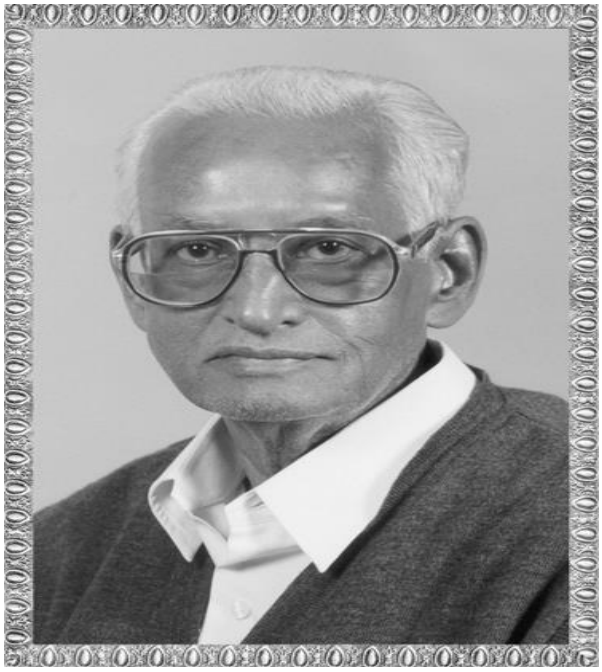
SHREE H. N. SHUKLA COLLEGE OF I.T. & MGMT.

(AFFILIATED TO SAURASHTRA UNIVERSITY)

2 – Vaishalinagar
Nr. Amrapali Under Bridge
Raiya Road
Rajkot – 360001
Ph.No–(0281)2440478,2472590

3 – Vaishalinagar
Nr. Amrapali Under Bridge
Raiya Road
Rajkot - 360001
Ph.No–(0281)2471645



Lt. Shree Chimanbhai Shukla

B.C.A./B.Sc.IT-SEMESTER-2

SUB :- DS

Shree H.N.Shukla College
Street No. 2, Vaishali Nagar,
Nr. Amrapali Under Bridge,
Raiya Road, Rajkot.
Ph. (0281)2440478, 2472590

Shree H.N.Shukla College
Street No. 2, Vaishali Nagar,
Nr. Amrapali Under Bridge,
Raiya Road, Rajkot.
Ph. (0281)2440478, 2472590

SHREE H. N. SHUKLA COLLEGE OF I.T. & MGMT.

(AFFILIATED TO SAURASHTRA UNIVERSITY)

2 – Vaishalinagar
Nr. Amrapali Under Bridge
Raiya Road
Rajkot – 360001
Ph.No–(0281)2440478,2472590

3 – Vaishalinagar
Nr. Amrapali Under Bridge
Raiya Road
Rajkot - 360001
Ph.No–(0281)2471645

CS-07: ADVANCE C AND DATA STRUCUTRE				
Sr No.	**Topic**	**Details**	**Marks**	
1	Algorithm Analysis	• The analysis of algorithm. • Time and space complexities. • Asymptotic notation. • Classes of algorithm. • Big-Oh Notation • Big-Omega Notation		

	Advanced Concepts of C and Introduction To data Structures	• Introduction
		• Data types
		• Arrays
		• Handling arrays
		  ○ Initializing the arrays
		• Multidimensional arrays
		  ○ Initialization of two dimensional array
		• Pointers
		  ○ Advantages and disadvantages of pointers
		  ○ Declaring and initializing pointers
		  ○ Pointer arithmetic
		• Array of pointers
		• Passing parameters to the functions
		• Relation between pointers and arrays
		• Scope rules and storage classes
		  ○ Automatic variables
		  ○ Static variables
		  ○ External variables
		  ○ Register variable
		• Dynamic allocation and de-allocation of memory
		  ○ function malloc(size)		

		○ function calloc(n,size) ○ function free(block) • Dangling pointer problem. • Structures. • Enumerated constants , Unions		
2	Sorting and Searching	• Bubble sorting • Insertion sorting • Quick sorting • Bucket sorting • Merge sorting • Selection sorting • Shell sorting • Basic searching technique • Sequential searching • Binary searching		
	Graph	• Introduction • Adjacency matrix and adjacency lists • Graph traversal ○ Depth first search (dfs) ○ Implementation ○ Breadth first search (bfs) ○ Implementation • Shortest path problem • Minimal spanning tree		
3	Introduction To data Structure	• Introduction • Primitive and simple structures • Linear andnonlinear structures file organization.		
	Elementary Data Structure	• Introduction • Stack ○ Definition ○ Operations on stack ○ Implementation of stacks using arrays		

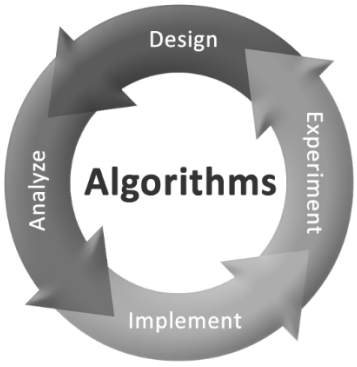
		○ Function to insert an element into the stack ○ Function to delete an element from the stack ○ Function to display the items • Recursion and stacks • Evaluation of expressions using stacks ○ Postfix expressions ○ Prefix expression • Queue ○ Introduction ○ Array implementation of queues ○ Function to insert an element into the queue ○ Function to delete an element from the queue • Circular queue ○ Function to insert an element into the queue ○ Function for deletion from circular queue ○ Circular queue with array implementation • Deques • Priority queues		
4	Link List	• Introduction • Singly linked lists. ○ Implementation of linked list ○ Insertion of a node at the beginning ○ Insertion of a node at the end ○ Insertion of a node after a specified node ○ Traversing the entire linked list		

		○ Deletion of a node from linked list • Concatenation of linked lists • Merging of linked lists • Reversing of linked list • Doubly linked list. • Circular linked list • Applications of the linked lists		
5	Tree	• Introduction ○ Basic terminology ○ Properties of a tree • Binary trees ○ Properties of binary trees ○ Implementation ○ Traversals of a binary tree ▪ In order traversal ▪ Post order traversal ▪ Preorder traversal • Binary search trees (bst) ○ Insertion in bst ○ Deletion of a node ○ Search for a key in bst ○ Height balanced tree ○ b-tree ▪ Insertion & Deletion		

2 – Vaishalinagar
Nr. Amrapali Under Bridge
Raiya Road
Rajkot – 360001
Ph.No–(0281)2440478,2472590

3 – Vaishalinagar
Nr. Amrapali Under Bridge
Raiya Road
Rajkot - 360001
Ph.No–(0281)2471645

UNIT -1 – PART – 1 - ALGORITHM ANALYSIS

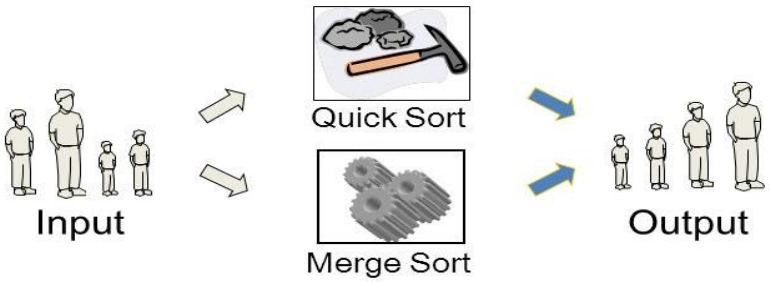


- The analysis of algorithm.
- Time and space complexities.
- Asymptotic notation.
- Classes of algorithm.
- Big-Oh Notation
- Big-Omega Notation

1. The analysis of algorithm

An **algorithm** is a step-by-step procedure for solving a problem in a finite amount of time.

How to evaluate algorithms?



- Which one is better?
- What are the criteria?

- An algorithm is a step by step sequence of instruction to solve the computational problem in a finite amount of time in an English language.
- An algorithm can be written in English but we are interested in algorithms which have been precisely specified using an appropriate mathematical formalism—such as programming language.
- Every algorithm should have the following five characteristics:
  - **i.** Input---The algorithm should take zero or more input.
  - **ii.** Output---The algorithm should produce one or more outputs.

iii. Definiteness---Each and every step of algorithm should be defined unambiguously.
iv. Effectiveness---A human should be able to calculate the values involved in the procedure of the algorithm using paper an pencil.
v. Termination---An algorithm must terminated after a finite number of steps.

- **Complexity of an algorithm** is the measure of analysis of algorithm. It is also known as computational complexity.
- Analyzing an algorithm means predicting the resources that the algorithm requires such as memory, communication, bandwidth, logic gates an time.
- The analysis of the program requires two main considerations:
  i. **Space Complexity**
  ii. **Time Complexity**

2. Time and space Complexities.

- **Time**
  - **Executing instructions take time**
  - **How fast does the algorithm run?**
  - **What affects its runtime?**
- **Space**
  - **Data structures take space**
  - **What kind of data structures can be used?**
  - **How does the choice of data structure affect the runtime?**

- The time complexity of a program/algorithm is the amount of computer time that it needs to run to completion.
- The space complexity of a program/algorithm is the amount of memory that it needs to run to completion.

i. **SPACE COMPLEXITY**
   - The amount of memory required to run and completion of an algorithm or program is known as space complexity.

