) Class scope d) Function scope

50. The standard C++ library file<stdlib.h> is used for:
a) Declares a utility function b) Declares a mathematical function
c) Declares time function d) Declares date function

51. The lifetime of a ___ variable is the lifetime of the function block.
a) Function scope b) File scope c) Local scope d) Global scope

52. A function can be invoked from another function using its:
a) Value b) Return **c) Name** d) Variables

53. The calling function parameters are called:
a) Actual parameters b) Dummy parameters
c) Duplicate parameters d) Formal parameters

54. Identify the correct statement regarding scope of variables:
a) Local variables are declared in the main body of the program and accessible only from functions.
b) Local variables are declared inside a function and accessible within the function only.
c) Global variables are declared in a separate file and accessible from any program.
d) Global variables are declared inside a function and accessible from anywhere in the program.

55. The difference between Declaration and Definition is:
a) Declaration does allocate memory for a variable. Definition does allocate memory for a variable.
b) Declaration does allocate memory for a variable. Definition does not allocate memory for a variable.
c) Declaration does not allocate memory for a variable. Definition does allocate memory for a variable.
d) Declaration does not allocate memory for a variable. Definition does not allocate memory for a variable.

56. Data shared among the functions is done with the help of:
a) register variable b) static variables c) local variables **d) global variables**

EXERCISE

A. Encircle or choose the correct answer:

1. The functions that are defined by the programmer are called:
a) Built-in functions **b) User-defined functions**
c) Subfunctions d) Functions

2. A programmer creates a function for a particular task and the programmer wants to include that function in program. This extension is required to save that function:
a) .obj **b) .h** c) .cpp d) .exe

3. In C++, intmain () returns this data type value by default:
a) float **b) int** c) char d) double

4. The parameters specified is the function header are called:
a) formal parameters b) actual parameters
c) default parameter d) command line parameters

5. The word "prototype" means:
a) Declaration b) Calling c) Definition d) Both a & b

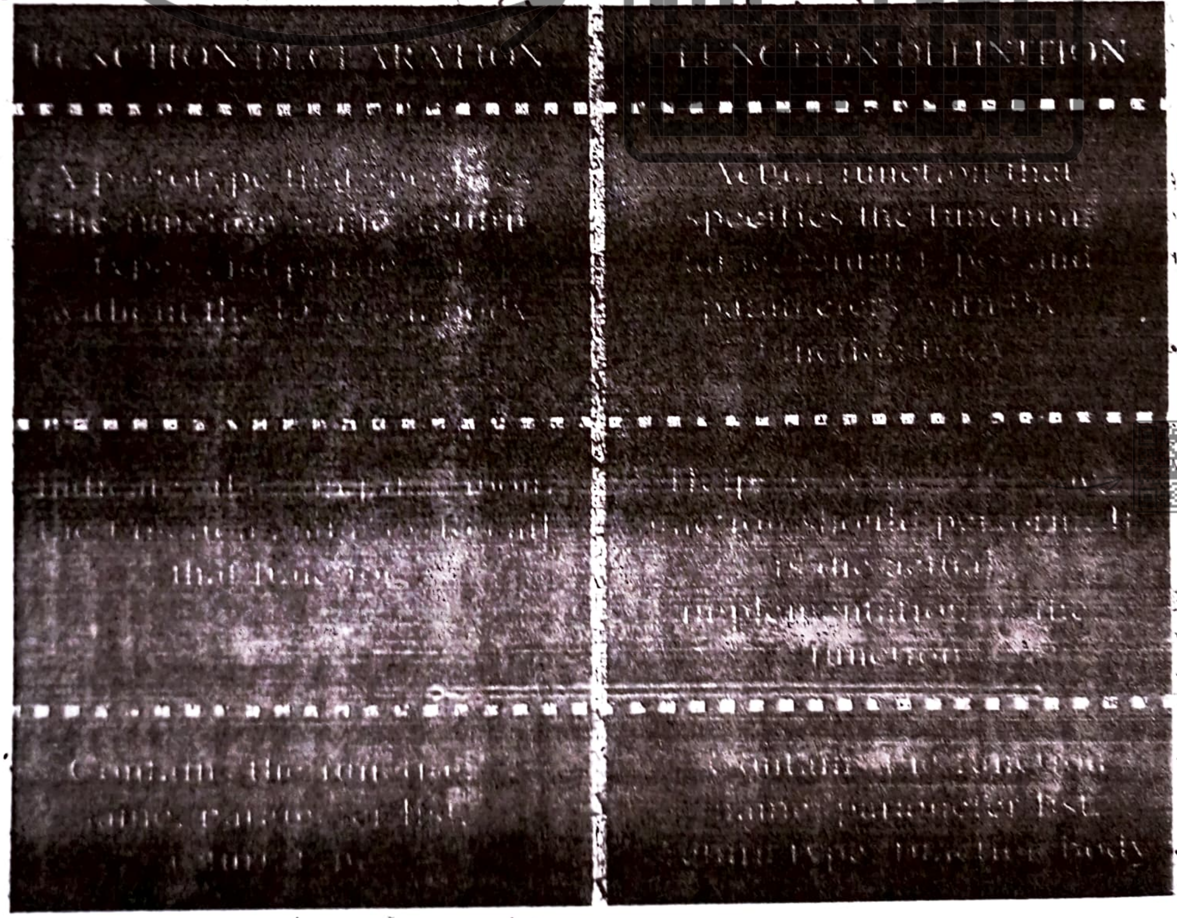
6. The function prototype consists of:
a) Name of function b) The parameters are passes to the function
c) The value return from function **d) all of these**

7. All variables declared in function definition are called:
a) Local variables b) Instance variables c) Global variables d) Stack variables

8. These are not the built-in functions:
a) sqrt() **b) time()** c) exp() d) sin()

2. Respond the following:

Q1. Differentiate between function declaration and function definition.



FUNCTION DECLARATION	FUNCTION DEFINITION
A prototype that specifies the function name, return types and parameters without the function body	Actual function that specifies the function name, return types and parameters with the function body
Indicates the compiler about the function [illegible] that function	[illegible] so the [illegible] it is the actual implementation of the function
Contains the function name, parameter list, return type	Contains the function name, parameter list, return type, function body

Q2. What is the purpose of keyword "void" in function?

The keyword void in a function declaration has a significant purpose. When used as a function return type, the void keyword specifies that the function does not return a value. When used for a function's parameter list, void specifies that the function takes no parameter.

Q3. Why do we use header file?

C++ language has numerous libraries that include predefined functions to make programming easier. Header files offer these features by importing them into your program with the help of a preprocessor directive called **#include.** These preprocessor directives are responsible for instructing the C++ compiler that these files need to be processed before compilation and includes all the necessary data type and function definitions. All the header file have a '**.h**' as extension. By including a header file, we can use its contents in our program.
C++ also offers its users a variety of functions, one of which is included in header files. In C++, all the header files may or may not end with the "**.h**" extension.

A header file contains:

1. Function definitions
2. Data type definitions
3. Macros

Q4. Differentiate between passing argument and return the value from function.

Function passing arguments or parameters:-
An argument is a part of data passed to the function. When function is called for execution, the actual values as parameters are also given with function call statement. Passing actual values to function as argument with function call statement are known as actual parameters. Actual parameters may be variables or constants. They are placed in parentheses after function name. These values are received in variables of the header of function definition. These receiving variables are called the formal parameters of the function. It acts as a local variable inside the function in which they are used.

Returning value from Function:-
In C++, the return keyword allows a function to return a value. When a function completes its execution, it can return a single value to calling function. Return data type must be specified with the function header in the function definition as well as function declaration. It is written before function name.
Syntax: int function name ();

Q5. What is the difference between external variables and functional local variables?

External Variable	Local Variable
External variables are declared outside all the function blocks.	Local variables are declared within a function block.
The scope remains throughout the program..	The scope is limited and remains within the function only in which they are declared.

Any change in external variable affects the whole program, wherever it is being used.	Any change in the local variable does not affect other functions of the program.
Anexternal variable exists in the program for the entire time the program is executed.	A local variable is created when the function is executed, and once the execution is finished, the variable is destroyed.
It can be accessed throughout the program by all the functions present in the program.	It can only be accessed by the function statements in which it is declared and not by the other functions.
If the external variable is not initialized, it takes zero by default.	If the local variable is not initialized, it takes the garbage value by default.
External variables are stored in the data segment of memory.	Local variables are stored in a stack in memory.
We cannot declare many variables with the same name.	We can declare various variables with the same name but in other functions.

Q6. List the five standard built-in functions with examples.

Five Standard Built in Functions.

The five-standard built in functions are as follows:

- **max (p,q):** It will return a maximum number between p and q.
- **min (p,q):** It will return a minimum number between p and q.
- **pow (m,n):** It will calculate m raised to the power n.
- **sqrt(m):** It will calculate the square root of m.
- **cbrt(n):** It will calculate the cube root of n.

Example:-

```
#include <iostream>
#include <math.h>
using namespace std;
int main()
{
cout << max(16,18) << "\n";
cout << min(16,18) << "\n";
cout << pow(2,3) << "\n";
cout << sqrt(16) << "\n";
cout << cbrt(27) << "\n";
return 0;
}
```

Ouput:-

```
18
16
8
4
3
```

Q7. Write the advantages of User-define functions in C++.

Advantages of User Defined Functions:

When not using user defined functions, for a large program the tasks of debugging, compiling etc. may become difficult in general. That is why user defined functions are extremely necessary for complex programs. The necessities or advantages are as follows:

1. It facilitates top-down modular programming. In this programming style, the high-level logic of the overall problem is solved first while the details of each lower-level function are addressed later.
2. The length of a source program can be reduced by using functions at appropriate places.
3. It is easy to locate and isolate a faulty function for further investigations.
4. A function may be used by many other programs. This means that a programmer can build on what others have already done, instead of starting all over again from scratch.

Q8. Get the output highlight the errors from the following program.

```
#include<iostream>
void Table(void);
void Table(void)
{
int m,n;
cout << "\n The value of m...";
cin >> m;
for (n=1;n<10;n++)
{
cout << "\t" << m << "*" << n << "=" << m*n << "\n";
}
void main(void)
{
      table ();
}
```

ERRORS:-

In function 'void Table ()':
[Error] 'cout' was not declared in this scope
[Note] Suggested alternative:
[Note] 'std::cout'

[Error] 'cin' was not declared in this scope
[Note] Suggested alternative:
[Note] 'std::cin'

At global scope:
[Error] 'main()' must return 'int'

In function 'int main()':
[Error] 'table' was not declared in this scope

Correct Program:-

```
#include<iostream>
void Table(void);
void Table(void)
{
int m,n;
std::cout << "\n The value of m :";
std::cin >> m;
for (n=1;n<=10;n++)
 {
 std::cout << "\t" << m << "*" << n << "=" << m*n << "\n";
 }
}

 int main(void)
 {
        Table ();
 return 0;
 }
```

Output:-

```
The value of m : 7
        7*1=7
        7*2=14
        7*3=21
        7*4=28
        7*5=35
        7*6=42
        7*7=49
        7*8=56
        7*9=63
        7*10=70
```

BOOLEAN ALGEBRA 6

CONTENTS

6.1 INTRODUCTION:

The English mathematicians felt that there was a connection between mathematics and logic, but no one before George Boole could find this missing link. In 1850s, he invented symbolic logic, known today as Boolean Algebra. Each variable in Boolean Algebra has either of the two values, True or False. Is an action, right or wrong? A motive, good or bad? A conclusion, true or false? Much of our thinking involves trying to find the answer to two-valued questions. The original purpose of these two states in algebra was to solve logic problems. In 1938, Shannon applied the new algebra to telephone-switching circuits. Because of Shannon's work, engineers soon realized that Boolean algebra could be used to analyze and design computer circuits.

Logic Hardware:

The processing of data is done by lots of complex electrical circuits in the CPU of the computer. The hardware devices, which do this, are transistors or integrated circuits.

1. Transistors:

These can function as amplifiers or logic elements. All types of transistors are made from semiconductor material, usually Silicon, which is carefully "doped" with impurities e.g. Phosphorus and Boron, so that the electrical current can be controlled.

2. Integrated Circuits:

These are also known as chips and small circuits, consisting of a number of transistors. Large Scale Integrated (LSI) circuits and Very Large Scale Integrated (VLSI) circuits represent thousands of transistors on a chip. Advancements in chip-making technology have allowed this miniaturization.

Logic:

- Sense of arguments.
- Logical arguments can have true or false answer only.
- May be represented in binary circuits.
- Can be evaluated through digital circuits.

Logic Gates:

- Two-states circuits (low or high).
- More inputs but have only one output.
- Often called logic circuits.
- Gates are constructed through.
  - Diodes
  - Transistors
- Input and output relationship can be represented through truth tables.

What is Gate?

The most common use of logic elements is to act as switches, although they have no moving parts. They open to pass on a pulse of electricity or close to shut it off. This is why they are known as gates.

There are three basic types of gates, i.e. , AND, OR, NOT.

6.2 BOOLEAN ALGEBRA TERMS:

1. Boolean Constants.
2. Boolean Variables.
3. Complement.
4. Truth Table.
5. Logical Operators. (a) OR (b) AND (c) NOT.
6. Operator Precedence order.
7. Boolean Expressions.
8. Boolean Functions.

1. Boolean Constants:

In Boolean algebra, a set of constants has only two elements 0 or 1. They are stored in Boolean variables. Thus, a Boolean constant is either 0 if not 1 or is 1 if not 0.

2. Boolean Variables:

The variables used in Boolean Algebra can be represented by the letters or the alphabets such as A, B, C,......X, Y, Z etc. Each variable must take one of the two constants 1 or 0 at a time. These two values may be given different names such as TRUE or FALSE, YES or NO, HIGH or LOW, UP or DOWN, ON or OFF etc.

3. Complement:

The complement is the inverse of a variable and is indicated by a bar over the variable. For example, complement of the variable A is written as $\overline{A}$.

EXAMPLES:

1. If $A = 1$ Then, $\overline{A} = 0$
2. If $A = 0$ Then, $\overline{A} = 1$

4. Truth Table:

A truth table is a table that shows the result of a Boolean expression for all the possible combinations of the values given to the variable used.

5. Logical Operators:

In Boolean algebra, there are three basic operators, one unary operator NOT and two binary operators AND and OR as shown in the following table.