- Its analysis is known as analysis of space complexity of an algorithm or program.
- There are specific reasons available for studying space complexity:
  - ✓ If the program is run on multi user system then it may be required to specify the amount of memory to be allocated to the program.
  - ✓ To know in advance that sufficient memory is available or not to run the program.
  - ✓ There may be several possible solutions with different space requirements.
  - ✓ It can be used to estimate the size of the largest problem that program can solve.
- The space needed by a program consists of following components.
  - ✓ Instruction space
  - ✓ Data space
  - ✓ Environment stack space

ii. TIME COMPLEXITY

- The time complexity of an algorithm or a program is the amount of time it needs to rum to completion.
- The exact time will depend on the implementation of the algorithm, programming language, optimizing capabilities of the compiler used and so on…
- Some of the reasons for studying time complexity are:
  - ✓ We may be interested to know in advance whether the program will provide a satisfactory real time response.
  - ✓ There may be several possible solutions with different time requirement.
- When we analyze an algorithm depends on the input data, there are three different types of time complexities which can be analyzed for an algorithm.
  - ✓ Best case time complexity
  - ✓ Average case time complexity
  - ✓ Worst case time complexity

3. Asymptotic notation

- When we study algorithms, we are interested in characterizing them according to their efficiency.
- We are usually interesting in the order of growth of the running time of an algorithm, not in the exact running time. This is also referred to as the asymptotic notation.
- We need to develop a way to talk about rate of growth of functions so that we can compare algorithms.
- Asymptotic notation gives us a method for classifying functions according to their rate of growth

4. Classes of algorithm

- **By implementation way**
  - **i.** It is further classified into following sub categories.
    - Recursion or Iteration
      - ✓ A recursive algorithm means that it invokes itself repeatedly until a certain condition matches.
      - ✓ An iterative algorithms use repetitive constructs like loops and sometimes additional data structures like stacks to solve the given problems.
    - Logical
      - ✓ The logic component expresses the axioms (maximum) that may be used in the computation and the control component determines the way in which deduction is applied to the axioms.
    - Serial or Parallel or Distributed
      - ✓ A computer which can execute on instruction of an algorithm at a time is known as serial computers. An algorithm designed for such an environment is called a serial algorithm.
    - Deterministic of Non-Deterministic
      - ✓ Deterministic algorithms solve the problem with exact decision at every step of the algorithm.

- Exact or Approximate
  - ✓ While many algorithms reach an exact solution, approximation algorithms try to find an approximation that is close to the true solution.
- Quantum Algorithm
  - ✓ This runs on a realistic model of quantum computation.
  - ✓ These algorithms use some essential feature of quantum computation such as quantum superposition or quantum embarrassing situation.

- **By design paradigm**

i. There is a number of paradigms which is different from each other. Also, it will include many different types of algorithm.

- Brute-force or exhaustive search
  - ✓ This is the natural method of trying every possible solution to see which is best.
- Divide and conquer
  - ✓ A divide and conquer algorithm repeatedly reduces into smaller problems until the problems are not enough to solve easily.
- Dynamic programming
  - ✓ When a problem shows optimal substructure, meaning the optimal solution to a problem can be constructed form optimal solutions to sub problems, and overlapping sub problems, meaning the same sub problems are used to solve many different problem instances, a quicker approach called dynamic programming avoids recomputing solutions that have already been computed
- The greedy method
  - ✓ A greedy algorithm is similar to a dynamic programming algorithm, but the difference is that solutions to the sub problems do not have to be known at each stage, instead a "greedy"

choice can be made of what looks best for the moment.

- Linear programming
  - ✓ When solving a problem using linear programming, specific inequalities involving the inputs are found and then an attempt is made to maximize some linear function of the inputs.
- Reduction
  - ✓ This technique involves solving a difficult problem by transforming it into a better known problem for which asymptotically optimal algorithms.
- Search and enumeration
  - ✓ Many problems can be modeled as problems on graphs.
  - ✓ A graph exploration algorithm specifies rules for moving around a graph and is useful for such problems.
  - ✓ This category also includes search algorithms, branch and bound enumeration and backtracking.

- **By field of study**
  - **i.** In the field of computer science has its own problem and requires efficient algorithm
  - **ii.** Related problems in one field are often studied together.
  - **iii.** Some example classes are search algorithms, sorting algorithms, merge algorithms etc…..
- **By Complexity**
  - **i.** Algorithms can be classified by the amount of time they require to complete compared to their input size.
  - **ii.** There is a wide variety of some algorithms
    - Complete in linear time
    - Some do so on in an exponential amount of time or even worse and some never hal

SHREE H. N. SHUKLA COLLEGE OF I.T. & MGMT.

(AFFILIATED TO SAURASHTRA UNIVERSITY)

2 – Vaishalinagar
Nr. Amrapali Under Bridge
Raiya Road
Rajkot – 360001
Ph.No–(0281)2440478,2472590

3 – Vaishalinagar
Nr. Amrapali Under Bridge
Raiya Road
Rajkot - 360001
Ph.No–(0281)2471645

1 WORD QUESTION ANSWER

SR.NO.	QUESTION	ANSWER
1	WHICH OF THE FOLLOWING CASE DOES NOT EXIST IN COMPLEXITY THEORY?	NULL CASE
2	AN ALGORITHM IS _____	A PROCEDURE FOR SOLVING A PROBLEM
3	AN ALGORITHM IN WHICH WE DIVIDE THE PROBLEM INTO SUBPROBLEM AND THEN WE COMBINE THE SUBSOLUTIONS TO FORM SOLUTION TO THE ORIGINAL PROBLEM IS KNOWN AS _____	DIVIDE AND CONQUER
4	AN ALGORITHM WHICH USES THE PAST RESULTS AND USES THEM TO FIND THE NEW RESULTS IS _____	DYNAMIC PROGRAMMING ALGORITHMS
5	A COMPLEXITY OF ALGORITHM DEPENDS UPON _____	TIME AND SPACE
6	AN ALGORITHM WHICH TRIES ALL THE POSSIBILITIES UNLESS RESULTS ARE SATISFACTORY IS AND GENERALLY IS TIME-CONSUMING IS _____	BRUTE FORCE
7	FOR A RECURSIVE ALGORITHM _____	A BASE CASE IS NOT NECESSARY
8	FOR AN ALGORITHM WHICH IS THE MOST IMPORTANT CHARACTERISTIC THAT MAKES IT ACCEPTABLE _____	CORRECTNESS AND PRECISION
9	IF FOR AN ALGORITHM TIME COMPLEXITY IS GIVEN BY O(1) THEN THE COMPLEXITY OF IT IS _____	CONSTANT
10	IF FOR AN ALGORITHM TIME COMPLEXITY	LINEAR

	IS GIVEN BY O(N) THEN THE COMPLEXITY OF IT IS _____	
11	WHICH ALGORITHM IS BETTER FOR SORTING BETWEEN BUBBLE SORT AND QUICKSORT?	QUICK SORT
12	PERFORMANCE BASED CRITERIA OF ALGORITHM, WHICH HAS TO DO WITH ITS COMPUTING TIME IS_____	TIME COMPLEXITY
13	PERFORMANCE BASED CRITERIA OF ALGORITHM, WHICH HAS TO DO WITH ITS STORAGE IS_____	SPACE COMPLEXITY

5. Big-Oh Notation

- "Big-O" notation was introduced in P.Bachmann's 1892 book Analytische Zahlentheorie.
- The notation works well to compare algorithm efficiencies because we want to say that the growth of effort of a given algorithm approximates the shape of standard function.
- Big –O is a characteristic scheme that measure properties of algorithm complexity performance and/or memory requirements.
- The algorithm complexity can be determined by elimination constant factors in the analysis of algorithm.
- Based on the complexity representation of the Big-O notation, the algorithm can be categorized as:
  - **i.** Constant time O(1)
  - **ii.** Logarithmic time O(log n) and o(n log n)
  - **iii.** Linear time O(n)
  - **iv.** Polynomial time $O(n^c)$
  - **v.** Exponential time$O(c^n)/O(2^n)$
  - **vi.** Cubic time$o(n^3)$
  - **vii.** Quadratic time$O(n^2)$

6. Big-Omega Notation

- Just as the big-OH notation provides an asymptotic way of saying that a function is "less than or equal to" another function, the following notations provide an asymptotic way of saying that a function grows at a rate that is "greater than or equal to" that of another.
- Let f(n) and g(n) be functions mapping non-negative integers to real numbers.
- We say that f(n) is Ω (g(n)) **(pronounce "f(n) is big-Omega of g(n)" i.e. f(n)=Ω (g(n)) if g(n) is O(f(n)),** if there is real constant c>0 and an integer constant n_0>1 such that

$$|f(n)| \geq c|g(n)|, \text{for } n \geq n_0$$

If f(n) is non-negative, we can simplify the last condition to f(n)=(Ωg(n)) {there exists positive constants such that

$$0 \leq c\, g(n) \leq f(n) \text{ for all } n,\ n \geq 0$$

- This definition allows us to say asymptotically that one function is greater than or equal to another, up to constant factor.**Ω(.)** is used to asymptotically lower bound a function. We say that **"f(n) is omega of g(n)".**

1 WORD QUESTION ANSWER

SR.NO.	QUESTION	ANSWER
1	TO VERIFY WHETHER A FUNCTION GROWS FASTER OR SLOWER THAN THE OTHER FUNCTION, WE HAVE SOME ASYMPTOTIC OR MATHEMATICAL NOTATIONS, WHICH IS_____.	BIG OMEGA Ω (F) BIG THETA Θ (F) BIG OH O (F)
2	AN ALGORITHM THAT INDICATES THE AMOUNT OF TEMPORARY STORAGE REQUIRED FOR RUNNING THE ALGORITHM,	SPACE COMPLEXITY

	I.E., THE AMOUNT OF MEMORY NEEDED BY THE ALGORITHM TO RUN TO COMPLETION IS TERMED AS_____.	
3	THE AMOUNT OF TIME THE COMPUTER NEEDS TO RUN TO COMPLETION IS KNOWN AS_____.	TIME COMPLEXITY
4	_____ALGORITHM IS ONE WHICH UTILIZES MINIMUM PROCESSOR TIME AND REQUIRES MINIMUM MEMORY SPACE DURING ITS EXECUTION.	BEST EFFICIENT
5	A FUNCTION IN WHICH F(N) IS Ω(G(N)), IF THERE EXIST POSITIVE VALUES K AND C SUCH THAT F(N)>=C*G(N), FOR ALL N>=K. THIS NOTATION DEFINES A LOWER BOUND FOR A FUNCTION F(N):	BIG OMEGA Ω (F)
6	THE COMPLEXITY OF ADDING TWO MATRICES OF ORDER M*N IS	MN
7	WHAT IS A SET OF STEPS FOR CARRYING OUT A SPECIFIC TASK CALLED?	ALGORITHM
8	AN ALGORITHM MAY HAVE _____'INPUTS' QUANTITIES.	ZERO OR MORE
9	_____ REFERS TO A FINITE SET OF STEPS, WHICH, WHEN FOLLOWED, SOLVES A PARTICULAR PROBLEM.	ALGORITHM
10	THE TWO MAIN RESOURCES THAT WE CONSIDER FOR AN ALGORITHM ARE_____.	MEMORY SPACE AND PROCESSOR TIME

2 – Vaishalinagar
Nr. Amrapali Under Bridge
Raiya Road
Rajkot – 360001
Ph.No–(0281)2440478,2472590

3 – Vaishalinagar
Nr. Amrapali Under Bridge
Raiya Road
Rajkot - 360001
Ph.No–(0281)2471645

SUMMARY

- ✓ Algorithms are the ideas behind computer programs.
- ✓ An algorithm is the thing which stays the same whether the program is in Pascal running on a Cray in New York or is in BASIC running on a Macintosh in Kathmandu!
- ✓ To be interesting, an algorithm has to solve a general, specified problem. An algorithmic problem is specified by describing the set of instances it must work on and what desired properties the output must have.

2 – Vaishalinagar
Nr. Amrapali Under Bridge
Raiya Road
Rajkot – 360001
Ph.No–(0281)2440478,2472590

3 – Vaishalinagar
Nr. Amrapali Under Bridge
Raiya Road
Rajkot - 360001
Ph.No–(0281)2471645

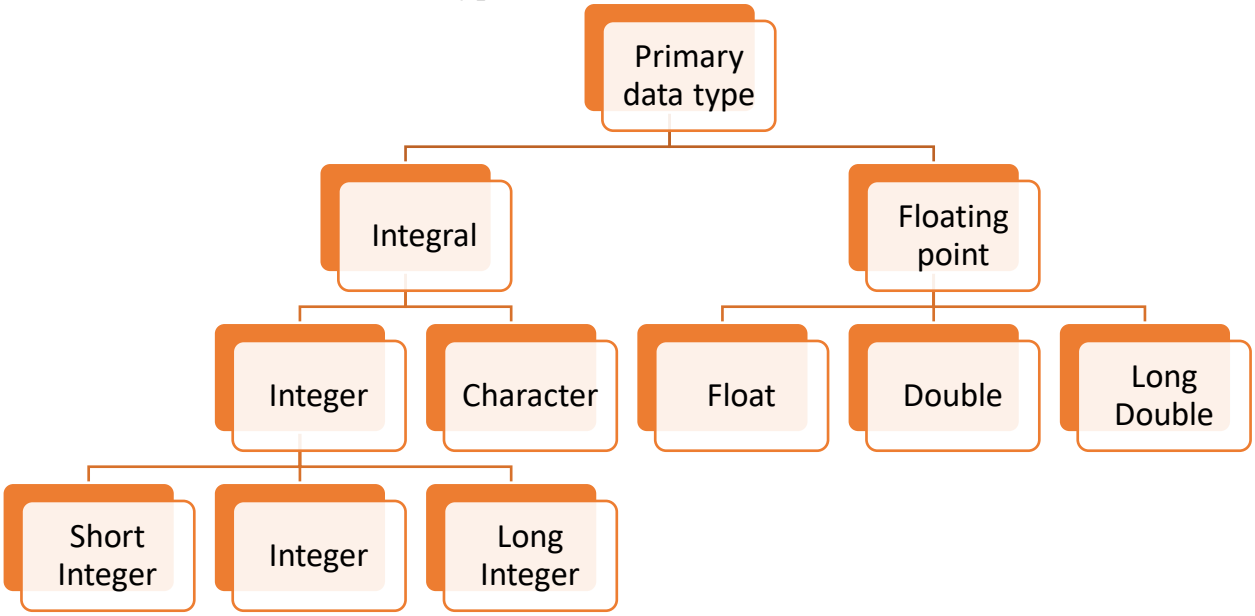
Unit -1 Part-2 (Advance Concept of C)

- Introduction
- Data types
- Arrays
- Handling arrays
  - Initializing the arrays
- Multidimensional arrays
  - Initialization of two dimensional array
- Pointers
  - Advantages and disadvantages of pointers
  - Declaring and initializing pointers
  - Pointer arithmetic
- Array of pointers
- Passing parameters to the functions
- Relation between pointers and arrays
- Scope rules and storage classes
  - Automatic variables
  - Static variables
  - External variables
  - Register variable
- Dynamic allocation and de-allocation of memory
  - function malloc(size)
  - function calloc(n,size)
  - function free(block)
- Dangling pointer problem.
- Structures.
- Enumerated constants

Unions

Data Types

- To store data in computer variable is require. Now, which type of value we want to store in variable that is known by **data type**
- C language has three basic data types:
  1. Fundamental/Primary data type
  2. Derived data type
  3. User define data type



- **Derived Data Type**
  - Derived data type are those which are provided by 'C' language to us.
  - In general we can say that we are using Array and Pointer which are derived data type of C language.
- **User Define Data Type**
  - C language provides facility to create our own new data type using **typedef**
  - We can create user define data type using following.
    - **Syntax: typedef <language data type><new data type>**
    - **Example typedef int number;**
      - Here now we can create a variable of integer type by using numeric keywords.
        - **Example: number a,b,sum;**

2 – Vaishalinagar
Nr. Amrapali Under Bridge
Raiya Road
Rajkot – 360001
Ph.No–(0281)2440478,2472590

3 – Vaishalinagar
Nr. Amrapali Under Bridge
Raiya Road
Rajkot - 360001
Ph.No–(0281)2471645

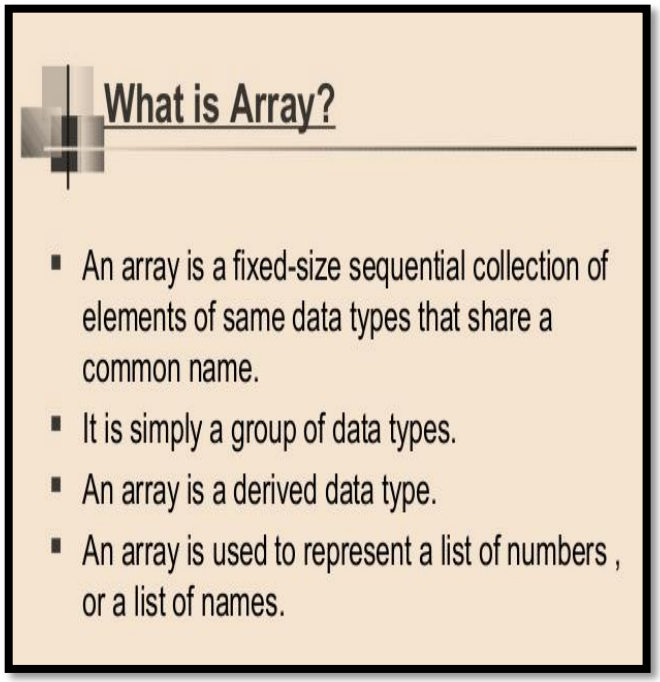
1 WORD QUESTION – ANSWER

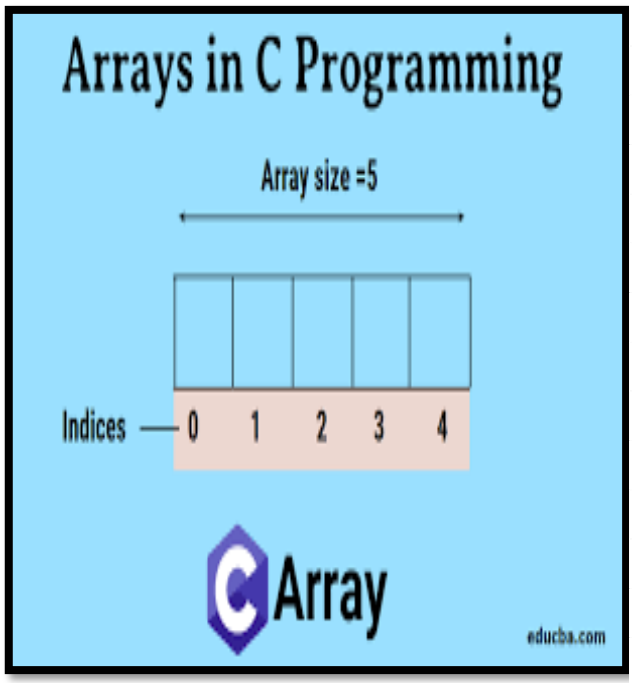
SR NO	QUESTION	ANSWER
1	DATA TYPES ARE USED TO DECIDE TYPE OF _____	DATA OR VALUE
2	C LANGUAGE SUPPORT _____ , _____ & _____ CATEGORIES OF DATA TYPE.	PRIMARY USER DEFINED DERIVED
3	TYPEDEF MEANS _____	TYPE DEFINITION
4	LIST OUR USER DEFINED DATA TYPES IN C.	STRUCTURE UNION ENUM
5	INTEGER DATATYPE OCCUPIES _____ BYTES OF MEMORY	2
6	FLOAT DATATYPE OCCUPIES _____ BYPTES OF MEMORY	4
7	CHARACTER DATATYPE OCCUPIES ___ BYTE OF MEMORY	1

2 – Vaishalinagar
Nr. Amrapali Under Bridge
Raiya Road
Rajkot – 360001
Ph.No–(0281)2440478,2472590

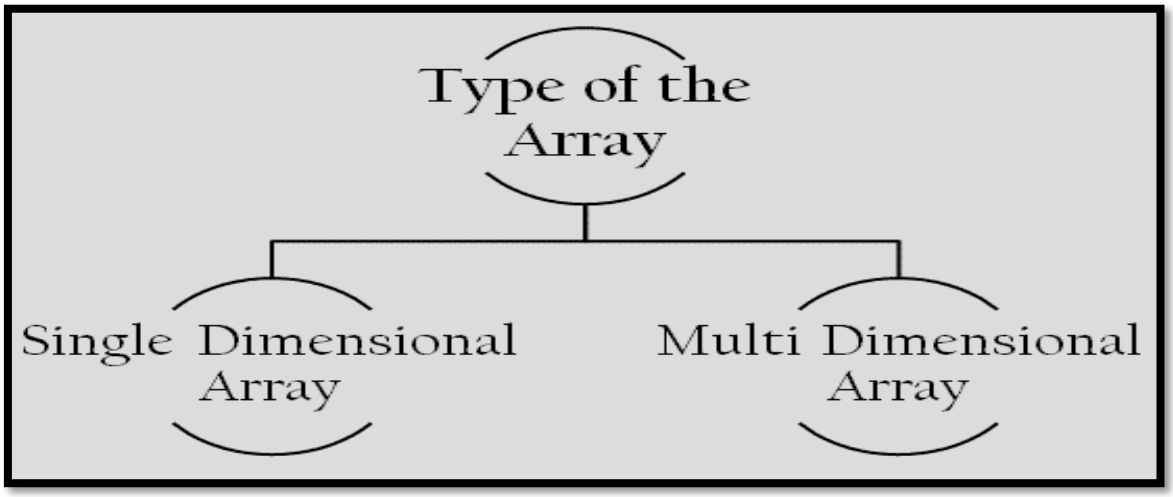
3 – Vaishalinagar
Nr. Amrapali Under Bridge
Raiya Road
Rajkot - 360001
Ph.No–(0281)2471645

Arrays





- An array is fixed size



ordered(sequence) collection of related elements(data items) that share a common name and it has same data type.