Operation	Symbol used	Comments
NOT	Prime (') or Bar (¯)	Negation of the value (unary operation)
OR	Plus (+) or Union (∪)	Logical addition
AND	Dot (.) or Intersection (∩)	Logical multiplication

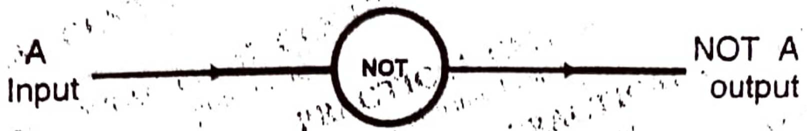
(a) NOT Operator:

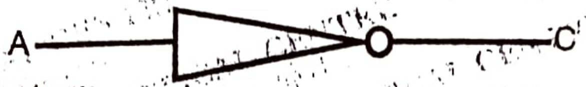
NOT operation is unary operation. It is the negation of a quantity on which it operates. For example $\overline{A}$ (read As A Bar) means complement of A or NOT A.

Truth Table for Logical NOT (¯) operator.

INPUT	OUTPUT
A	$\overline{A}$
0	1
1	0

Not Gate:





(b) AND Operator:

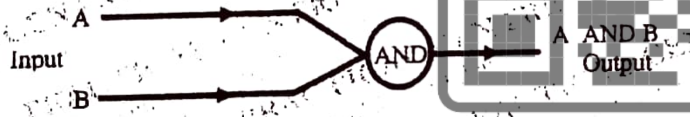
In Boolean algebra, the operation AND means logical multiplication and it is represented with or without a dot between the variables. For example A AND B gives the resulting value of C in the equation.
A.B = C or AB = C read as "A AND B is equal to C".

Truth Table for logical AND (.) operator.

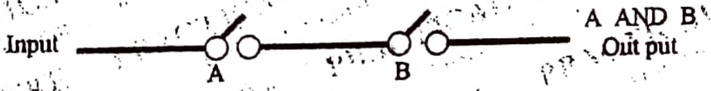
INPUT		OUTPUT
A	B	A. B = C
0	0	0
0	1	0
1	0	0
1	1	1

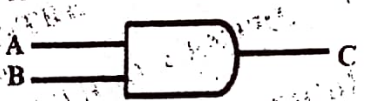
Express as AB, A.B OR A∩B

Two input AND gate:



Another way of showing this is by means of a switching circuit.





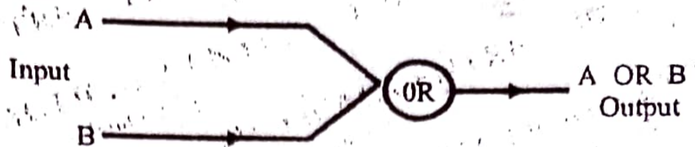
(c) OR Operator:

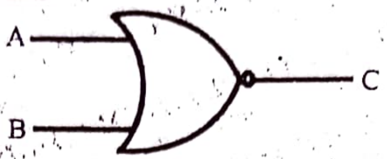
In Boolean Algebra, OR operation means logical addition and it is represented by Plus sign between two variables. For example, observe that
C = 1 if A = 1 or B = 1 or if both are equal to 1. If both are equal to 0, then C = 0

Truth Table for logical OR (+) operator.

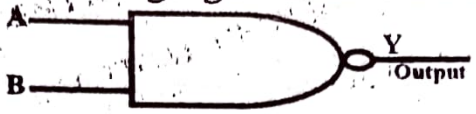
INPUT		OUTPUT
A	B	A+B=C
0	0	0
0	1	1
1	0	1
1	1	1

Two input OR gate:





(d)NAND GATE: This basic logic gate is the combination of AND and NOT gate.

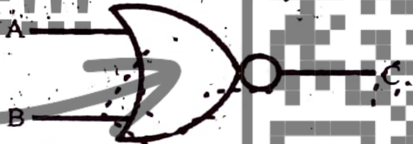


The Boolean expression of NAND gate is Y = /bar {A.B} A.B.
The truth table of a NAND gate is given as;

TRUTH TABLE:

Input	Input	Output
A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0

(e) NOR GATE: This gate is the combination of OR and NOT gate.



The Boolean expressions of NOR gate is Y = \bar {A+B} A+B
The truth table of a NOR gate is as follows;

TRUTH TABLE:

Input	Input	Output
A	B	Y
0	0	1
0	1	0
1	0	0
1	1	0

DIFFERENTIATE BETWEEN BASIC LOGIC GATE AND UNIVERSAL LOGIC GATE

BASIC LOGIC GATE	UNIVERSAL LOGIC GATES
AND, OR, and NOT gate are the most basic logic gates. By using the set of logic gates, it is possible to implement all the possible Boolean Expressions by using these three gates.	The NAND gate and NOR gate can be called the universal gates, the collection of NAND & NOR gate can be used to achieve any of the basic AND, OR and NOT operations.
Individual logic gates can be connected to form a variety of different combinational logic circuits.	A universal gate is a gate which can implement any Boolean expression without using other type of gate.

6.3 RULES OF BOOLEAN ALGEBRA:

Rules of Boolean algebra are useful in manipulating and simplifying expressions.

Rule 1: $A + 0 = A$

A variable ORed with O is always equal to the variable.

Proof:

For A = O	For A = 1
L.H.S. = O + O	L.H.S. = 1 + O
= O	= 1
= A (R.H.S.).	= A (R.H.S.).

Rule 2: $A + 1 = 1$

A variable ORed with 1 is always equal to 1.

Proof:

For A = O	For A = 1
L.H.S. = O + 1	L.H.S. = 1 + 1
= 1	= 1
= (R.H.S.).	= (R.H.S.).

Rule 3: $A . O = O$

A variable ANDed with O is always equal to O

Proof:

For A = O	For A = 1
L.H.S. = O . O	L.H.S. = 1 . O
= O	= O
= (R.H.S).	= (R.H.S.).

Rule 4: $A . 1 = A$

A variable ANDed with 1 is always equal to the variable.

Proof:

For A = O	For A = 1
L.H.S. = O . 1	L.H.S. = 1 . 1
= O	= 1
= A (R.H.S.).	= A (R.H.S.).

Rule 5: $A + A = A$

A variable ORed with itself is always equal to the variable.

Proof:

For A = O	For A = 1
L.H.S. = O + O	L.H.S. = 1 + 1
= O	= 1
= A (R.H.S.).	= A (R.H.S.).

Rule 6: $A . A = A$

A variable ANDed with itself is always equal to the variable.

Proof:

For A = O	For A = 1
L.H.S. = O . O	L.H.S. = 1 . 1
= O	= 1
= A (R.H.S.).	= A (R.H.S.).

Rule 7: $A + \bar{A} = 1$

A variable ORed with its complement is always equal to 1.

Proof:

For A = 0		For A = 1	
L.H.S.	$= 0 + 1$	L.H.S.	$= 1 . 0$
	$= 1$		$= 1$
	$=$ R.H.S.		$=$ R.H.S.

Rule 8: $A . \bar{A} = 0$

A variable ANDed with its complement is always equal to 0.

Proof:

For A = 0		For A = 1	
L.H.S.	$= 0 . 1$	L.H.S.	$= 1 . 0$
	$= 0$		$= 0$
	$=$ R.H.S.		$=$ R.H.S.

Rule 9: $A = \bar{\bar{A}}$

The double complement of a variable is always equal to the variable.

Proof:

For A = 0		For A = 1	
L.H.S.	$\bar{A} = 1, \bar{\bar{A}} = 0$	L.H.S.	$\bar{A} = 0, \bar{\bar{A}} = 1$
	$= A$		$= A$
	$=$ R.H.S.		$=$ (R.H.S.)

A	$\bar{A}$	$\bar{\bar{A}}$
0	1	0
1	0	1

Equal

Rule 10: $A + A . B = A$

Proof:

This rule can be proved by applying the distributive law, rule 2 and rule 4 as follows.

$A + A . B = A(1 + B)$	Factoring A (Distributive law)
$= A . 1$	Rule 2: $B + 1 = 1$
$= A$	Rule 4: $A . 1 = A$

The proof is shown in the Truth Table.

A	B	A . B	A + A . B
0	0	0	0
0	1	0	0
1	0	0	1
1	1	1	1

Equal

Rule 11: $A + \bar{A}.B = A + B$

Proof:

This rule can be proved as:

$= A + AB + \bar{A}B$
$= A + B(A + \bar{A})$
$= A + B.1$
$= A + B$

The proof is shown in the Truth Table.

A	B	$\bar{A}$	$\bar{A}B$	$A + \bar{A}B$	$A + B$
0	0	1	0	0	0
0	1	1	1	1	1
1	0	0	0	1	1
1	1	0	0	1	1

Equal

Rule 12: $(A + B)(A + C) = A + BC$

Proof:

This rule can be proved as follows:

$(A + B)(A + C)$	$= AA + AC + AB + BC$	Distributive law
	$= A \; + AC + AB + BC$	Rule 6 : AA = A
	$= A(1 + C) + AB + BC$	Factoring (Distributive law)
	$= A . 1 + AB + BC$	Rule 2 : 1 + C = 1
	$= A(1 + B) + BC$	Factoring (Distributive law)
	$= A . 1 + BC$	Rule 2 : 1 + B = 1
	$= A + BC$	Rule 4 : A . 1 = A

The proof is shown in the Truth Table.

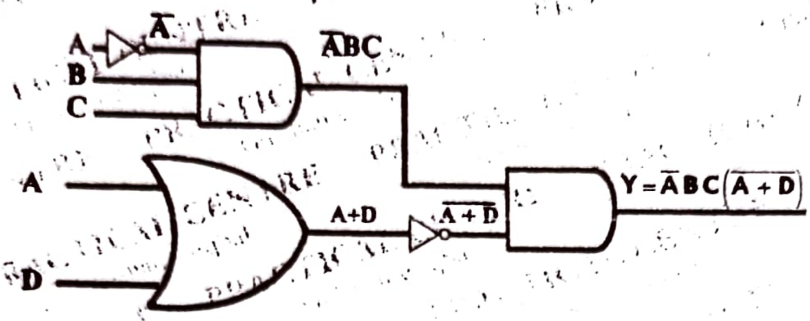
A	B	C	A + B	A + C	(A + B)(A + C)	BC	A + BC
0	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0
0	1	0	1	0	0	0	0
0	1	1	1	1	1	1	1
1	0	0	1	1	1	0	1
1	0	1	1	1	1	0	1
1	1	0	1	1	1	0	1
1	1	1	1	1	1	1	1

Equal

Draw Logic circuit of the given Boolean expressions.

a. $Y = ABC\,(\overline{A + D})$

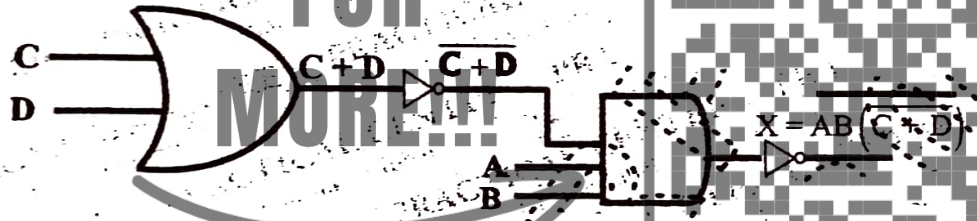
Solution:



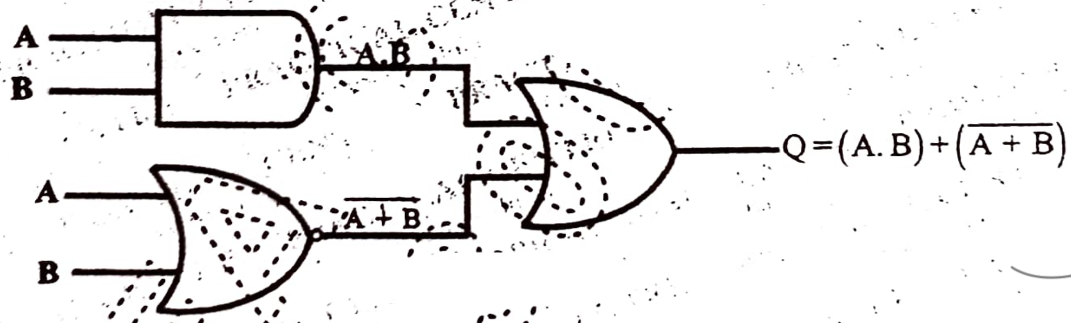
Explanation:

Above logic circuit consist of AND, OR and, NOR gates. The expression A NOT and B, C gate connected with AND gate and A, D is connected with OR gate and converted in NOR gate. Finally, $\bar{A}BC$ and $\overline{A+D}$ connected to AND gate.

b. $X = \overline{AB(C + D)}$

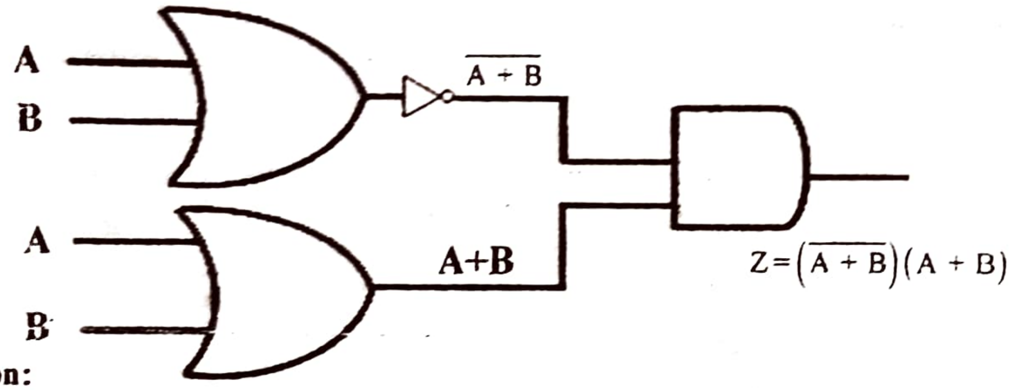


c. Let:- $Q = (A.B) + (\overline{A + B})$



Logic Circuit

6.3.3.3 Derive the Boolean expression from the given circuit and make a truth table of that Boolean expression.



Solution:

Boolean expression of the above circuit is $=Z=(\overline{A+B})(A+B)$

Truth Table of $Z=(\overline{A+B})(A+B)$

INPUTS		TRUTH TABLE		
A	B	A+B	$\overline{A+B}$	$(\overline{A+B}).(A+B)$
0	0	0	1	0
1	0	1	0	0
0	1	1	0	0
1	1	1	0	0

Explain the Function of inverter

A NOT gate, often called an inverter, is a nice digital logic gate to start with because it has only a single input with simple behaviour. **A NOT gate performs logical negation on** its input.

Is an inverter a gate?

In digital logic, an inverter or NOT gate is a **a logic gate which implements logical negation.** In mathematical logic, it is equivalent to the logical negation operator.

Explain the purpose of truth table

The truth table **displays the logical operations on input signals in a table format.** Every Boolean expression can be viewed as a truth table. The truth table identifies all possible input combinations and the output for each.

Q.3 Describe the properties of truth table.

Ans: **Properties of Truth Table:**

It has the following properties:

- Truth table consists of rows and columns.
- It shows the relationship between inputs to an output from a digital logic circuit.
- It shows output for all the possible combinations of input 0 for Low and 1 for High.
- All the combinations of input are listed in columns on the left and output is shown in the extreme right column.
- The input columns are constructed in the order of binary counting with a number of bits equal to the number of inputs.
- Each row of truth table contains one possible combination of inputs and the result of the operation for those values is shown in the extreme right column.

6.3.2.1 Example to apply Boolean Rules for simplification expression.

a. $AB+A\overline{B} = A$

$AB+ A\overline{B}=A$

Solution:

Here take common variable A

$A(B+\overline{B})$

According to 6th Rule of Boolean Algebra, $A+\overline{A} = 1$ so, $B + \overline{B} = 1$

A.(1) Or A.1 → According to 4th Rule of Boolean Algebra, i.e. A.1 = A

A.1 = A

A=A

Hence L.H.S.=R.H.S.

A=A Proved

b. (A + B) + (A + B) = A

Solution:

$(A + B) + (A + \overline{B}) = A$

Take L.H.S.

$(A + B) + (A + \overline{B})$

ANDing (Multiplication) of both expressions

$AA+A\overline{B} + BA+B\overline{B}$

Apply Rule 7th of Boolean Algebra i.e. A.A = A and Rule 8th of Boolean Algebra i.e. $A.\overline{A} = 0$ or $B\overline{B}$

$A+A\overline{B} + BA + 0$

Take common variable A from $A + A\overline{B} + BA$ expression

$A+A(B+\overline{B}) + 0$

Apply Rule 6th of Boolean Algebra i.e. $A+\overline{A} = 1$ or $B + \overline{B} = 1$

A + A (1) + 0 = A + A + 0

Apply Rule 5th of Boolean Algebra i.e. A + A = A

A + 0 = A

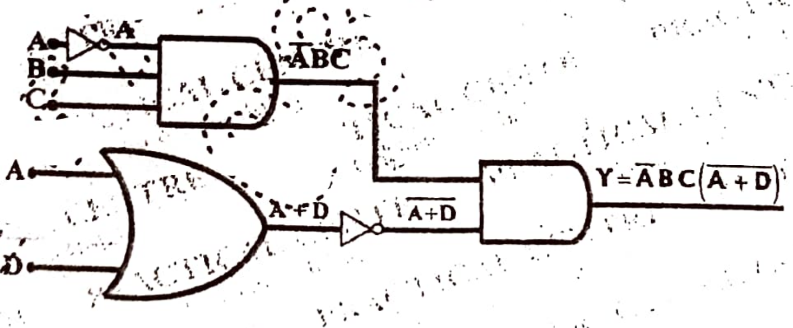
Hence, L.H.S. = R.H.S.

A = A Proved

6.3.2.2 Draw Logic Circuit of the given Boolean expressions.

a. $Y = \overline{A}BC(\overline{A+D})$

Solution:

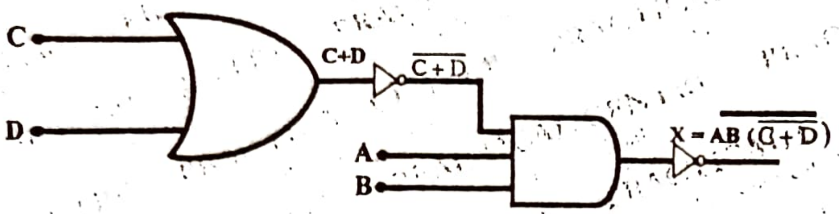


Explanation:

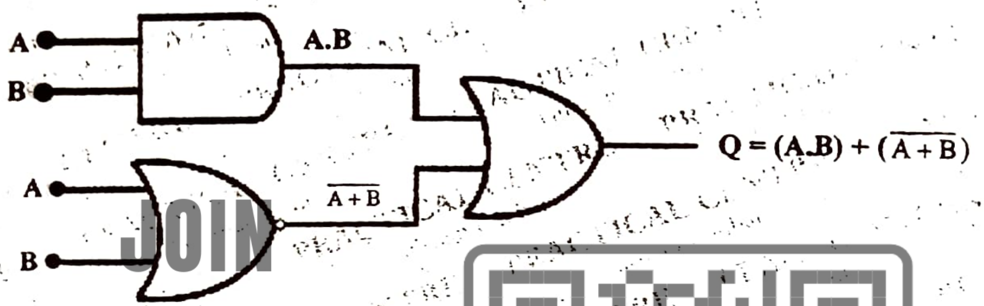
Above logic circuit consists of AND, OR and, NOR gates. The expression $\overline{A}$ NOT and B, C gate connected with AND gate and A, D is connected with OR gate and converted in NOR gate. Finally, $\overline{A}$ BC and $\overline{A+D}$ connected to AND gate.

b. X = $\overline{AB(\overline{C+D})}$

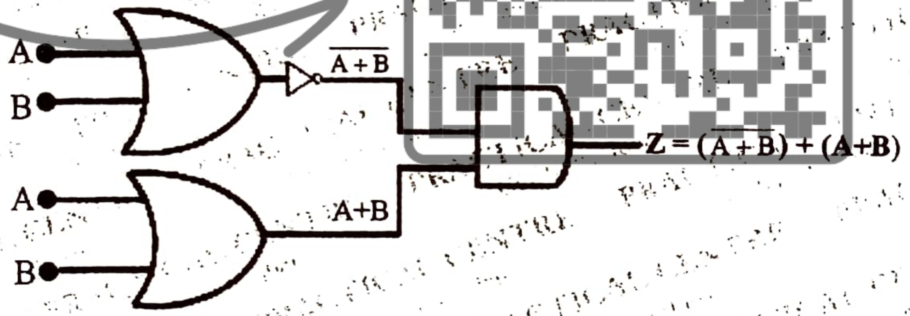
Solution:



c. Let:- Q = (A.B) + ($\overline{A+B}$)



6.3.3.3 Derive the Boolean expression from the given circuit and make a truth table of that Boolean expression.



Solution:

Boolean Expression of the above circuit is Z = ($\overline{A+B}$) (A+B)

Truth table of Z = ($\overline{A+B}$) (A+B)

INPUTS		TRUTH TABLE		
A	B	A+B	$\overline{A+B}$	($\overline{A+B}$).(A+B)
0	0	0	1	0
1	0	1	0	0
0	1	1	0	0
1	1	1	0	0

Explain the function in inverter

A NOTE gate, often called an inverter, is a nice digital logic gate starts with because it has only a single input with simple behaviour. A NOT gate performs logical negation on its input.

Is an inverter a gate?

In digital logic, an inverter or NOT gate is a logic gate which implements logical negation. In mathematical logic, it is equivalent to the logical negation operator.

Explain the purpose of truth table.

The truth table displays the logical operation on inputs signal in a table format. Every Boolean expression can be viewed as a truth table. The truth table identifies all possible input combinations and the output for each.

Q.3 Describe the properties of truth table.

Ans. Properties of Truth Table:

It has the following properties.

- Truth table consists of rows and columns.
- It shows the relationship between inputs to an output from a digital logic circuit.
- It shows output for all the possible combinations of input 0 for LOW and 1 for HIGH.
- All the combinations of inputs are listed in columns on the left and output is shown in the right-most columns.
- The input columns are constructed in the order of binary counting with a number of bits equal to the number of inputs.
- Each row of truth table contains one possible combination of inputs and the result of the operation for those values is shown in the right-most columns.

6.4 MULTIPLE CHOICE QUESTIONS (MCQs)

(1) The founder of Boolean algebra was __________ (2014)
(a) Charles Babbage (b) Blaise Pascal (c) **George Boole** (d) none of these

(2) Boolean algebra used Binary values __________ as Boolean constants.
(a) **0 and 1** (b) 0 to 9 (c) 0 to 7 (d) 0 to 15

(3) OR, AND and NOT are the __________ operations.
(a) Relational (b) **Logical** (c) Arithmetic (d) none of these

(7) The complement is represented by a __________ on the variable.
(a) + sign (b) **Bar sign** (c) – sign (d) Dot sign

(4) This one of the following operators is unary operation:
(a) OR (b) **NOT** (c) AND (d) None of these

(4) Inverse of the variable is called __________
(a) Constant (b) **Complement** (c) Variable (d) none of these

(6) Boolean Algebra provides the operation and the rules for working with the set:
(a) A and B (b) 0 to 9 (c) 0 to 7 (d) **0 and 1**

(7) $A + \overline{A} = 1$ is read as:
(a) **A OR $\overline{A}$ is equal to 1** (b) A AND $\overline{A}$ always equal to 1
(c) A NOT $\overline{A}$ is equal to 1 (d) none of these

(8) Boolean Algebra is being used extensively in designing the _______ used in computers.
(a) **Circuits** (b) Transistors (c) IC's (d) none of these

(9) In _______, an English mathematician, George Boole developed a mathematical system.
(a) **1850's** (b) 1860's (c) 1870's (d) 1880's

(10) Truth table is a table of all possible combinations of the:
(a) Complements (b) Constants (c) **Variables** (d) Logical operators

(11) There are __________ logical operators used in Boolean Algebra
(a) 2 (b) **3** (c) 4 (d) 5

(12) OR operator is represented by:
(a) "+" (b) "•" (c) " " (d) none of these

INTRODUCTION TO SCRATCH 7

Q1. What is Scratch?

SCRATCH:

Scratch is a visual programming language. It is very easy to create programs in scratch. It allows students to create their own interactive stories, games and animations, etc. This programming tool was developed by Massachusetts Institute of Technology (MIT) Media Lab. It is free to use and distribute and it will remain free forever.

Q2. Write down the advantages of Scratch?

ADVANTAGES OF SCRATCH:

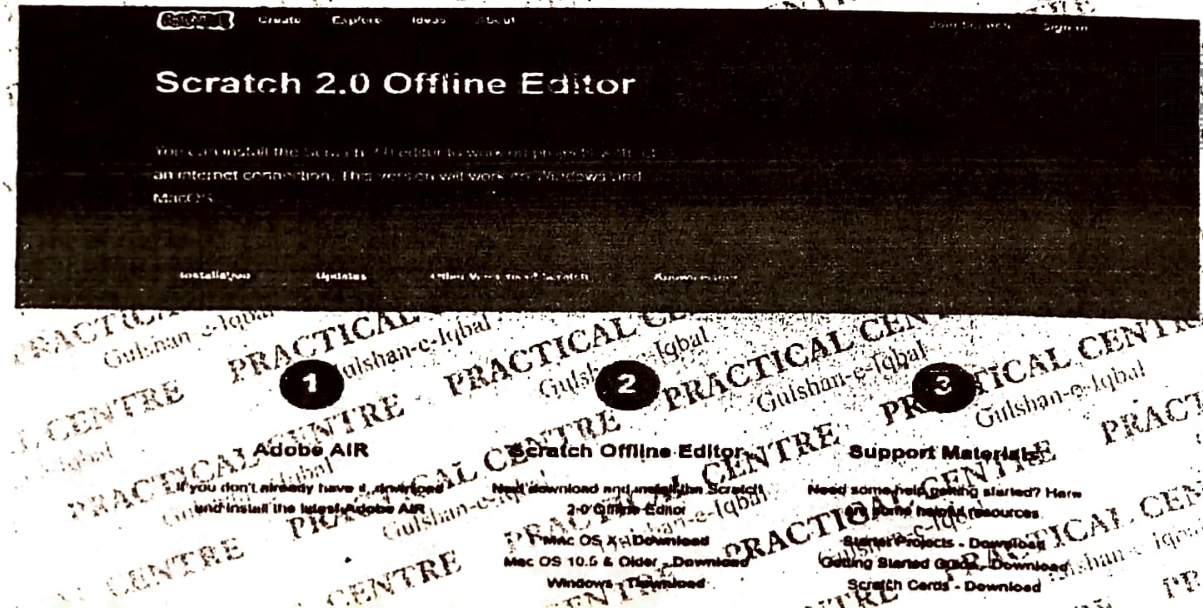
Scratch can be used for multiple purposes:

1. Interactive stories, games, animations, music, art and presentations can be created simply by dragging and dropping color blocks.
2. While working with scratch students learn important mathematical and computing concepts that improve their creative thinking and logical reasoning problem solving.
3. As students design Scratch projects, they learn to think creatively, reason systematically, and work collaboratively.
4. Kids can create animation easily by using scratch.
5. Teens can develop games.
6. Teachers and students can use it to create effective education tools such as math quiz, science simulation and educational videos.
7. It is so easy to use that anyone can master it within short period of time.

Q.3: Write down the procedure and steps to download and install scratch offline.

Downloading Scratch :

1. Scratch can be downloaded or installed on computer or any android device to work offline.
2. The latest version can be downloaded for Windows 10 or Android 6 or the later versions.
3. Scratch 2.0 is a good version that can work on most of the computers and can be downloaded from website: **http:// scratch.mit.edu/download/scratch 2**



Scratch 2.0 Offline Editor

You can install the Scratch 2.0 editor to work on projects without an internet connection. This version will work on Windows and MacOS.