OR

- An array is useful to store similar data items in a one single name.
- There are mainly three types of array available

1. One dimensional array
2. Two dimensional array
3. Multi dimensional array

Array Element memory allocation with different data types

Data Type	Memory occupied	Example	Total Size (in Bytes)
Int	**2 bytes**	**int arr[5] (5*2)**	**10 bytes**
Float	**4 bytes**	**float a[5] (5*4)**	**20 bytes**
Char	**1 bytes**	**char ch[10] (10*1)**	**10 bytes**
Double	**8 bytes**	**double a[5] (5*8)**	**40 bytes**
Long double	**10 bytes**	**long double a[2] (2*10)**	**20 bytes**

One- Dimensional Array

- If data items in an array are arranged in a single row or line, it is called one dimensional array.

Syntax:

datatype arrayname[size];

- Here, while declaring array we require following three things.
  - A **data type** is the types of elements that will contained in the array like int, char, float, double etc.
  - An **arrayname** is name of valid array variable name and it must follow all the identifiers rules.
  - An **size** represents the maximum number of elements that can be stored inside the array
  - Example **:** **inta avg[10];**
  -

Compile time initialization of array elements

Syntax:

datatype arrayname[size]={val1,val2,val3,……};

- First way

int number[5]={1,4,6,7,3};

- Second way

int number[5]={1,2,3} //rest of elements will be set to zero

- Third way

int number[ ]={1,2,3,4,5}

- Fourth way

char nm[ ]={'h','a','r','d','I','k','\0'};

Run time initialization of array elements

```
int a[5];
for(i=0;i<5;i++)
{
        printf("\n enter array elements:");
        scanf("%d",&a[i]);
}
```

Two- Dimensional Array

- If we want to represents data in table format means rows and columns at that time two dimensional arrays is used.

Syntax:

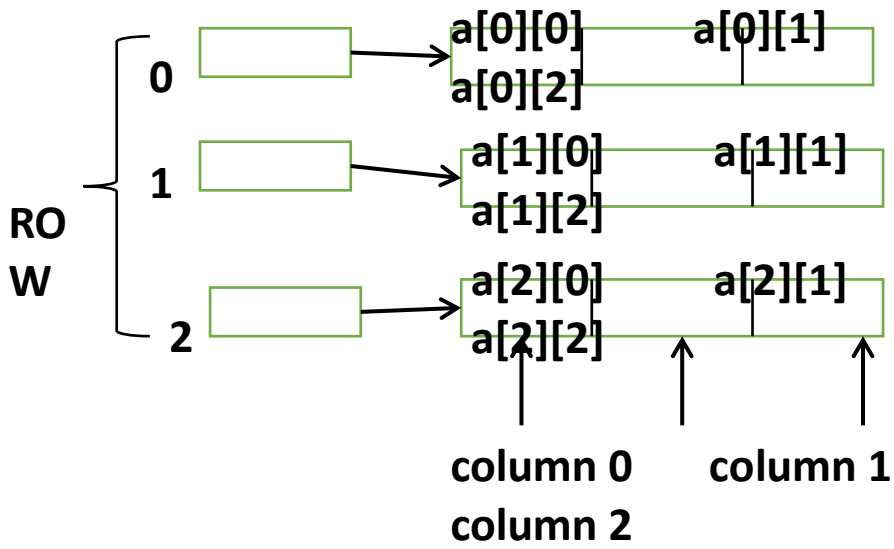
datatype arrayname[row_size][column_size];

- Here, while declaring array we require following three things.
- Initialization of array is done at compile time as well as run time.
  - Int a[2][3]={0,1,2,0,1,2};
  - Int a[2][3]={{0,1,2},{0,1,2}};

2 – Vaishalinagar
Nr. Amrapali Under Bridge
Raiya Road
Rajkot – 360001
Ph.No–(0281)2440478,2472590

3 – Vaishalinagar
Nr. Amrapali Under Bridge
Raiya Road
Rajkot - 360001
Ph.No–(0281)2471645

- Int a[2][3]={{0,1},{2}};//rest of the elements will be initialized to zero.



MULTI-DIMENSIONAL ARRAY

- Multi dimensional array is also defined same like a one-dimensional array, with a separate pair of square brackets for each subscript.
- It means that we know that two dimensional arrays will require tw0 pairs of square brackets, so if we assume three dimensional arrays then it will require three pairs of square brackets and so on.

Syntax:

```
datatype arrayname[exp1][exp2].....[expn];
```

- Example
  - float table[5][5][5]; char page[2][2][2];

2 – Vaishalinagar
Nr. Amrapali Under Bridge
Raiya Road
Rajkot – 360001
Ph.No–(0281)2440478,2472590

3 – Vaishalinagar
Nr. Amrapali Under Bridge
Raiya Road
Rajkot - 360001
Ph.No–(0281)2471645

1 WORD QUESTION – ANSWER

SR. NO.	QUESTION	ANSWER
1	WHAT IS ARRAY?	GROUP OF ELEMENTS HAVING SAME NAME AND TYPE.
2	ARRAY IS _____ DATATYPE.	DERIVED
3	ARRAY IS USED TO REPRESENT _____	COLLECTION
4	TYPES OF ARRAY CAN BE _____ & _____	SINGLE/ONE DIMENSION& MULTI/TWO DIMENSION
5	IF ARRAY ELEMENTS ARE INITIALIZED AT THE TIME OF DECLARATION THEN IT IS CALLED _____ INITIALIZATION.	COMPILE TIME
6	IF ARRAY ELEMENTS ARE INITIALIZED AT THE RUNTIME THEN IT IS CALLED _____ INITIALIZATION.	RUNTIME
7	IN _____ ARRAY INITIALIZATION WE HAVE TO ASSIGN FIX VALUE OR SIZE COMPULSORY.	COMPILE TIME

Pointers

DEFINITION OF POINTER

- The variable that holds(stores) address of other variable are called pointers.
- In other words pointer is simply variable that contains address, which is a location of another simple or ordinary variable or array elements in memory.
- A pointer is a variable that “points to” another variable by storing its address.

Pointer has three concepts

1. **Pointer constant**

In computer memory, memory addresses are known as pointer constants. We cannot change its value if we want to use it then only store its value in another variable. It is same like house number.

2. **Pointer value**

We cannot access the value of a memory address directly. If we want to access then with the use of address operator(&) with pointer variable we can do it.

3. **Pointer variable**

If we have a pointer value, we need to store into another variable. It means that the variable that contains a pointer value is called pointer variable.

ADVANTAGES OF POINTER

- ✓ Pointers are used to create dynamic data structures.
- ✓ A pointer can hold the address of any valid data element, array, single variable, a function, a structure and a union.
- ✓ Pointers are strongly associated with arrays and therefore provide an alternative way to access individual array elements.
- ✓ With the use of pointer, we can also return multiple data from a function via arguments.

Array of Pointers

- As we know that pointer contains address, so an array of pointers would be an collection of addresses.
- Syntax: datatype *array[subscript1];

datatype *array[subscript1][subscript2]….;

- Example: int *a[5];

int *arr[2][2];

Passing Parameters to the function

CALL BY VALUE	CALL BY REFERENCE
This is the usual method to call a function in which only the value of the variable is passed as an argument	**In this method, the address of the variable is passed as an argument**
Any alternation in the value of the argument passed is local to the function	**Any alternation in the value of the argument passed is accepted in the**

and is not accepted in the calling program	**calling program(since alternation is made indirectly in the memory location using the pointer)**
Memory location occupied by formal and actual arguments is different	**Memory location occupied by formal and actual arguments is same and there is a saving of memory location**
Since a new location is created, this method is slow	**Since the existing memory location is used through its address, this method is fast**
There is no possibility of wrong data manipulation since the arguments are directly used in an application	**There is a possibility of wrong data manipulation since the addresses are used in an expression. A good skill of programming is required here**