1 Adobe AIR

If you don't already have it, download and install the latest Adobe AIR

2 Scratch Offline Editor

Next download and install the Scratch 2.0 Offline Editor

Mac OS X - Download
Mac OS 10.5 & Older - Download
Windows - Download

3 Support Materials

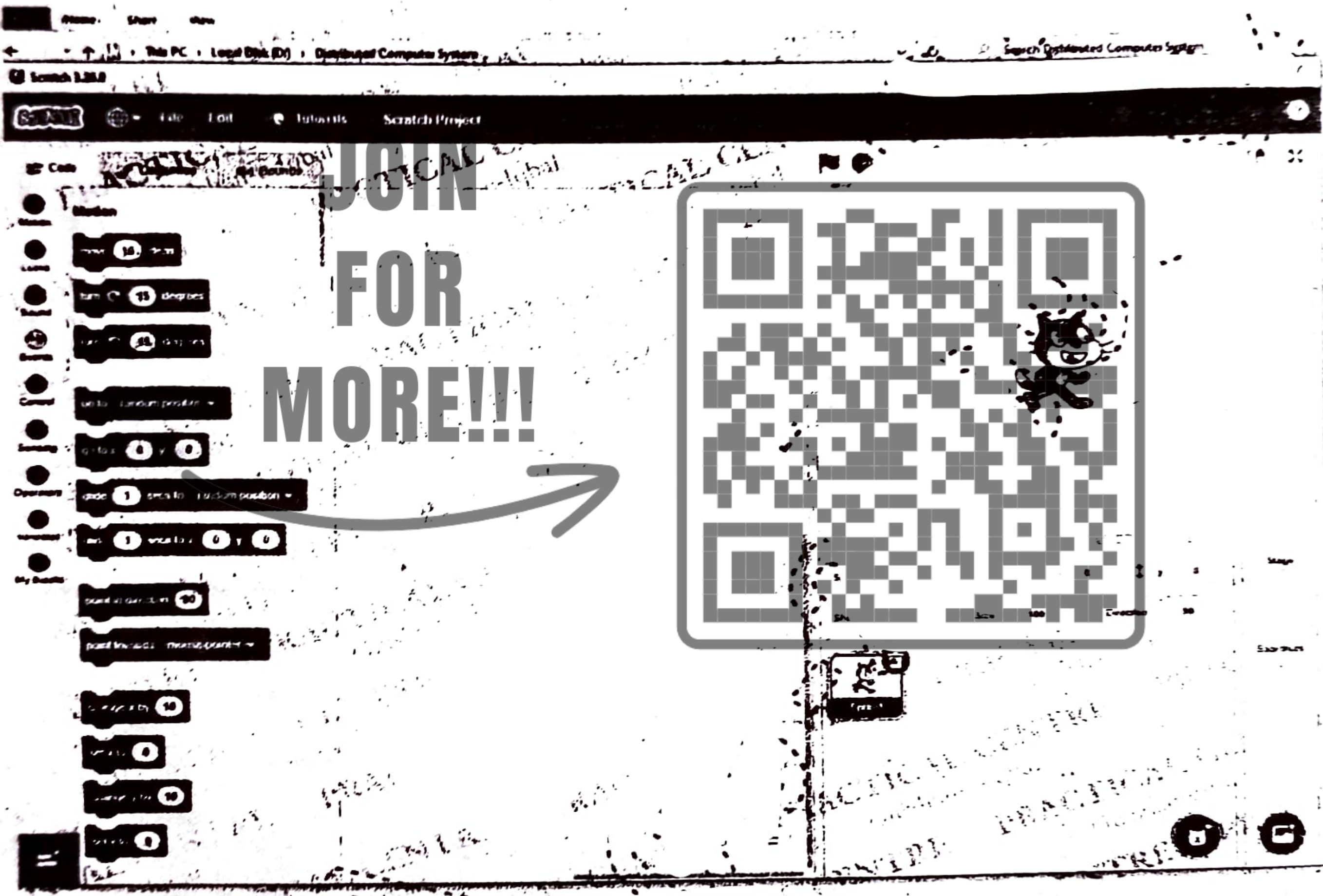
Need some help getting started? Here are some helpful resources.

Starter Projects - Download
Getting Started Guide - Download
Scratch Cards - Download

OFFLINE INSTALLATION:

nstallation of scratch is very easy. The instructions for installation of Scratch 2.0 are given below:

1. To begin scratch Offline Editor for Windows needs to be installed on computer.
2. After down loading we need to just run the executable (.exe) file.
3. The first screen will appear which will ask the location where we want to install the Scratch.
4. It will also ask about the shortcuts to create. We need to just continue with default options.
5. This will initiate the installation process and after installation, Scratch will be ready to use.



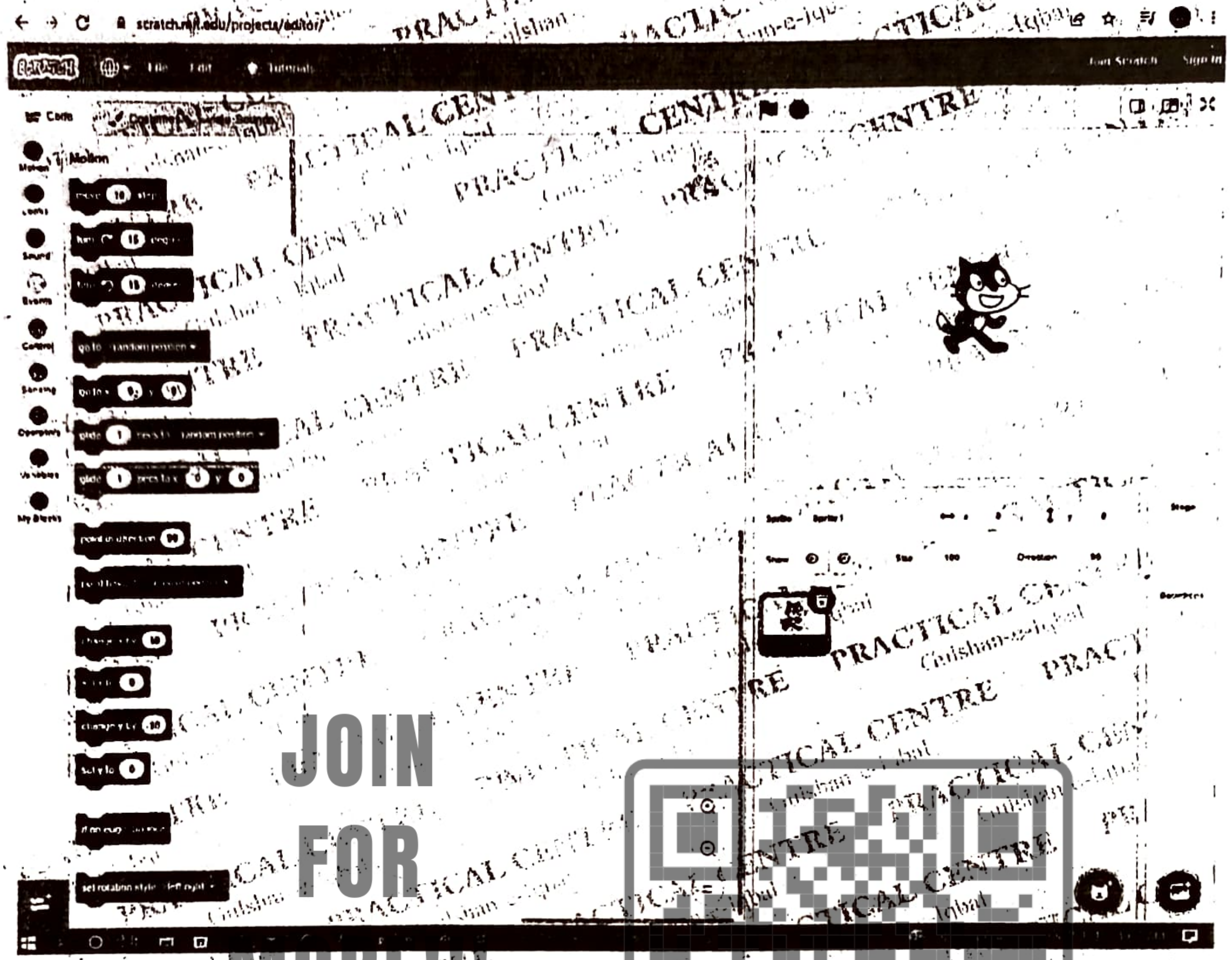
Q4. Can we use the Scratch editor without having it installed on the computer?

SCRATCH ONLINE:

1. Scratch can also be used online by simply loading its editor in our web- browser which is available at:

   https:// scratch.mit.edu/projects /editor/

2. After loading the editor, it will function just like the offline editor. We can also create our account on Scratch. This will enable us to save our projects online.



Q5. What are two basic concepts of the Scratch Environment?

Two basic concepts of Scratch Environment are Sprites and Scripts.

i.SPRITE:

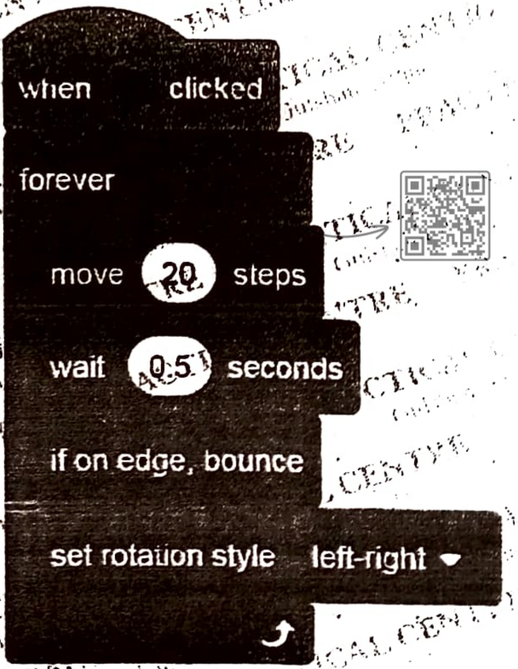
The sprites are the images of cartoons, characters or objects that we add in our project. We can have multiple sprites in our project but at least one sprite is always needed for the project. Cat is the default sprite in scratch.

ii.SCRIPT:

To create a game, interactive story, animation or art work in scratch, We must add visual instructions to tell a sprite exactly what to do. The scripts are instructions that make sprites perform a task. Each sprite in a scratch project has an area for scripts through which it is programmed. Clicking on a script's thumbnail in the sprite pane will bring up the script area of that sprite.

OR

The scripts are the visual instructions to tell a sprite what to exactly do. In short, Scripts are the instructions in a visual form that make a sprite perform a specific task.

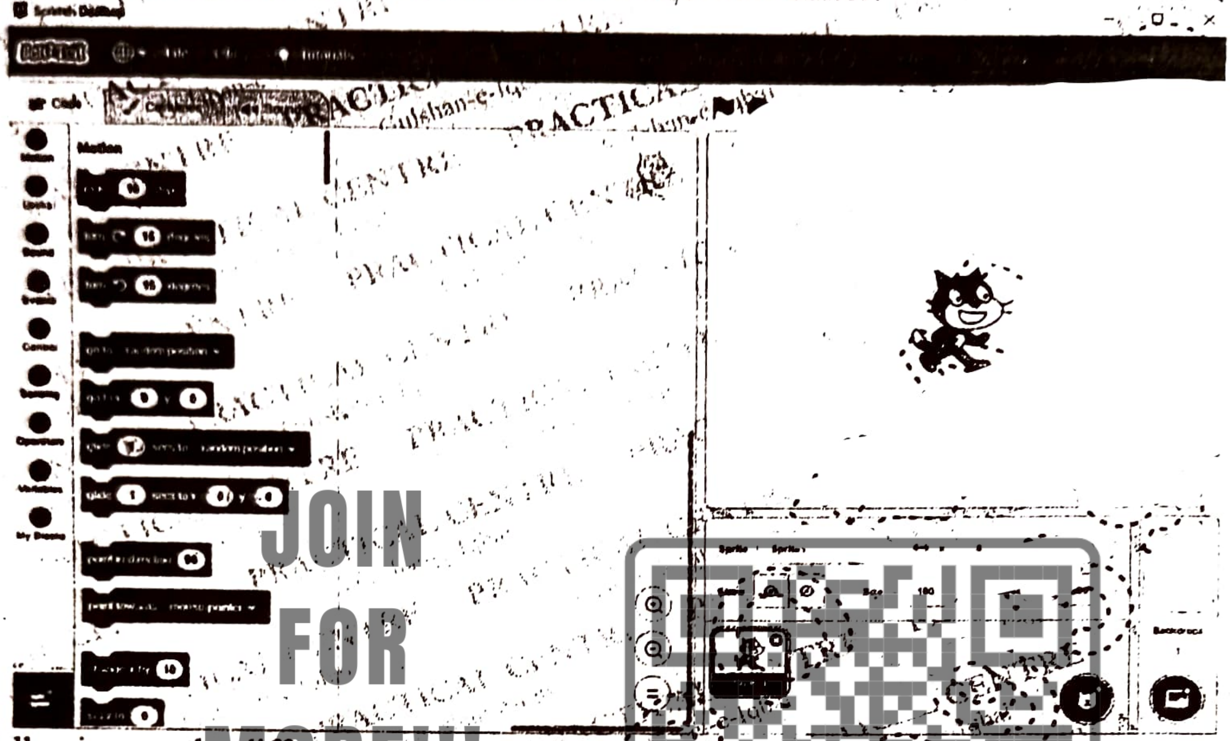


```
when [flag] clicked
forever
    move (20) steps
    wait (0.5) seconds
    if on edge, bounce
    set rotation style [left-right ▾]
```

Q6. Describe Scratch Editor. Enlist its components?

Scratch Editor:

Scratch 2.0 also knowns as, Scratch 2.0, was the second major version of Scratch, following Scratch 1.4.It featured a redesigned editor and website, and it was the first version that included the online editor as well as offline editor.



Following are the different components of Scratch Editor.
Stage, ScriptArea, SpriteList, Backdrop, ScriptBlock, Script Tab.

Q7. What is stage or stage preview window?

Stage or Stage Preview Window:

It is the main working area where the sprite moves and performs actions according to the given instructions. Here we can immediately see the output of our codes. The project runs physically in this window. It is divided into X (horizontal) and Y (Vertical) coordinates. The coordinates are displayed at the bottom right corner of the stage. These coordinates indicate the position of the sprite on stage.

Q8. Describe Script Area in Scratch Editor?

Script area:

This is the area where the code or program is seen. Blocks from the palette can be dragged to this area and create scripts by placing them together. Script is the set of step wise instructions that we give to the Sprite to do a particular task. Each sprite has its own script area.

Q9. What is Script List?

Sprite list:

It displays the thumbnails of all the sprites available in a project. The blue information icon can be clicked on any sprite to change its name or behavior.

Q10. How do you add Background to a project?

Back drop:

A backdrop is a background that can be added to the stage. By default, no backdrop is added in a project. The look of the stage can be altered by adding a backdrop.

Q11. What is Script Block?

Script Block:

Script area has three different tabs. A tab is a small area that contains similar commands or options.

Q12. Define Script Tab in Scratch?

Script Tab:

The scripts tab shows you any scripts that currently exist as well as to develop new scripts pertaining to the current sprite. This area can be considered as tool box. On clicking this tab block palette is opened. The sprites are used to give commands. A command is an instruction to do specific task. In soratch these commands are shown in the form of Code Blocks in the Block Palette.

Q13. How are the Block Palletsare helpful for user?

Block Palette:

The block palette consists of every block of instruction that is built into scratch.
The commands regarding a specific task are joined together in different blocks like Motion, Looks and Sensing. Each block has associated colour that differentiate different commands.

Q14. Describe the purpose of block used in the Scratch programming language.

The following are the functions of the blocks used in block palette:

i. ***Motion:*** These codes are used to move Sprite on the stage.

ii. ***Event:*** Event trigger specific codes at a particular time or action.

iii. ***Sound:*** is used to play different sounds.

iv. ***Looks:*** These codes are used to change the appearance of sprite and backdrop.

v. ***Control:*** These codes are used to control the action on the stage.

vi. ***Sensing:*** These codes are used to sense any specific happening.

vii. ***Data:*** These are used to initialize variables and list.

viii. ***Pen:*** This used to draw lines, rectangles and other shapes.

ix. ***Operators:*** This shows the available arithmetic, logical and relational operators.

Q15. Where we can select the costume of Sprite?

Costume Tab:

By clicking on the "Costumes" tab the appearance of a sprite can also be changed into the desired costume of choice. It can also be changed by using blocks to select the sprite's costume. New costumes for the sprite can be Imported, Created, and Edited in the Scratch Paint Editor.

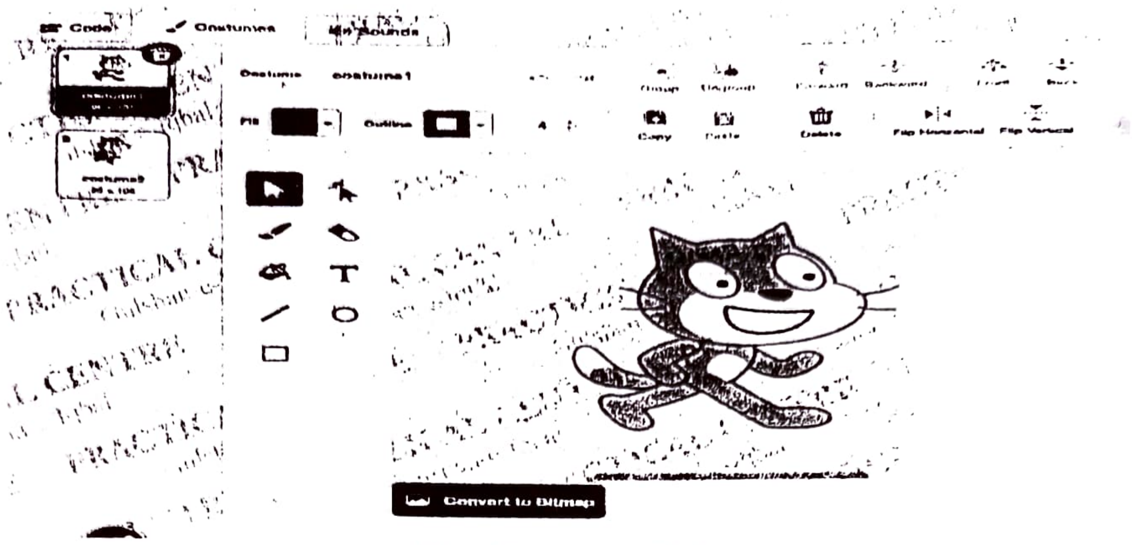


Fig. Costumes Tab

Q16. Why do we use Sound Tab in Scratch editor?

Sound Tab:

Sound is an optional field. We can have sprites with no sounds. Some sprites additionally have at least one sound. The sounds tab allows to add, delete, and edit sounds. Sounds can be played in the sound editor or with blocks that play a specific sound.

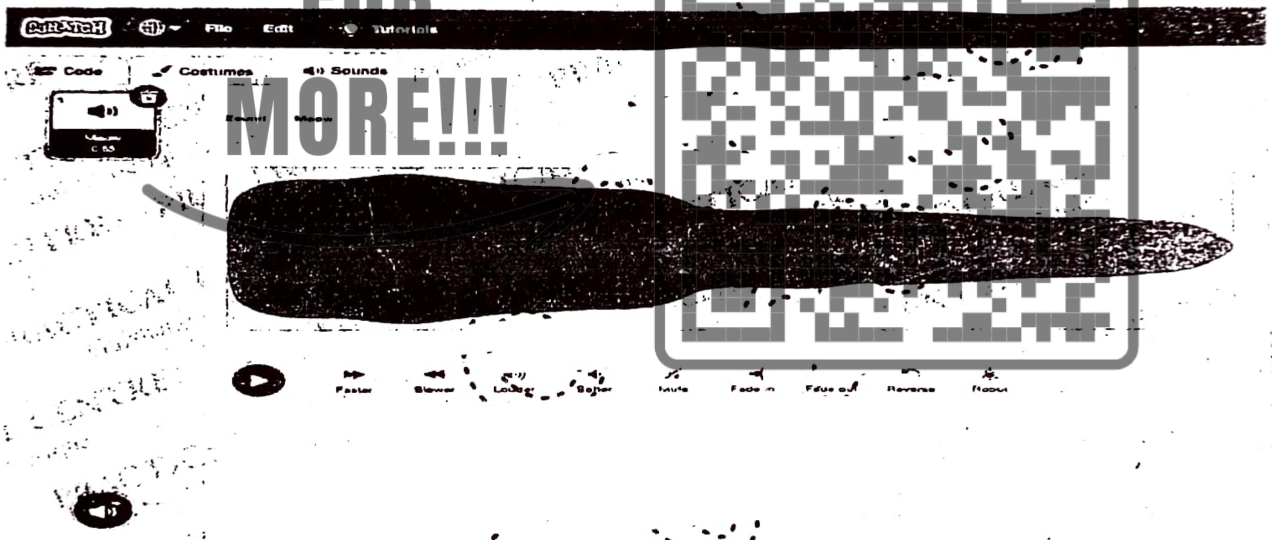
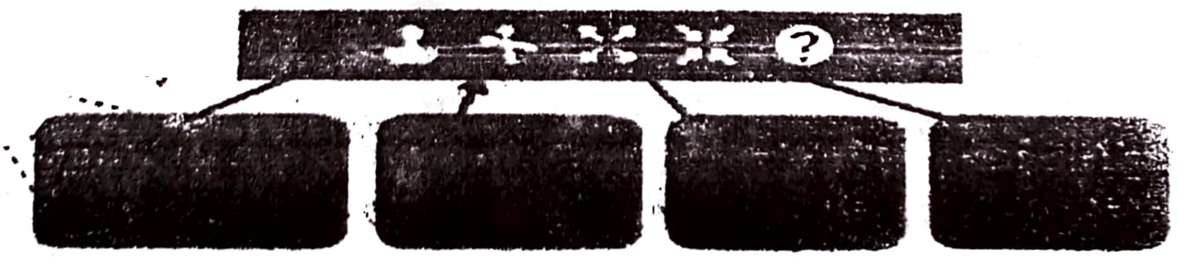


Fig.Sound Tab

Q17. Define Cursor Tool.

Cursor Tool:

Cursor Tool is located on the right top of the editor. It includes five options; Duplicate, Delete, Grow, Shrink and Help. To Duplicate a sprite, just click on the stamp and then on the Sprite. Same is applicable on Delete, Grow and Shrink.



Q.18. What points should be kept in mind before developing a project?

POINTS TO BE KEPT IN MIND BEFORE DEVELOPING A PROJECT:

The following points should be kept in mind before developing a project:

- Most of the Scratch Programs contain Sprites, Backdrops and Code Blocks.
- Every Sprite in a program has a separate Code Block which controls its actions.
- We can develop programs for different logics like storytelling, sprite animation, simple game and others.

Q19. What are the steps involved in developing a simple program?

STEPS TO DEVELOP A SIMPLE PROGRAM:

Following are the general steps to develop a program in scratch:

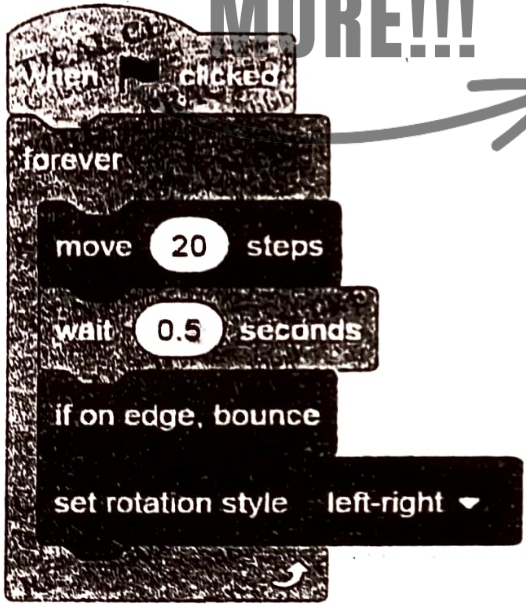
Open Scratch Editor

Events:

From Events Option in Script, drag and drop on script area.

Control:

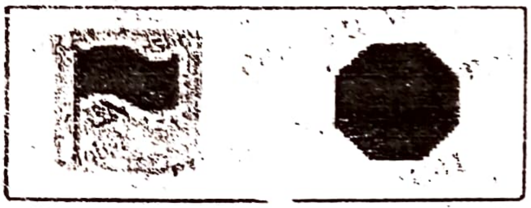
From Control Option, drag forever and drop on script area. Inside the forever block, drag and drop the illustrated commands and change values accordingly. Colors will help you to find the option. After completing the codes blocks, Play (Run) and Stop the program.



```
when [flag] clicked
forever
    move (20) steps
    wait (0.5) seconds
    if on edge, bounce
    set rotation style [left-right ▾]
```

Playing and Stopping the Animation/Program:

To start your program or to test your code click the Green flag icon located above the Stage panel. To stop your program, click the Red Stop icon.



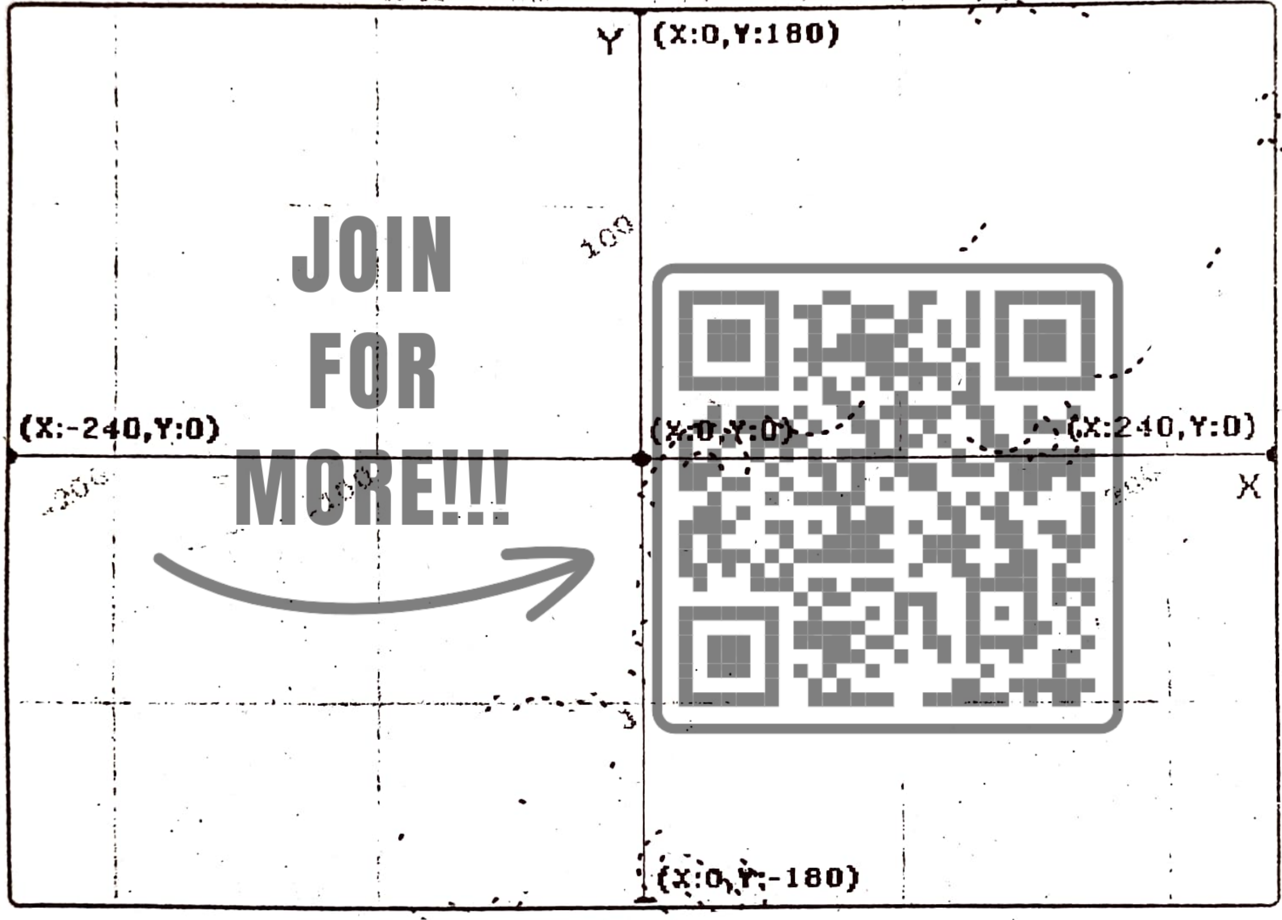
Play and Stop

Q20. What are the stage coordinates of the scratch editor?

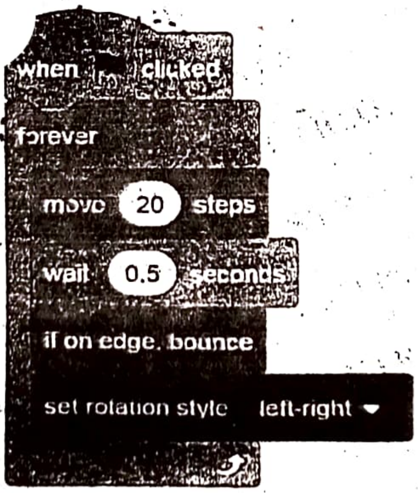
Coordinates On Scratch Stage:

Scratch has two-dimensional (2D) coordinate system; "X" position which and "Y" position to determines the location of a sprite on the stage. The "X position value determine the horizontal location of the sprite and "Y" position value determines the vertical location.

The screen in scratch is a 480 x 360 rectangle. The "X" position can range from -240 to 240, where -240 is the leftmost a sprite can be and 240 is the rightmost. "Y" position can range from 180 to -180, where 180 is the topmost it can be and -180 is the bottom-most it can be.