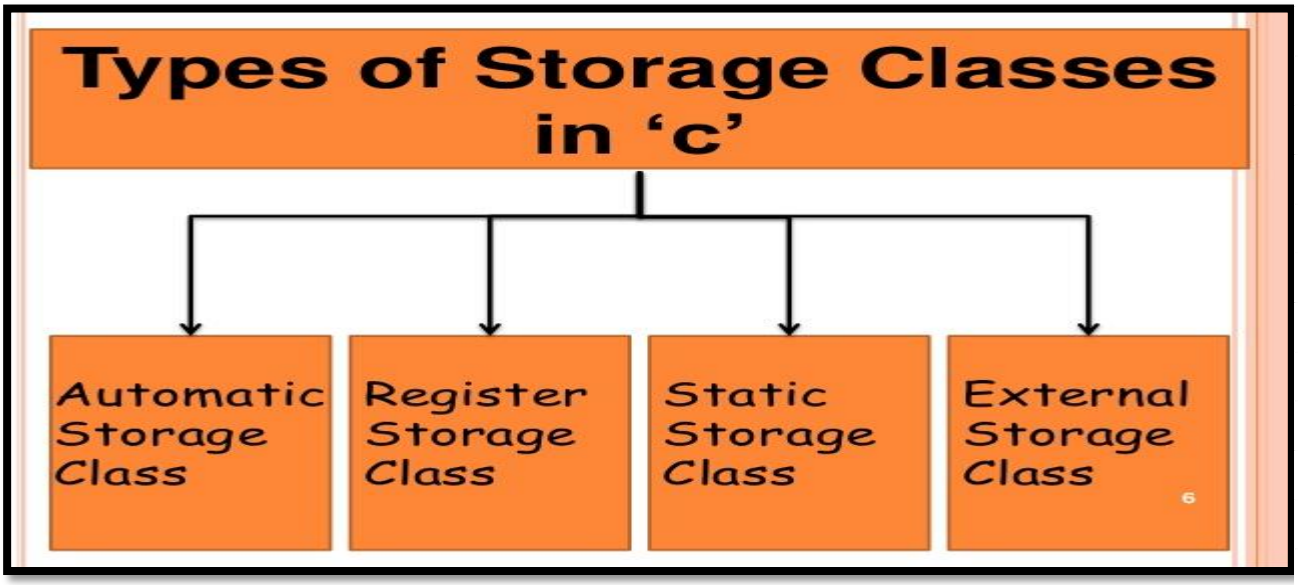
1 WORD QUESTION – ANSWER

SR. NO.	QUESTION	ANSWER
1	COMMENT ON THE FOLLOWING POINTER DECLARATION. INT *PTR, P;	PTR IS A POINTER TO INTEGER, P IS NOT
2	COMMENT ON THE FOLLOWING C STATEMENT. CONST INT *PTR;	YOU CANNOT CHANGE THE VALUE POINTED BY PTR
3	WHICH IS AN INDIRECTION OPERATOR?	*
4	WHICH DOES NOT INITIALIZE PTR TO NULL (ASSUMING VARIABLE DECLARATION OF A AS INT A=0;)?	INT *PTR = &A;

2 – Vaishalinagar
Nr. Amrapali Under Bridge
Raiya Road
Rajkot – 360001
Ph.No–(0281)2440478,2472590

3 – Vaishalinagar
Nr. Amrapali Under Bridge
Raiya Road
Rajkot - 360001
Ph.No–(0281)2471645

Scope Rules & Storage Class



Storage Class	Declaration Location	Scope (Visibility)	Lifetime (Alive)
auto	Inside a function/block	Within the function/block	Until the function/block completes
register	Inside a function/block	Within the function/block	Until the function/block completes
extern	Outside all functions	Entire file plus other files where the variable is declared as extern	Until the program terminates
static (local)	Inside a function/block	Within the function/block	Until the program terminates
static (global)	Outside all functions	Entire file in which it is declared	Until the program terminates

2 – Vaishalinagar
Nr. Amrapali Under Bridge
Raiya Road
Rajkot – 360001
Ph.No–(0281)2440478,2472590

3 – Vaishalinagar
Nr. Amrapali Under Bridge
Raiya Road
Rajkot - 360001
Ph.No–(0281)2471645

Storage class determines the scope (region of the program in which a variable is available for use.) and lifetime (duration in which a variable exist in the memory during execution of a program.). Storage class tells:

- Where the variable is stored.
- Initial value of the variable.
- Scope of the variable. Scope specifies the part of the program which a variable is accessed.
- Life of the variable.

There are four types of storage class:

1. Automatic Storage class
2. Register Storage class
3. Static Storage class
4. External Storage Class

1. Automatic Storage Class:

The variables that are local to the function are known as automatic variables i.e. declared within the function. It is the default storage class for the variables declared in a function.

- **Keyword** : auto
- **Storage Location** : Main memory
- **Initial Value** : Garbage Value
- **Life** : Local (function in which it is declared).
- **Scope** : Local to the block in which variable is declared.

Example:

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

void main()
{
        auto int i=10;
        clrscr();
        {
                auto int i=20;
```

Output:

20
10

```
            printf("\n\t %d",i);
        }
            printf("\n\n\t %d",i);
        getch();
}
```

2. Register Storage Class:

Register variables are declared inside the function. When the variables are required very often during the program, they may be declared as register variable. Register access is faster than memory access. Generally, the register variables are used as loop counters which are used multiple times in the program. Register variables are not applicable for arrays, structures or pointers.

- **Keyword :** register
- **Storage Location** : CPU Register
- **Initial Value** : Garbage
- **Life** : Local to the block in which variable is declared.
- **Scope** : Local to the block.

Example:

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

void main()
{
        register int i;
        for(i=1;i<10;i++)
        printf("%d",i);
        getch();
}
```

Output:

1 2 3 4 5 6 7 8 9 10

3. Static Storage Class:

Static storage class can be used only if we want the value of a variable to persist between different function calls. If the control comes

back to the same function again, the static variables have the same values they had last time around. By default, global variables are static variables.

- **Keyword** : static
- **Storage Location** : Main memory
- **Initial Value**: Zero and can be initialize only once.
- **Life** : depends on function calls and the whole application or program.
- **Scope** : Local to the block.

Example:

```
#include<stdio.h>
#include<conio.h>
void print();
void main()
{
        clrscr();
        print();
        print();
        getch();
}
void print()
{
        static int k=0;
        printf("\n the value of static variable is %d",k);
        k++;
}
```

Output:

value of static variable is 0
value of static variable is 1

4. External Storage Class:

The variables of external storage class can be referred to as 'global or external variables.' They are declared outside the functions and can be invoked at anywhere in a program. The only drawback of declaring the variables as extern is it wastes lot of memory because these variables remain active throughout the life of the program.

- **Keyword** : extern

- **Storage Location** : Main memory
- **Initial Value** : Zero
- **Life** : Until the program ends.
- **Scope** : Global to the program.

Example:

```
#include <stdio.h>
#include <conio.h>
extern int i=10;//global
void main()
{
        int i=20;//local
        void show();
        clrscr();
        printf("\n\t %d",i);
        show();
        getch();
}
void show()
{
        printf("\n\n\t %d",i);
}
```

Output:

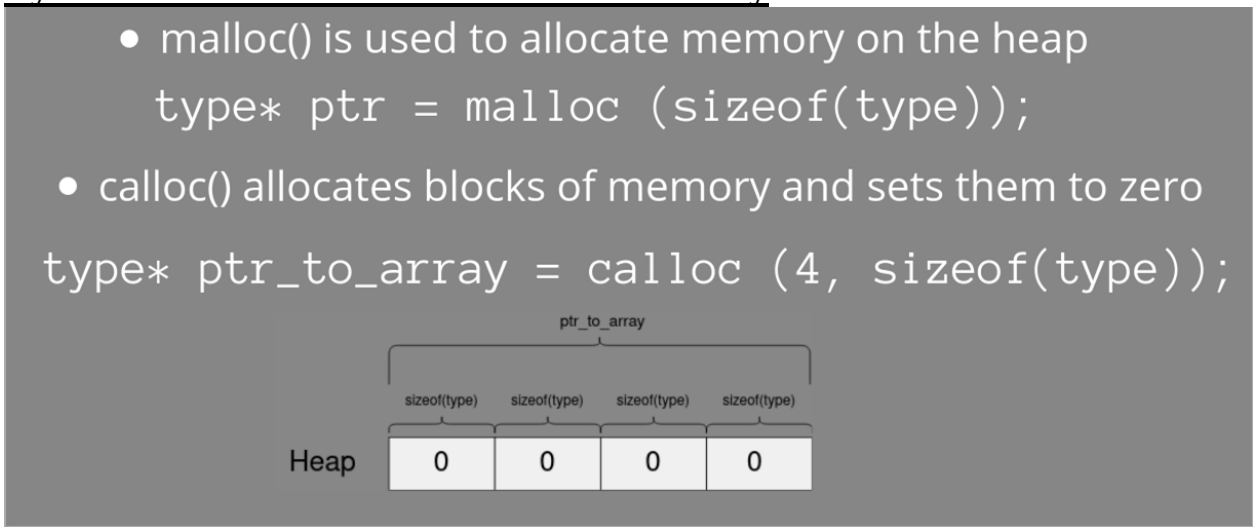
```
20

10
```

1 WORD QUESTION – ANSWER

SR. NO.	QUESTION	ANSWER
1	WHICH IS NOT A STORAGE CLASS SPECIFIER?	AUTO
2	WHAT IS THE INITIAL VALUE OF REGISTER STORAGE CLASS SPECIFIER?	GARBAGE
3	WHAT IS THE SCOPE OF EXTERN CLASS SPECIFIER?	GLOBAL MULTIPLE FILES
4	WHAT IS THE SCOPE OF STATIC CLASS SPECIFIER?	WITHIN BLOCK

5	WHAT IS THE INITIAL VALUE OF EXTERN STORAGE CLASS SPECIFIER?	0
6	HOW MANY STORAGE CLASSES IN C?	4

Dynamic Allocation & De-allocation of Memory



- DMA means Dynamic Memory Allocation. DMA means allocating the memory dynamically (run time).
- DMA is used when we don't know in advance the number of variables that we will require.
- DMA saves the memory because in this the memory is allocated at run time.
- The functions of DMA are:
  - malloc()
  - calloc()
  - realloc()
  - free()

1)**malloc():-**

Description:

malloc() is used to allocate the memory dynamically that is at run time.