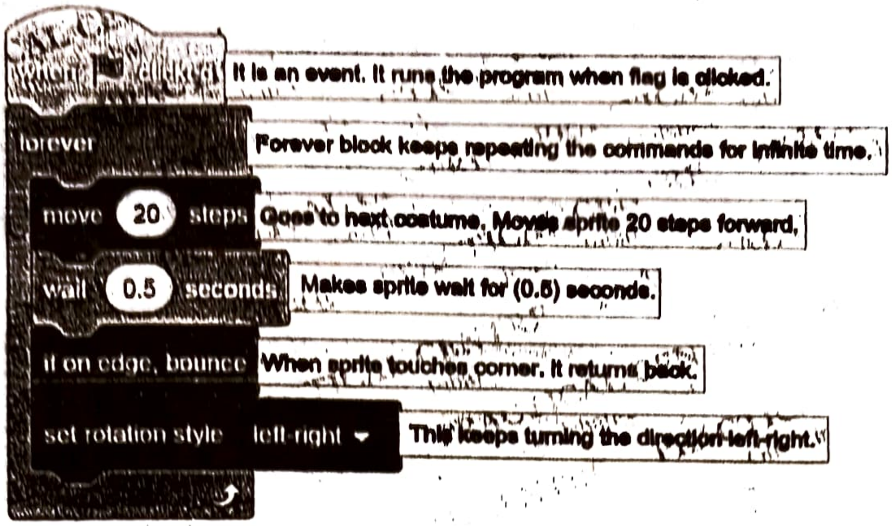


Q.21 Explain this program of scratch? Or What is the Output of this program?



```
when [flag] clicked
forever
    move (20) steps
    wait (0.5) seconds
    if on edge, bounce
    set rotation style [left-right ▾]
```

Program Explanation:



It is an event. It runs the program when flag is clicked.

Forever block keeps repeating the commands for infinite time.

Goes to next costume. Moves sprite 20 steps forward.

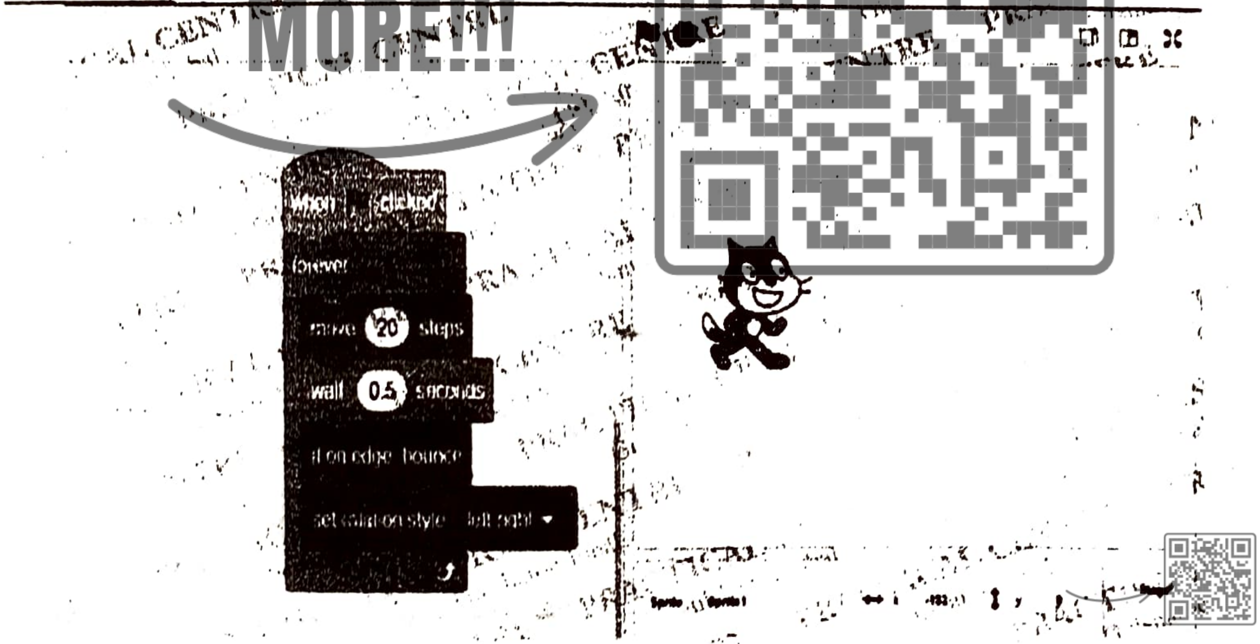
Makes sprite wait for (0.5) seconds.

When sprite touches corner. It returns back.

This keeps turning the direction left-right.

- **When clicked:** It is an event. It runs the program when flag is clicked.
- [illegible] block keeps repeating the commands for infinite time.
- [illegible] Goes to next costume. Moves sprite 20 steps forward.
- **Wait 0.5 seconds:** Makes sprite wait for (0.5) seconds.
- [illegible] When sprite touches corner. it returns back.
- [illegible] This keeps turning the direction left-right.

OUTPUT:



Q22. List some important commands in scratch with group and purpose.

SOME IMPORTANT COMMANDS IN SCRATCH:

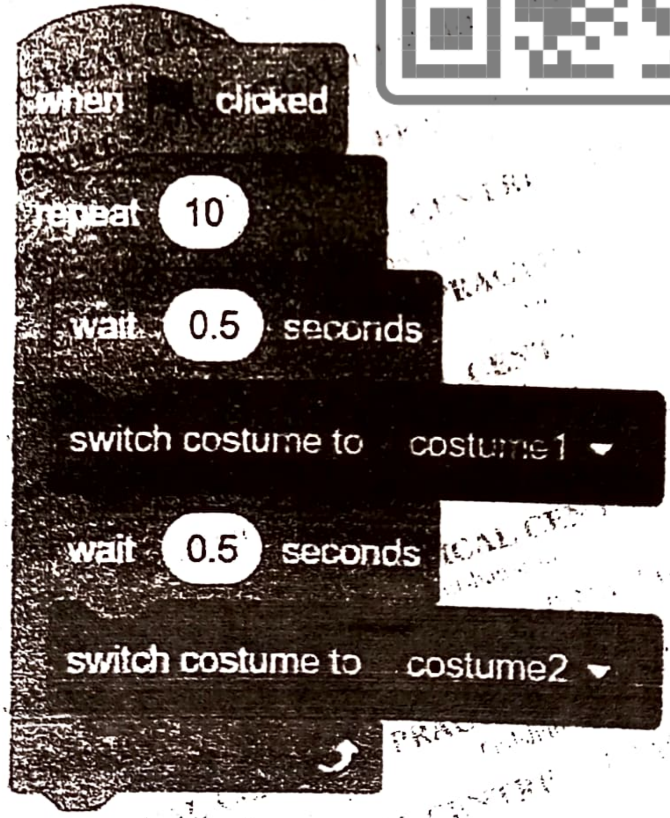
Some important commands which are usually used to developed simple programs are:

Command	Group	Purpose
when clicked	Event	It triggers the following code blocks when is clicked
when space key pressed		It triggers the following code blocks when a specific key is pressed
move 10 steps	Motion	Moves the sprite in current direction for specified steps
turn 15 degrees		Turns Sprite to specified degree
go to x: 0 y: 0		Sends Sprite to specified x, y coordinates (position on stage)
glide 1 secs to x: 0 y: 0		Sprite glides to specified x, y coordinates (posiuon on stage)
wait 1 secs	Control	Waits for specified seconds
repeat 10		Repeats the following code blocks for specified number of times.
forever		Keeps repeating the following code blocks for infinite times.
if then		Executes the block when given condition is true.
play sound meow	Sound	Plays the specified sound.
play drum 1▾ for 0.25 beats		Plays the specified drums sound for specified beats.
play note 60▾ for 0.5 beats		Plays the specified note for specified beats.
set instrument to 1▾		Changes the musical instrument.

`say Hello! for 2 secs`	Looks	Says the given words for specified seconds.
`say Hello!`		Says the given phrase.
`think Hmm... for 2 secs`		Pretends to think by showing given phrase for specified seconds.
`think Hmm...`		Pretends to think by showing given phrase.
`switch costume to costume2`		Changes the costume to the selected value.
`switch backdrop to backdrop1`		Changes the background to the selected value.
`set size to 100 %`		Changes the size of Sprite.
`pen down`	Pen	Pen is down. Now a line is drawn as Sprite moves.
`pen up`		Pen is up.
`set pen color to`		Sets the color of the pen to draw line.

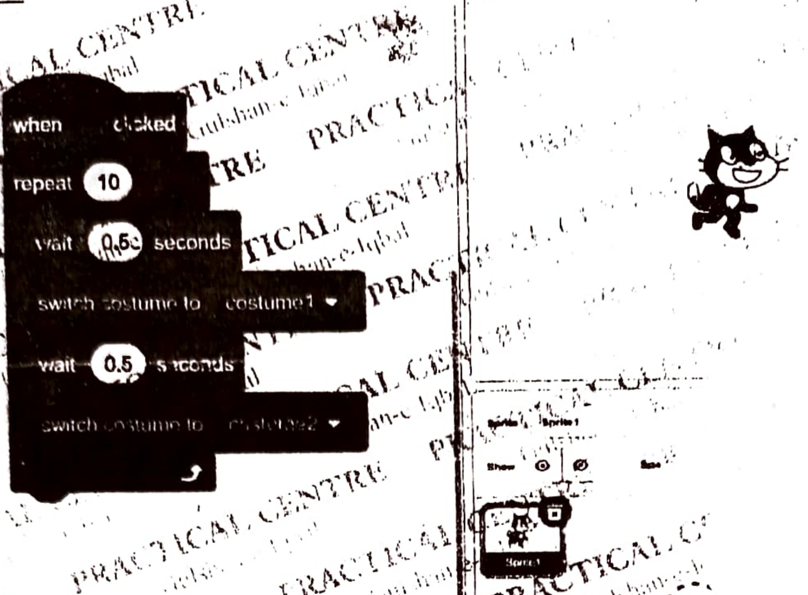
Q23. Develop a program in Scratch to change the sprite costume.

Program to change the sprite costume:



```
when [flag] clicked
repeat 10
    wait 0.5 seconds
    switch costume to costume1
    wait 0.5 seconds
    switch costume to costume2
```

OUTPUT:



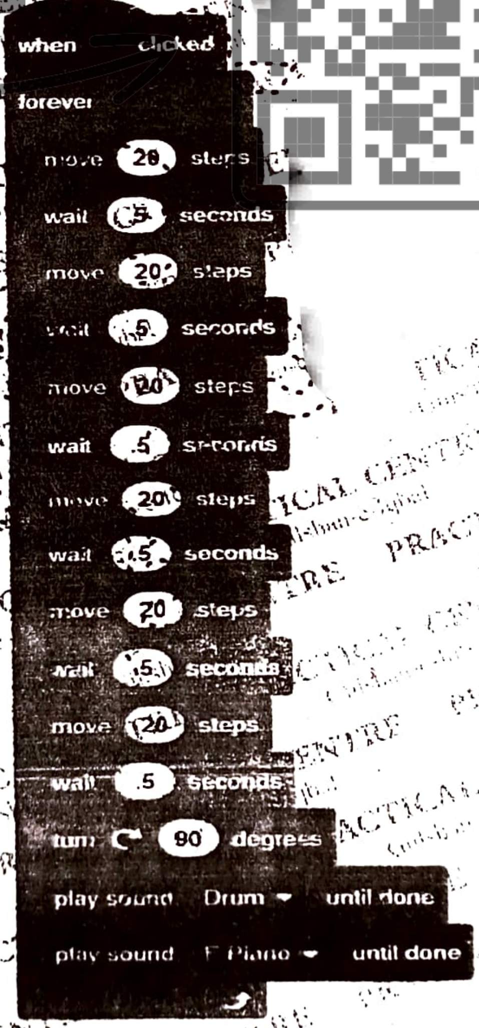
```
when [flag] clicked
repeat 10
    wait 0.5 seconds
    switch costume to costume1
    wait 0.5 seconds
    switch costume to costume2
```

Q24. Develop a program to Adding and Moving Sprite.

Program to Adding and Moving Sprite:

For the following program you should right click on the Sprite (Cat) and delete it. Now you don't have any sprite in your project.

- To add a Sprite, click on "Choose sprite from library". This will open a dialog box showing all available Sprites. Select Beetle from the list.
- On the Stage, Move the Beetle to the Upper-left corner.
- Now add the following codes.



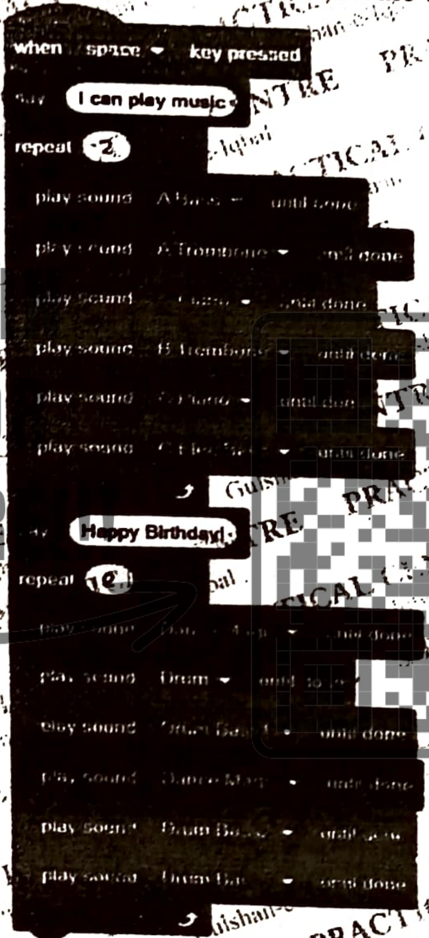
```
when [flag] clicked
forever
    move 20 steps
    wait .5 seconds
    move 20 steps
    wait .5 seconds
    move 20 steps
    wait .5 seconds
    move 20 steps
    wait .5 seconds
    move 20 steps
    wait .5 seconds
    move 20 steps
    wait .5 seconds
    turn ↻ 90 degrees
    play sound Drum until done
    play sound F Piano until done
```

Output: -
The beetle is moving clock-wise in rectangular shape.

Q25. Design a program in Scratch to play sound?

Playing Sound in Scratch:

We can add sound in our projects. It is very simple and easy. We can play different sounds like beat the drum or play different notes. You can use these sounds to make any animation, game or even just creating music. You can find complete list of notes and beats on following link.
https://en.scratch-wiki-info/
The following program, notes and drums are used to create the music of a song.

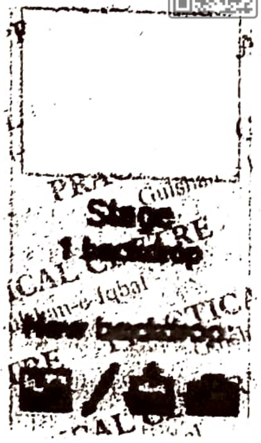


```
when space key pressed
say I can play music
repeat 2
    play sound A Bass until done
    play sound A Trombone until done
    play sound [illegible] until done
    play sound B Trombone until done
    play sound C Piano until done
    play sound [illegible] until done
say Happy Birthday!
repeat 10
    play sound [illegible] until done
    play sound Drum until done
    play sound [illegible] until done
    play sound Dance [illegible] until done
    play sound Drum Bass until done
    play sound Drum Bass until done
```

Q26. How to change the Backdrop of stage in scratch?

Changing the Backdrops:

Bydefault, the Backdrop of stage is white/blank. It can be changed. Place the cursor on "Choose a Backdrop" button and click. Itconsists of four different options:

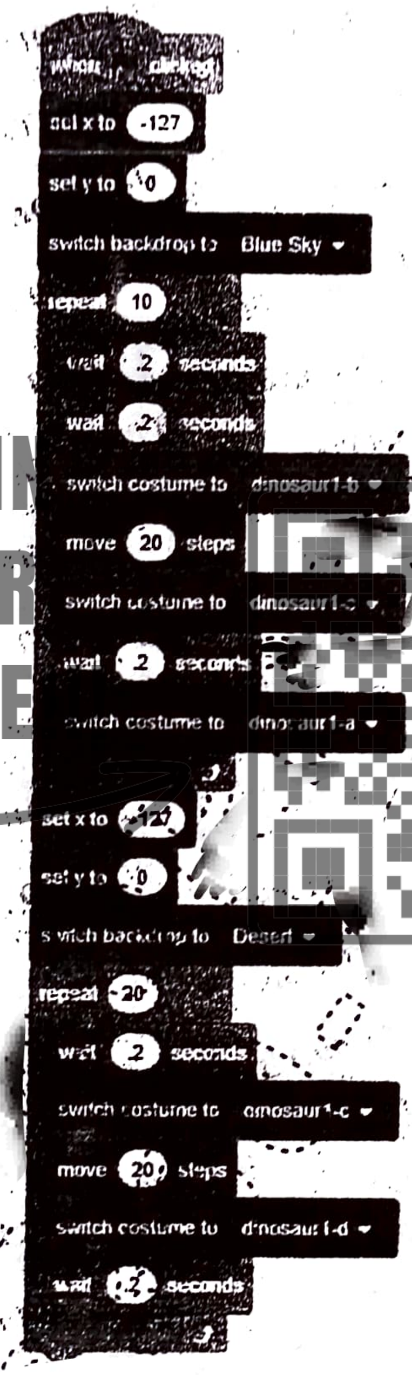


- **Choose a backdrop from library:**
  You can choose a backdrop from available library of backdrops.
- **Paint New backdrop:**
  You can also paint a new backdrop.
- **Upload backdrop from file:**
  An image on your computer can also be used as backdrop.
- **New backdrop from Camera:**
  You can also capture a picture through webcam and use it as backdrop.

Q27. Design a program in Scratch to change the Backdrops.

<u>Program to change the Backdrops in Scratch:</u>

- Add two backdrops in a program; add Blue Sky and Desert backdrop.
- Also delete cat from Sprite List and add Dinosaurs1 as Sprite.
- Now add the following codes.



```
when [flag] clicked
set x to -127
set y to 0
switch backdrop to Blue Sky ▾
repeat 10
    wait .2 seconds
    wait .2 seconds
    switch costume to dinosaur1-b ▾
    move 20 steps
    switch costume to dinosaur1-c ▾
    wait .2 seconds
    switch costume to dinosaur1-a ▾

set x to -127
set y to 0
switch backdrop to Desert ▾
repeat 20
    wait .2 seconds
    switch costume to dinosaur1-c ▾
    move 20 steps
    switch costume to dinosaur1-d ▾
    wait .2 seconds
```

<u>Output: -</u>

A dinosaur, moving lazily on your stage and as it crosses the stage, the other Backdrop (Desert) appears on the stage and the dinosaur feels happy.

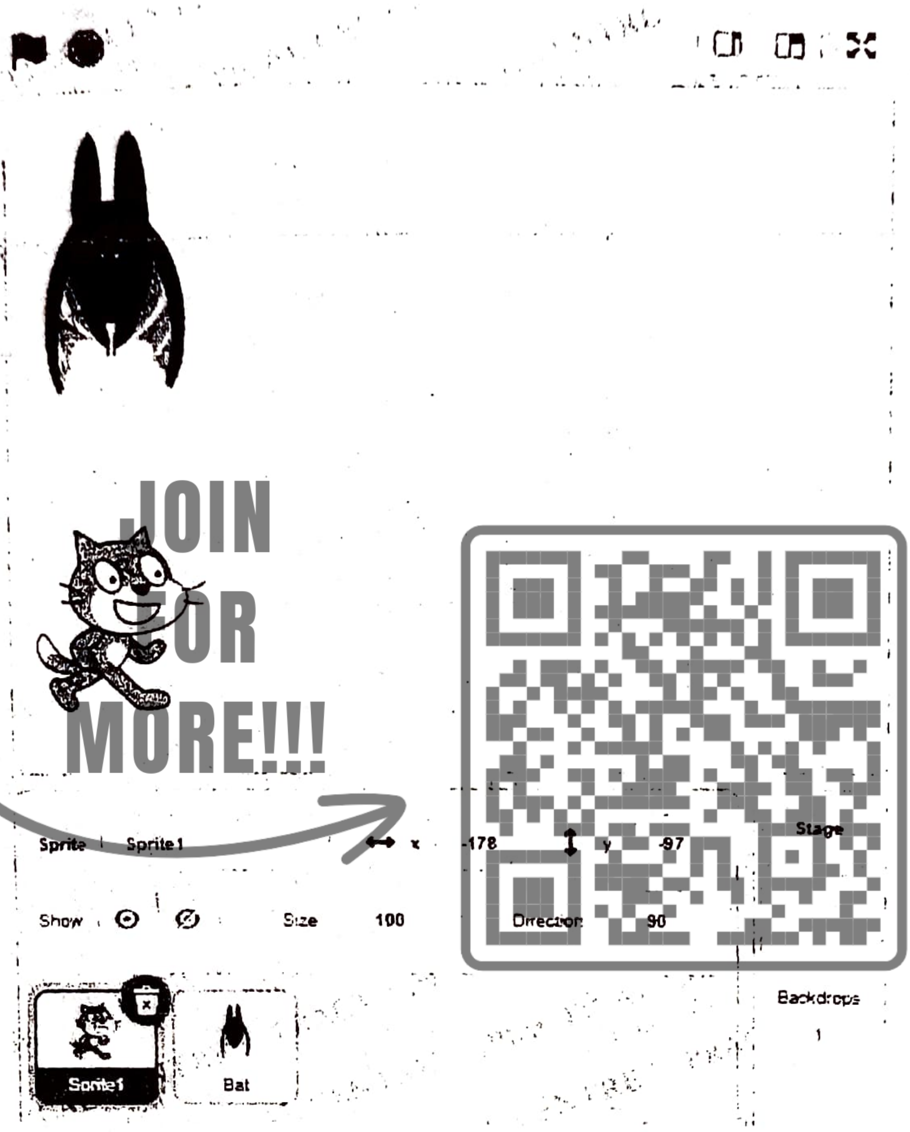
Q28. How can we use multiple sprites in our project? Design a project and execute it.

<u>Using Multiple Sprites: -</u>

We can have multiple sprites in our project. Atleast one sprite is always needed for the project. There are four ways to add different Sprites in project.

These are:

- Choose sprite from library
- Paint New Sprite
- Upload Sprite from file
- New Sprite from Camera.



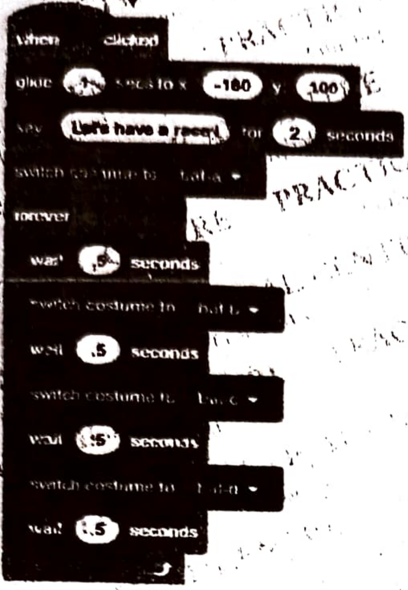
Program to add Multiple Sprites in Scratch:

Here is a program in which two Sprites are used. Follow these steps to develop this program.

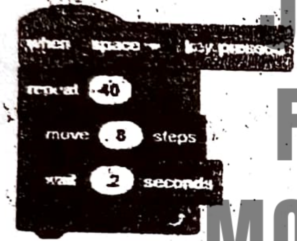
- Add Bat as the second Sprite in your Sprite List.
- This time you need to write codes for both Sprites.
- Write the following code for each Sprite.

Bat

For Bat



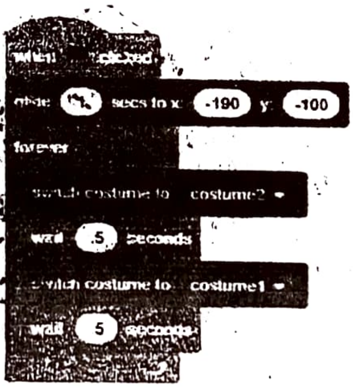
```
when clicked
glide 1 secs to x -180 y 100
say Let's have a race! for 2 seconds
switch costume to bat-a
forever
    wait .5 seconds
    switch costume to bat-b
    wait .5 seconds
    switch costume to bat-c
    wait .5 seconds
    switch costume to bat-d
    wait .5 seconds
```



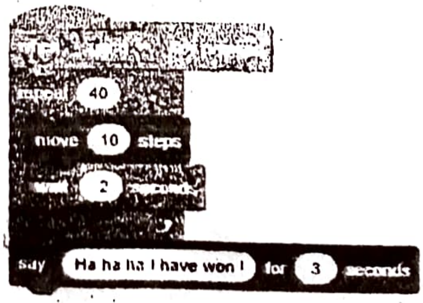
```
when space key pressed
repeat 40
    move 8 steps
    wait 2 seconds
```

Sprite 1

For Sprite 1



```
when clicked
glide 1 secs to x -190 y -100
forever
    switch costume to costume2
    wait .5 seconds
    switch costume to costume1
    wait 5 seconds
```



```
[illegible]
repeat 40
    move 10 steps
    wait 2 seconds
say Ha ha ha I have won ! for 3 seconds
```

To run this program, click on flag icon. Then press the Space Key from keyboard.

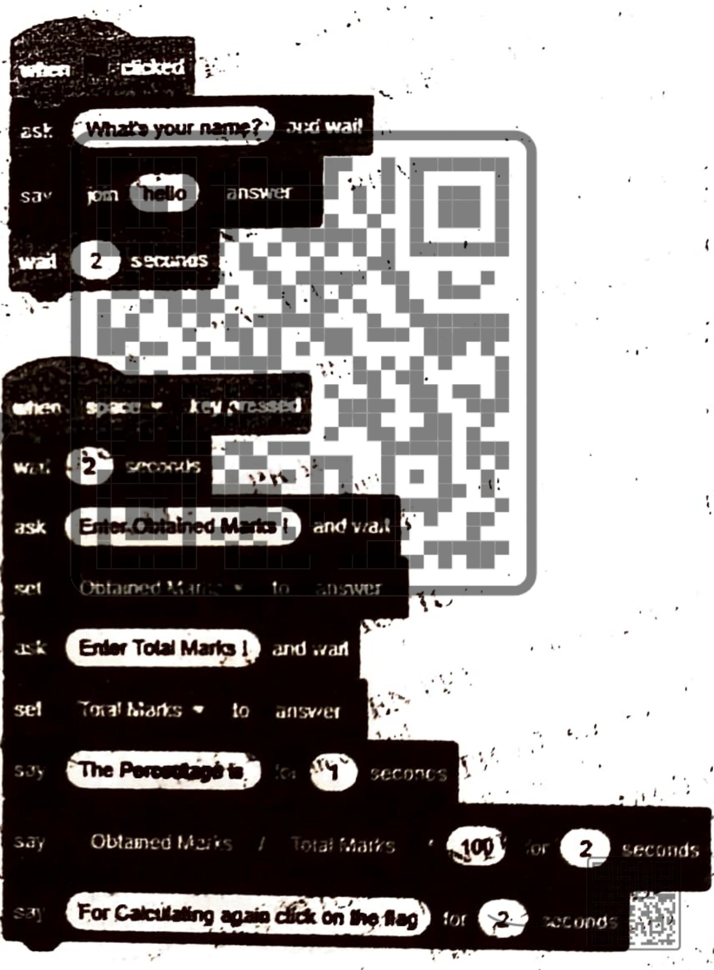
Q29. How to take input into the program?

Scratch also allows developer to take input from the user. Input can be as simple as pressing a key to make something happen. The input can be a number that will be entered as a variable and used for a calculation. You can use "ask" command in which you may write your message. You can use "answer" command with the "ask" command. The input you take in "ask" is stored in "answer" and it is displayed by using the "say" command.

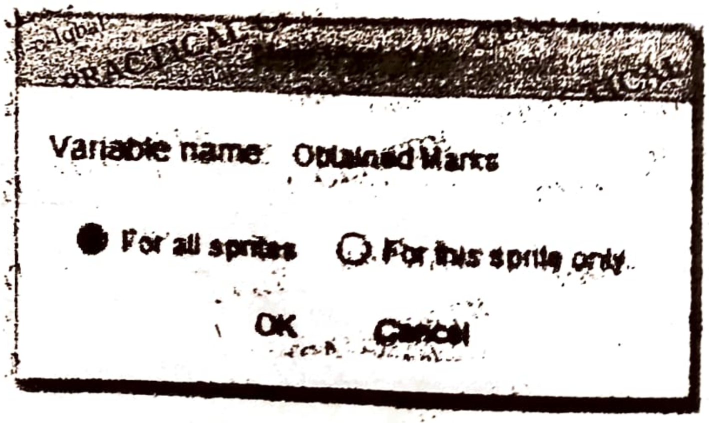
Q30. How to create a variable in Scratch Programming Language? Make a program to find the percentage of obtained marks by creating a variable.