Syntax:

p=(int *)malloc(n*sizeof(int));

Header File: <stdlib.h>, <alloc.h>

Example:

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
void main()
{
        int *array;
        int n,i;
        clrscr();

        printf("\n enter the size of array");
        scanf("%d",&n);
        array=(int *)malloc(n*sizeof(int)); //memory
allocation

        for(i=0;i<n;i++)
        {
                printf("\n enter array elements");
                scanf("%d",&array[i]);
        }

        for(i=0;i<n;i++)
        {
                printf("\n array elements=%d",array[i]);
        }
getch();
}
```

2) calloc():

Description:

- calloc () is also used to allocate the memory dynamically (run time) but calloc() and malloc() have the following differences:

2 – Vaishalinagar
Nr. Amrapali Under Bridge
Raiya Road
Rajkot – 360001
Ph.No–(0281)2440478,2472590

3 – Vaishalinagar
Nr. Amrapali Under Bridge
Raiya Road
Rajkot - 360001
Ph.No–(0281)2471645

malloc()	*calloc()*
1) malloc() has only one argument. 2) malloc() allocates single block of storage space 3) malloc() gives the garbage value of memory	1) calloc() has two arguments. 2) calloc() allocates multiple blocks of storage space. 3) calloc() initializes the memory to zero.

Syntax:

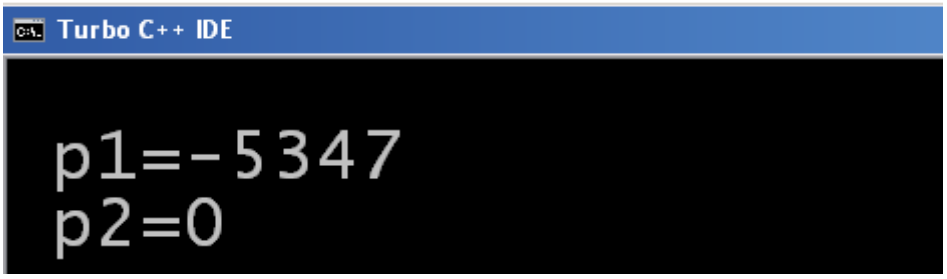
p=(int *)calloc(n,sizeof(int));

Header File: <stdlib.h>, <alloc.h>

Example showing the difference between malloc() and calloc() :

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
void main()
{
        int *p1;
        int *p2;
        int n=5;
        clrscr();
        p1=(int *)malloc(n*sizeof(int));
        p2=(int *)calloc(n,sizeof(int));

        printf("\n p1=%d",*p1);
        printf("\n p2=%d",*p2);
getch();
}
```

Output:



3) realloc():

Description:

- realloc() is used to change the size of previously allocated memory(malloc() or calloc()).
- The new size of the memory can be smaller or larger. If the size of memory is larger then it is appended at the end of the data that is data will not be changed but if the size of the memory is smaller then the data will be changed. This is known as reallocation of memory.

Syntax:

```
p=(int *)realloc(p,50);
```

new size to be allocated

Original block of memory

Header File:

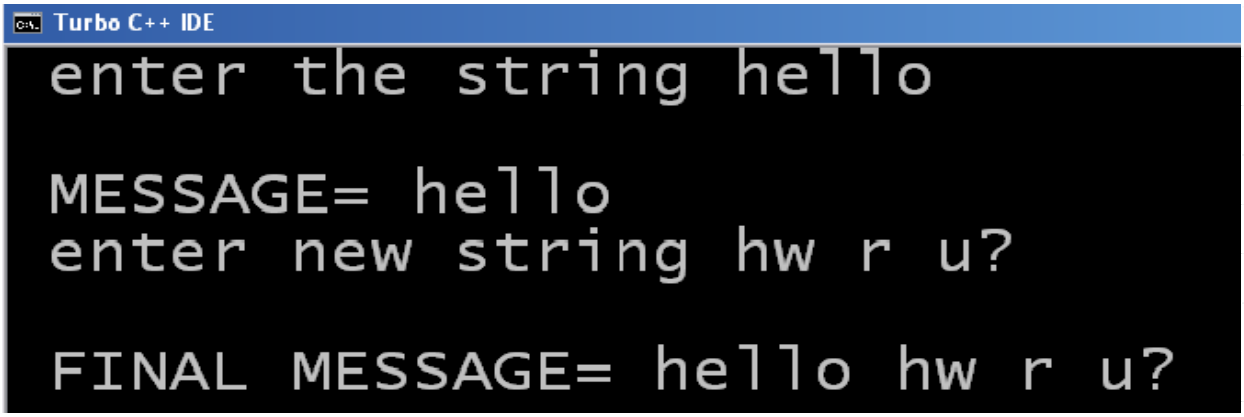
<stdlib.h>, <alloc.h>

Example:

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#include<string.h>

void main()
{
        char s[50];
        char *msg;
        clrscr();
        printf("\n enter the string");
```

```
        gets(s);
        msg=(char *)malloc(strlen(s)+1);
        strcpy(msg,s);
        printf("\n MESSAGE=%s",msg);
        printf("\n");
        printf("\n enter new string");
        gets(s);
        msg=(char *)realloc(msg,(strlen(msg)+strlen(s)+1));
        strcat(msg,s);
        printf("\n FINAL MESSAGE=%s",msg);
    getch();
    }
```

Output:



4) free():

Description:

- When memory is allocated using malloc() and calloc() then that memory must be freed when the program finishes using free() because the memory area is finite.

Syntax:

```
free(p1);
p1=NULL;
```

2 – Vaishalinagar
Nr. Amrapali Under Bridge
Raiya Road
Rajkot – 360001
Ph.No–(0281)2440478,2472590

3 – Vaishalinagar
Nr. Amrapali Under Bridge
Raiya Road
Rajkot - 360001
Ph.No–(0281)2471645

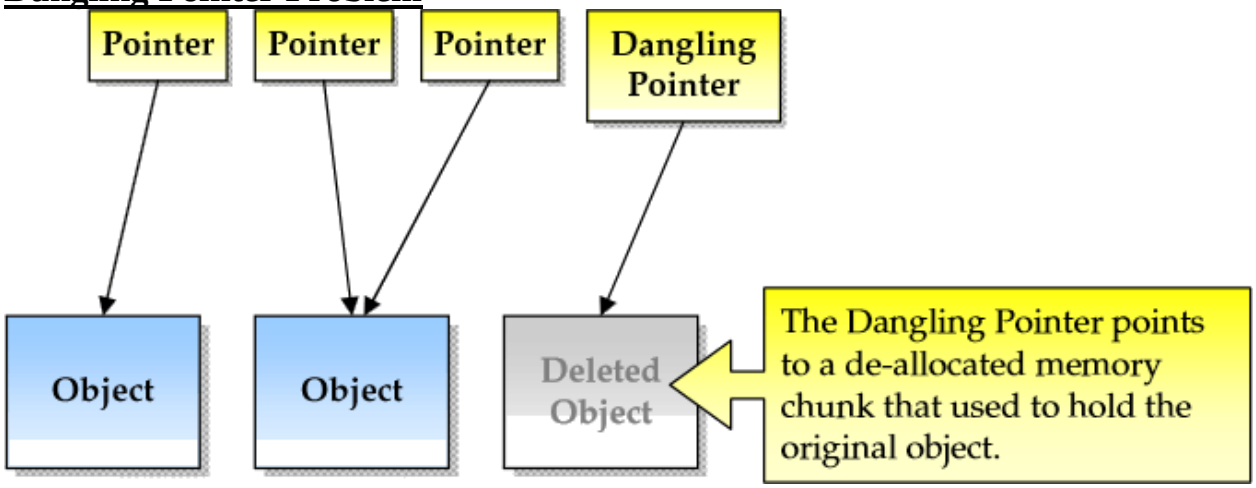
Static Memory Allocation	*Dynamic Memory Allocation*
1) In static memory allocation, size is fixed which can not be changed during runtime. 2) In static memory allocation, memory is allocated at compile time. 3) In static memory allocation, memory wastage takes place. 4) In static memory allocation, there is no need to free the memory.	1) In Dynamic memory allocation, size can be changed during runtime. 2) In dynamic memory allocation, memory is allocated at run time. 3) In dynamic memory allocation, memory is saved. 4) In dynamic memory allocation, memory must be freed using free().

1 WORD QUESTION – ANSWER

SR. NO.	QUESTION	ANSWER
1	THE FUNCTION ____ OBTAINS A BLOCK OF MEMORY DYNAMICALLY.	CALLOC & MALLOC
2	VOID * MALLOC(SIZE_T N) RETURNS?	POINTER TO N BYTES OF UNINITIALIZED STORAGE
3	CALLOC() RETURNS STORAGE THAT IS INITIALIZED TO.	L
4	IN FUNCTION FREE(P), P IS A _____	POINTER RETURNED BY MALLOC() & CALLOC()

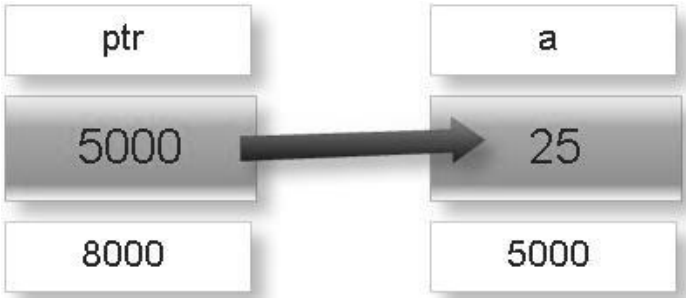
5	CALLOC() INITIALIZE MEMORY WITH ALL BITS SET TO ZERO.(TRUE OR FALSE)	TRUE
6	MEMORY ALLOCATION USING MALLOC() IS DONE IN _____	HEAP AREA

Dangling Pointer Problem

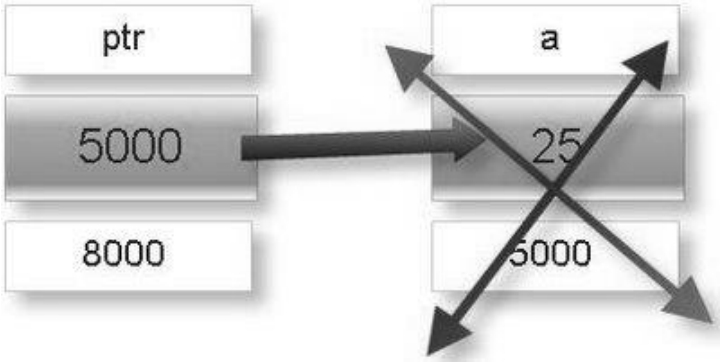


- If any pointer is pointing the memory address of any variable but after ne variable has deleted from that memory location while pointer is still poi ng such memory location. Such pointer is known as dangling pointer an his problem is known as dangling pointer problem.

Initially:



Later:



Structure

- A structure is a collection of variables under a single name. These variables can be of different types, and each has a name which is used to select it from the structure. A structure is a convenient way of grouping several pieces of related information together.
- A structure can be defined as a new named type, thus extending the number of available types. It can use other structures, arrays or pointers as some of its members, though this can get complicated unless you are careful.

Enumerated Constants

- Enumeration type allows programmer to define their own data type . Keyword enum is used to defined enumerated data type.
- enum type_name{ value1, value2,...,valueN };
- Here, *type_name* is the name of enumerated data type or tag. And *value1*, *value2*,....,*valueN* are values of type *type_name*.
- By default, *value1* will be equal to 0, *value2* will be 1 and so on but, the programmer can change the default value as below:

```
enum suit{
        club=0;
        diamonds=10;
        hearts=20;
        spades=3;
};
```

- **Declaration of enumerated variable**

Above code defines the type of the data but, no any variable is created. Variable of type enum can be created as:

```
enum boolean{
        false;
        true;
};
enum boolean check;
```

- Here, a variable check is declared which is of type **enum boolean**.