<u>Program:</u>

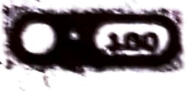
- First line will start the program.
- Add **"ask"** and **"wait"** command from Sensing Block. You may add your message in these code blocks.
- You can find **"set to"** command in **"Data Block"**. For creating a variable click on **"Make a Variable"** button. A message box will appear, which will ask you for a name of the variable. Enter a name to create the variable. Drag **"set"** command and set variable name, also add **"answer"** from **"Sensing Block"**. This line will store the value given by the user into variable called **"Obtained Marks"**.
- Same line 2
- This line will store the value given by the user into variable called **"Total Marks"**
- You can add **"say"** command from **"Looks Block"** and enter you message.
- For creating this code first add multiplication

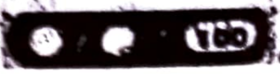
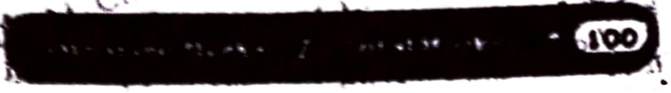


```
when [flag] clicked
ask (What's your name?) and wait
say (join (hello) (answer))
wait (2) seconds

when [space ▾] key pressed
wait (2) seconds
ask (Enter Obtained Marks !) and wait
set [Obtained Marks ▾] to (answer)
ask (Enter Total Marks !) and wait
set [Total Marks ▾] to (answer)
say (The Percentage is) for (1) seconds
say ((Obtained Marks) / (Total Marks) * (100)) for (2) seconds
say (For Calculating again click on the flag) for (2) seconds
```



Variable name: Obtained Marks

[✓ ● For all sprites] ○ For this sprite only operator  from **"Operator Block"** and put the variable **"Obtained Marks"** into the left hole.

- Now add Division Operators  on to the right hole of multiplication operator.
- Place this code  inside the **"say"** command.
- Same as line 6.
- To execute this program, clickon "Flag" sign.

MCQS

Q1. Choose the right answer:

1. What is the stage?

a) Place where actors stand and perform
b) A box you stand on when you can't see over the crowd
c) The graphical area of a scratch program
d) The coding area of a scratch program, where blocks go

2. What is this **cat type of?**

a) Animal
b) Cat species
c) Pet
d) Sprite

3. What is the coordinate of the center of the screen?

a) (X:0, Y:-180)
b) (X:0, Y:0)
c) (X:240, Y:180)
d) (X:-240, Y:0)

4. What will happen if we place the instruction blocks in the scratch at the wrong order?

a) The program will start working slowly
b) The program will not work well
c) The program will terminate
d) The speed of the program will increase.

5. What is the function of the 'Move-10 steps' command block?

a) Runs the blocks inside over and over
b) Move sprite forward
c) Runs script below when specified key is pressed
d) Move sprite backward

6. What happen to the sprite if the number increases in the 'Move_steps' command block?

a) The sprite moves quicker
b) The sprite stops
c) The sprite moves backward

d) The sprite moves slower

7. What kind of block do I use to make sprite move around?
a) Motion
b) Sound
c) Sensing
d) Control

8. How do I control sprite with the keyboard keys?
a) Mash lots of buttons until something happens
b) Call someone to do it for me
c) Use the "Move#steps" motion block
d) Use the "When <something>key pressed" events block

9. How can I make sprite use a movement animation?
a) Give it more costumes, and have them change on some event
b) Use the "when<something>key pressed" events block
c) Add a new sprite that's slightly different to the old one
d) Drag the sprite around the stage with the mouse

10. What is Scratch?
a) A programming Language
b) A Code
c) An Animation
d) A Computer System

11. A good way to get started in Scratch is to?
a) Make a complicated game
b) Create a sample project
c) Remix a sample project
d) Start from Scratch

12. As well as being a programming language Scratch is also a?
a) Tutorial
b) Community
c) Social Media
d) Project

13. How much does Scratch cost to download?
a) $24.99
b) $1.99 per month
c) $9.99 per month
d) Free

14. Where on the stage is position(X:240,Y:180)?
a) Top Left Corner
b) Bottom Right Corner
c) The Top middle
d) Top Right Corner

15. How do you create a loop in Scratch?
a) Use a Repeat Block
b) Snap a Block
c) Use a Cat Sprite
d) Use a Condition Block

16. How do you create a code in Scratch?
a) snap blocks together
b) write code in the editor
c) draws block in the editor
d) copy and paste code

17. Collaborating on your program means?
a) working with others
b) plagiarising others
c) sharing your project online
d) using example projects

18. Who develops the Scratch Software?
a) Manchester Institute of Technology
b) MontrealInstitute of Technology
c) Massachusetts Institute of Technology
d) Manchester Information Technology

19. The cat in Scratch is an example of a______.
a) Avatar
b) Character
c) Sprite
d) Template

20. In Scratch which blocks are yellow?
a) Motion
b) Control
c) Sensing
d) Variable

21. In Scratch which blocks are blue?
a) Control
b) Looks
c) Sound
d) Motion

22. In Scratch which blocks are purple?
a) Sensing
b) Looks
c) Motion
d) Variables

23. The blocks/code that controls a sprite is called the _____.
a) Lines
b) Script
c) Actions
d) Instructions

24. What do are the blocks that stores values called e.g." eaten" in shark attack?
a) Operators
b) Looks
c) Constants
d) Variables

25. To change the appearance of a sprite we can give it a different______________.
a) Costume
b) Look
c) Style
d) Stage

26. How many sprites can you add?
a) 1234
b) 3
c) 10
d) as much as

27. What do you call the character in Scratch?
a) Sprite
b) Script
c) Scratch
d) Sprite Scripts

28. Where do you go to get help in Scratch?
a) Sprite
b) Script
c) Scratch
d) Question mark

29. How many blocs in a Scratch?
a) 2
b) 4
c) 8
d) 10

30. The background of the project is called:
a) Stage
b) Script
c) Scratch
d) Backdrops

31. An organized set of rules or steps to perform an action, is also known as ___________.

a) Scratch
b) Software
c) Algorithm
d) Block

32. **which command let the sprites walk?**

a) Walk
b) Run
c) Move
d) Go!

33. The computational concept of running the same sequence multiple times.

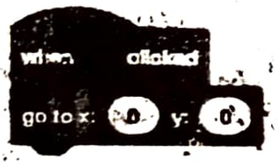
a) Parallelism
b) Events
c) Loops
d) Broadcast

34. The computational concept of one thing causing another thing to happen.

a) Parallelism
b) Events
c) Loops
d) Broadcast

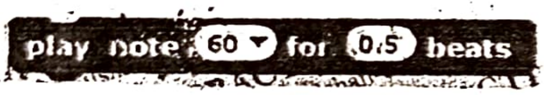
35. Which animals Scratch logo contains?

a) Koala
b) Cat
c) Lion
d) Dog

36. 

Where will the sprite go when flag is clicked?

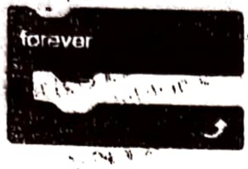
a) to the upper left corner of the stage
b) to the middle of the stage
c) to the upper right corner of the stage
d) to the bottom right corner of the stage

37. 

What does this block do?

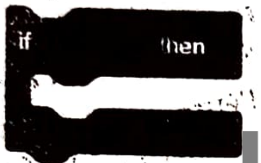
a) it writes notes for you
b) it moves the sprite
c) it plays a musical note
d) it beats the drum

38. Who created Scratch?
a) Mark Zuckerberg
b) Bill Gates
c) Google
d) Scratch Team

39. 

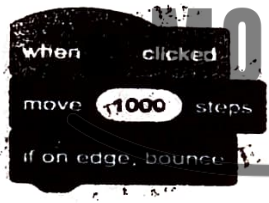
What type of block is this?

a) Iteration
b) Sequence
c) Selection
d) Variable

40. 

What type of block is this?

a) Iteration
b) Sequence
c) Selection
d) Variable

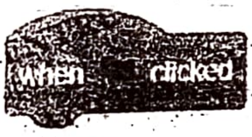
41. 

A set of three blocks placed in order is known as what?

a) Iteration
b) Sequence
c) Selection
d) Variable

42. Which one from the following indicate Scratch file extension?
a) scr
b) sh
c) scar
d) sb

43. What is a costume in a Scratch?
a) A motion
b) A different way a sprite looks
c) A code block
d) A sound

44. 

What does this block do?
a) It says hello
b) It executes the instructions when you press the green flag
c) It moves the cat
d) It asks a question

45. The variable is used for one particular sprite.
a) global
b) single
c) particular
d) local

46. What is Scratch?
a) A program that will follow your orders
b) A word processing software
c) A book
d) An spreadsheet

47. Scratch is developed by:
a) Microsoft
b) MIT
c) Apple
d) None of these

48. Why are the looks blocks used?
a) Used for drawing
b) Makes noises and plays music
c) Moving a sprite around the screen
d) Appearance of the sprite

49. How do we create a loop on Scratch?
a) Use a sprite
b) Use a wait block
c) Use an if then block
d) Use a repeat block

50. In scratch which blocks are light blue in colou
a) Motion
b) Looks
c) Variables
d) Sensing

EXERCISE

A. Encircle or choose the correct answer:

1. The feature of Scratch is ____________.

a) It is a visual

b) Its free forever

c) No need to remember codes

d) All of the above

2. In Scratch, the character which moves on the Stage is called a ____________.

a) Sprite

b) Command

c) Script

d) Event

3. Repeat 10, forever and if ... then codes are available in ____________.

a) Motion

b) Control

c) Look

d) Sensing

4. The Looks of Script can be changed by using ________.

a) Backdrop Tab

b) Costume Tab

c) Script Tab

d) Control Tab

5. A Scratch program is at least made up with _________.

a) Many Sprites

b) One Sprite and Code

c) One Sprite

d) One Sprite and Backdrop

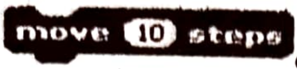
6. To change the position of Sprite on screen, we use _________.

a) Coordinates

b) Stage Information

c) Command Pallets

d) Costume Tab

7. 

can be found under __________.

a) Look

b) Motion

c) Sound

d) Control

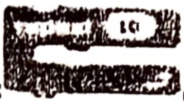
8. Turn command, turns sprite to specified:

a) Coordinates

b) Degree

c) Steps

d) Seconds

9. This  **command is available in _________**

a) Event

b) Control

c) Motion

d) Looks

10. In the given picture identify the x and y coordinate (positions) of Point – A.

a) X=100, Y=-100

b) X=-100, Y=100

c) X=100, Y=100

d) X=-100, Y=-100

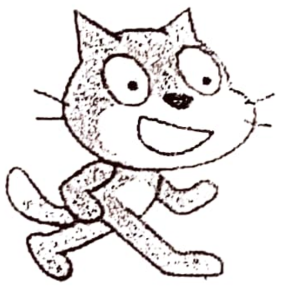
2. Respond the following:

Q1. Explain all following:

- **i. Script**
- **ii. Sprite**
- **iii. Backdrop**

i.SPRITE:

The sprites are the images of cartoons, characters or objects that we add in our project. We can have multiple sprites in our project but at least one sprite is always needed for the project. Cat is the default sprite in scratch.

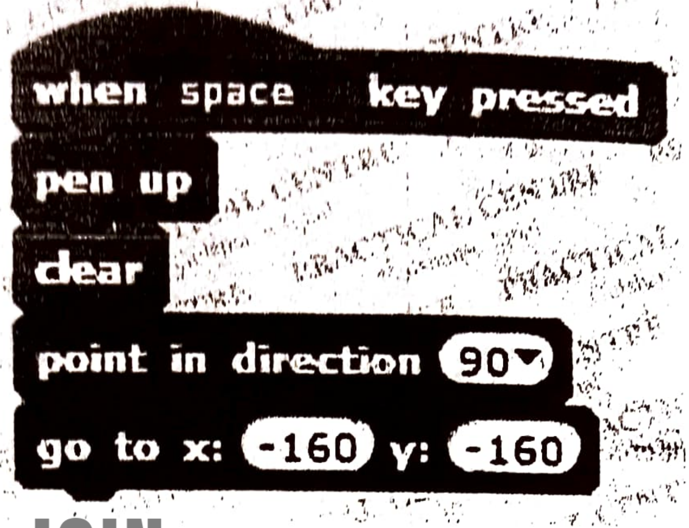


ii.SCRIPT:

To create a game, interactive story, animation or art work in scratch, We must add visual instructions to tell a sprite exactly what to do. The scripts are instructions that make sprites perform a task. Each sprite in a scratch project has an area for scripts through which it is programmed. Clicking on a script's thumbnail in the sprite pane will bring up the script area of that sprite.

OR

The scripts are the visual instructions to tell a sprite what to exactly do. In short, Scripts are the instructions in a visual form that make a sprite perform a specific task.



```
when space key pressed
pen up
clear
point in direction 90
go to x: -160 y: -160
```

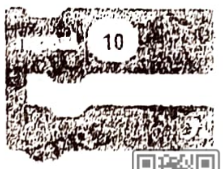
ii.BACKDROP:

A backdrop is the background that we can add on our stage. By default, there is no backdrop is added in a project. You may change how your stage looks by adding new backdrops.

Q2. State the differentiate between repeat 10 and forever commands.

Difference Between Repeat 10 and Forever Commandsis:

Forever: Forever command keeps on repeating the command for infinite times. The instructions enclosed in the forever block keep on executing till the time the project is running or the Stop button is not pressed.



Repeat 10: The instructions enclosed in the repeat block execute for the given number of times. Repeat 10 command repeats the command only for 10 times.

Q3. Write the use of the following codes: forever, wait, say, playsound, go to x, y.

Codes with Their Uses:

S.NO.	Words
forever	Keep repeating the code blocks for infinite times
wait	Makes sprite waits for specific seconds
Play sound	Used to play specific sound
say	Commands the spite to say words or phrase in a specific second
Go to x, y	Make the sprite move to specific location at x or y coordinates (position on stage).

Q4. What is the difference between using Scratch online and offline?

	Scratch Online	Scratch Offline
	In Online scratch we can communicate with the community and see shared projects of others.	While using scratch offline we can only make projects and upload or download them to USB's.
	Online we can also share our own projects	
	Discussions and chats can also be done while using on line scratch.	
	Need to the internet in online scratch	No need to the internet in offline scratch
	External memory does not require for online scratch.	External memory is required to download offline scratch.
	The script area is located on the left-hand side in the online scratch editor.	The script area is located on the right-hand side in the offline scratch editor.

Difference Between Using Scratch Online And Offline:

There is a difference between using scratch online and offline.

Scratch Online:

- In Online scratch we can communicate with the community and see shared projects of others.
- Online we can also share our own projects.
- Discussions and chats can also be done while using on line scratch.

Scratch Offline:

While using scratch offline we can only make projects and upload or download them to USB's.

Q5. In your note book mark the color of each pallet in Script Tab. One is done for you. How are these colors helpful for the user?

■	Pen

The colours in script tab are helpful for the user in the following ways:

- The colours prevent the user from getting confused.
- The colours code makes it easier for the user to use different commands.
- When there are too many scripts in one sprite the colours make it easier to find them