```
#include<stdio.h>
enum week{ sunday, monday, tuesday, wednesday, thursday, friday,
saturday};
int main(){
enum week today;
today=wednesday;
printf("%d day",today+1);
return0;
}
```

Output

4 day

Unions

- Unions are quite similar to the structures in C. Union is also a derived type as structure. Union can be defined in same manner as structures just the keyword used in defining union in **union** where keyword used in defining structure was **struct.**

```
union car{
  char name[50];
  int price;
};
```

Union variables can be created in similar manner as structure variable.

```
union car{
```

```
        char name[50];
        int price;
      }c1, c2, *c3;
OR;
      union car{
        char name[50];
        int price;
      };
-------Inside Function----------
      union car c1, c2, *c3;
```

In both cases, union variables *c1*, *c2* and union pointer variable *c3* of type **union car** is created.

Accessing members of an union

The member of unions can be accessed in similar manner as that structure. Suppose, we you want to access price for union variable *c1* in above example, it can be accessed as c1.price. If you want to access price for union pointer variable *c3*, it can be accessed as (*c3).price or as c3->price.

1 WORD QUESTION – ANSWER

SR. NO.	QUESTION	ANSWER
1	WHICH IS A COLLECTION OF DIFFERENT DATA TYPES?	STRUCTURES
2	USER-DEFINED DATA TYPE CAN BE DERIVED BY_____	STRUCT ENUM TYPEDEF
3	WHICH OPERATOR CONNECTS THE STRUCTURE NAME TO ITS MEMBER NAME?	L . (DOT)
4	WHICH CANNOT BE A STRUCTURE MEMBER?	FUNCTION
5	THE SIZE OF A UNION IS DETERMINED BY THE SIZE OF THE _____	BIGGEST MEMBER IN THE UNION

6	MEMBERS OF A UNION ARE ACCESSED AS_____	UNION-NAME.MEMBER & UNION-POINTER->MEMBER

SUMMARY

- ✓ To make programming very simple and easy to debug then it is required to break out large program into smaller sub programs that perform well defined tasks which is known as function.
- ✓ OR It is self contained executable codes that can be called form any other functions.
- ✓ There are two types of functions supported by C language.
  - **Library function:** It is a directly used in our program because already stored in a header files.For example printf(),scanf(),getch(),clrscr() etc….
  - **User Define function:**We have to write coding for user defined function. For example, main() is a user define function.
- ✓ Any user define function has mainly three components
  - **Function declaration:**If we want to call(use) any function before that is must be declated an then after we can use it, same like as variable. This is as function declaration.
  - **Function call:**If we want to use function definition(program) than we need to invoke it at a specified place in program. This is known as function call. The program or function that call the function is known as calling program or calling function. If function call does not require any arguments, an empty pair of parenthesis must follow the functions name
  - **Function definition:**It is one type of program in which we have to write the coding of main functionality of our program. It is also known as function implementation. It divides into two main parts.
    - **Function Header:**It contains main three elements
      - **Function name** is the name of the function.

- **Function type** is which type of function will return value
- **Parameter list** is the argument passing if required. It becomes the input data to the function to complete the specified task. It is also known as formal parameter.

▪ **Function Body:**It contains one or more statement. It contains three elements:

- **Local variable declaration:** is a one kind of variable which contains any data and its scope is limited to that particular block of code only.
- **Function executable statement** to perform specific task it is specified task it is required to write some instruction which describe in this section.
- **A return statement** if our function return a value to the calling function then it is done with the use of the return statement. A function may return a value or may not return value to the calling program. If function doesn't return any value means that void specifier is used in the function declaration. It is possible to pass more than one arguments to the function but can return only one value at a time. If functions return type is not specified then its default type is "int" means it returns integer value.

✓ Depending on whether arguments are passed to a function or not and whether a function returns a value or not it divides a function into four types as below:

- Function with no argument and no return value
- Function with argument and no return value
- Function with argument and with return value
- Function with no argument with return value

✓ When we pass data from one function to another then it is known as parameter passing. It is done in two ways.

2 – Vaishalinagar
Nr. Amrapali Under Bridge
Raiya Road
Rajkot – 360001
Ph.No–(0281)2440478,2472590

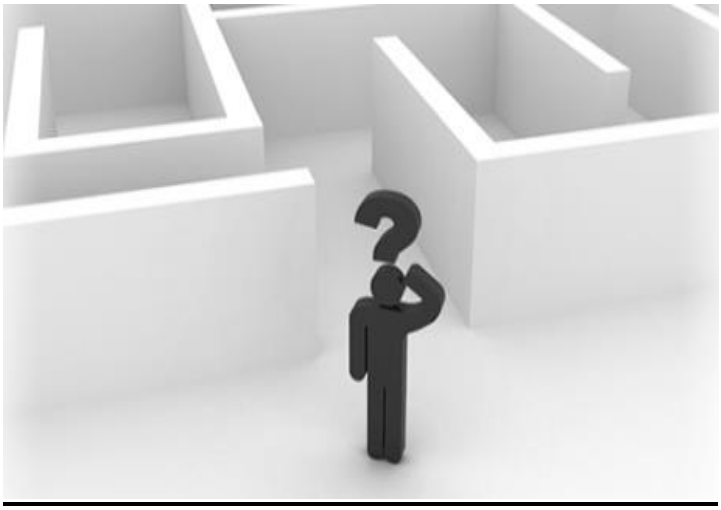
3 – Vaishalinagar
Nr. Amrapali Under Bridge
Raiya Road
Rajkot - 360001
Ph.No–(0281)2471645

- ✓ A recursion is a process by which a function calls itself repeatedly until some specified condition has been fulfilled.
- ✓ It is also possible to pass array structure as function argument. There are two ways (1) simply pass structure as simple parameter values by passing the structure name (2) with the use of pointer.
- ✓ It is also possible to pass pointer as function argument. When we are passing address to the function at that time changes made in the function implementation is reflected back to the calling program.

2 – Vaishalinagar
Nr. Amrapali Under Bridge
Raiya Road
Rajkot – 360001
Ph.No–(0281)2440478,2472590

3 – Vaishalinagar
Nr. Amrapali Under Bridge
Raiya Road
Rajkot - 360001
Ph.No–(0281)2471645

UNIT -2 – PART -1 - SORTING & SEARCHING



- Bubble sorting
- Insertion sorting
- Quick sorting
- Bucket sorting
- Merge sorting
- Selection sorting
- Shell sorting
- Basic searching technique
- Sequential searching
- Binary searching

- The process of "looking up" a particular record in the data is called "searching".
- The process of ordering the records in a database is called "sorting".

Sorting

- Sorting is important!
- Things that would be much more difficult without sorting:
  - finding a phone number in the phone book
  - looking up a word in the dictionary
  - finding a book in the library
  - buying a cd/dvd
  - renting a video
  - buying groceries

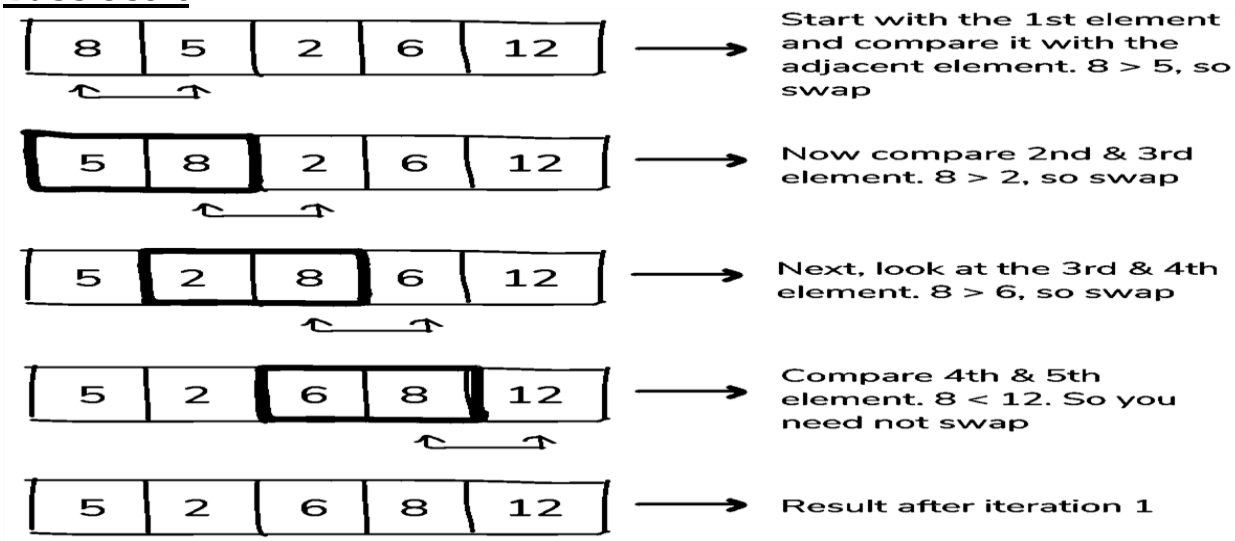
- The operation is the most common task performed by computers.
- Sorting is the process of arranging data information in some logical order.
- This logical order may be ascending or descending in case of numeric values.

- Various techniques are available to sort data depending on length of data, speed of sorting, number of swapping done during procedure of sorting etc.

Types of sorting techniques

1. Bubble sort
2. Insertion sort
3. Quick sort
4. Bucket sort
5. Merge sort
6. Selection sort
7. Shell sort

Bubble sort



- It is the simplest sorting algorithm techniques.
- In this technique, we continually compare two adjacent items (elements) from the list. If the first element is larger than the second one, then the position of the elements are interchanged (swap) otherwise not changed and then the after sorting is completed. This process is used frequently until no swaps are needed.

- It is also known as "comparison sort" because it continually compares two adjacent elements from the list.

Advantages & Disadvantages of Bubble sort

- One of the primary advantages of the bubble sort is that it is comparatively easy to write and learn.
- It is also comparatively easy to understand in terms of sorting algorithm.
- Unfortunately, the bubble sort is also a relatively slow algorithm, taking $O(n^2)$ to complete sorting and therefore, should not be used on large tables.

Algorithm for Bubble sort

- Let "a" be an array of n numbers. "temp" is a temporary variable for swapping the position of the numbers.

Step 1: Input n numbers for an array "a"
Step 2: Initialize i=0 and repeat through step 4 if(i<n)
Step 3: Initialize j=0 and repeat through step 4 if(j<n-1)
Step 4: if(a[j]>a[j+1])
temp=a[j];
a[j]=a[j+1];
a[j+1]=temp;
Step 5: Display the sorted numbers of array a
Step 6: Exit

Program for Bubble sort

```
#include<stdio.h>
#include<conio.h>
void main()
{
    int a[100],temp,i,j,n,changes,k;
    clrscr();
    printf("\nEnter array size:");
    scanf("%d",&n);
    for(i=0;i<n;i++)
```

```
    {
        printf("\nEnter array elements a[%d]:",i);
        scanf("%d",&a[i]);
    }
    for(i=0;i<n;i++)
    {
        changes=0;
        for(j=0;j<n-1;j++)
        {
                if(a[j]>a[j+1])
                {
                        temp=a[j];
                        a[j]=a[j+1];
                        a[j+1]=temp;
                        changes++;
                }
        }
        if(changes==0)
                break;
        printf("\nAfter pass %d elements are:",i+1);
        for(k=0;k<n;k++)
                printf("\t%d",a[k]);
        printf("\n");
    }
    printf("\nAfter sorting array elements are:");
    for(i=0;i<n;i++)
    {
        printf("\n%d",a[i]);
    }
    getch();
}
```

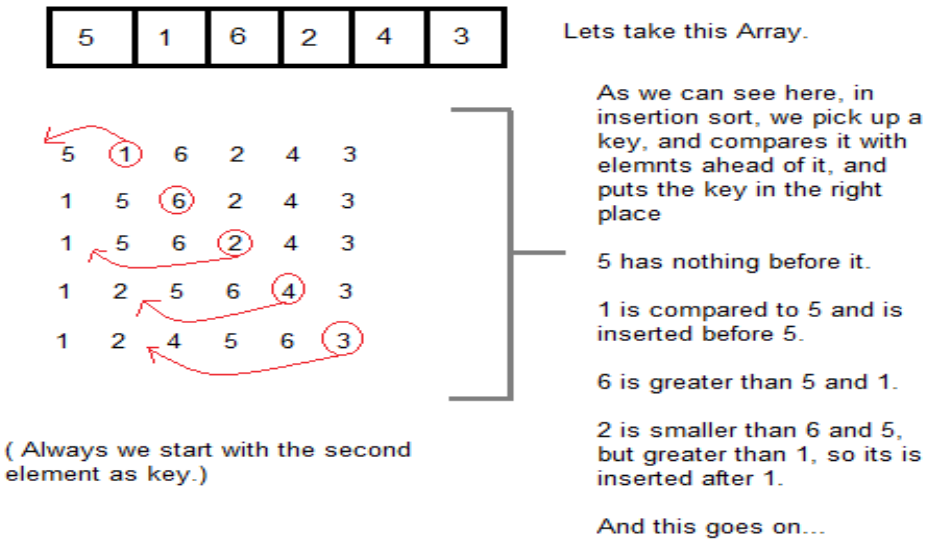
2 – Vaishalinagar
Nr. Amrapali Under Bridge
Raiya Road
Rajkot – 360001
Ph.No–(0281)2440478,2472590

3 – Vaishalinagar
Nr. Amrapali Under Bridge
Raiya Road
Rajkot - 360001
Ph.No–(0281)2471645

1 WORD QUESTION ANSWER

SR.NO.	QUESTION	ANSWER
1	WHAT IS AN EXTERNAL SORTING ALGORITHM?	ALGORITHM THAT USES TAPE OR DISK DURING THE SORT
2	WHAT IS AN INTERNAL SORTING ALGORITHM?	ALGORITHM THAT USES MAIN MEMORY DURING THE SORT
3	WHAT IS THE WORST CASE COMPLEXITY OF BUBBLE SORT?	O(N2)
4	WHAT IS THE AVERAGE CASE COMPLEXITY OF BUBBLE SORT?	O(N2)
5	THE GIVEN ARRAY IS ARR = {1, 2, 4, 3}. BUBBLE SORT IS USED TO SORT THE ARRAY ELEMENTS. HOW MANY ITERATIONS WILL BE DONE TO SORT THE ARRAY?	4
6	WHAT IS THE BEST CASE EFFICIENCY OF BUBBLE SORT IN THE IMPROVISED VERSION?	O(N)

Insertion sort



- It is very simple and efficient algorithms for the smallest lists.
- Its mechanism is very simple just take elements from the list one by one and insert them in their correct position into a new sorted list.
- The name inserting sorting means that sorting is occurred by inserting a particular element at proper position.

Algorithm for Insertion sort

- Let "a" be an array of n numbers. "temp" is a temporary variable for swapping the position of the numbers. "pos" is the control variable to hold the position of each pass.

Step 1: Input n numbers for an array "a"
Step 2: Initialize i=0 and repeat through step 4 if(i<n-1)
temp=a[i]
j=i-1
Step 3: Repeat the step 3 if(temp<a[j] and (j>=0))
a[j+1]=a[j]
j=j-1
Step 4: a[j]=temp
Step 5: Exit

2 – Vaishalinagar
Nr. Amrapali Under Bridge
Raiya Road
Rajkot – 360001
Ph.No–(0281)2440478,2472590

3 – Vaishalinagar
Nr. Amrapali Under Bridge
Raiya Road
Rajkot - 360001
Ph.No–(0281)2471645

Program for insertion sort

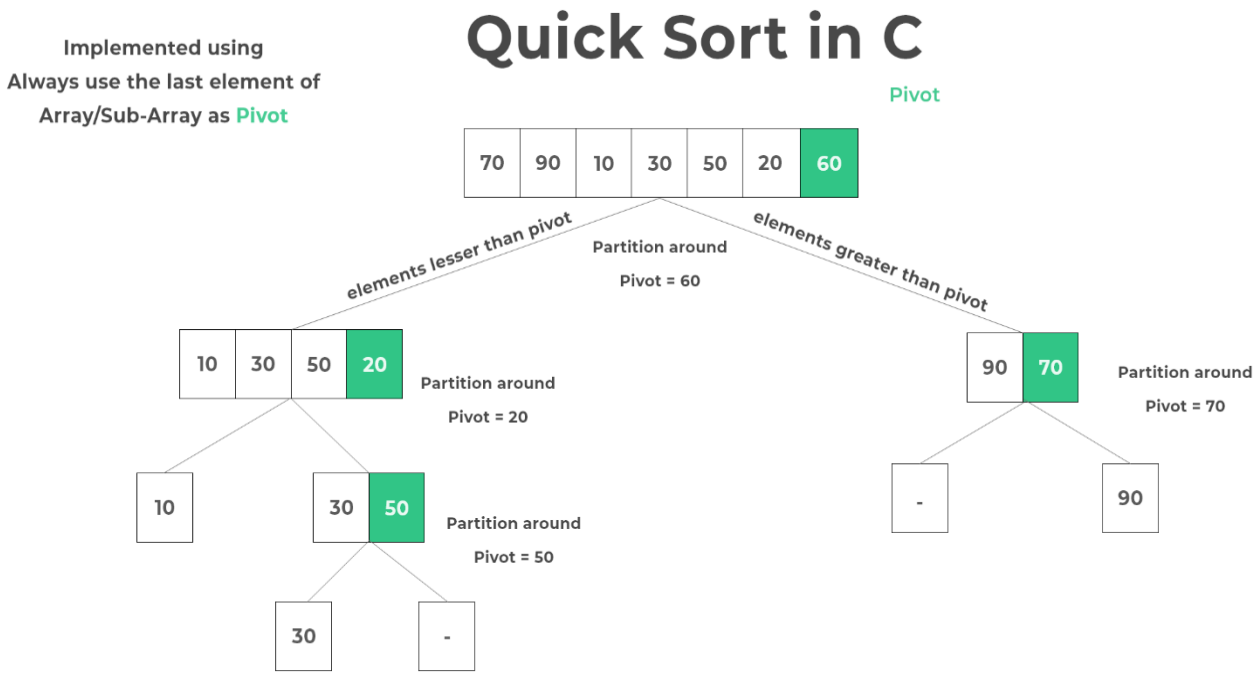
```
#include<stdio.h>
#include<conio.h>
void main()
{
    int a[100],n,i,j,k,tmp,changes;
    clrscr();
    printf("\nEnter array size:");
    scanf("%d",&n);
    for(i=0;i<n;i++)
    {
        printf("\nEnter array element a[%d]:",i);
        scanf("%d",&a[i]);
    }
    printf("\n\nUnsorted array");
    for(i=0;i<n;i++)
    {
        printf("\t%d",a[i]);
    }
    for(i=1;i<n;i++)
    {
        tmp=a[i];
        changes=0;
        for(j=i-1;j>=0;j--)
        {
            if(tmp<a[j])
            {
                a[j+1]=a[j];
                a[j]=tmp;
            }
            changes++;
        }
        if(changes==0)
            break;
    printf("\nPass %d,element inserted at proper place:%d",i,tmp);
        for(k=0;k<n;k++)
```

```
                printf("\t%d",a[k]);
            printf("\n");

        }
        printf("\n\nSorted array");
        for(i=0;i<n;i++)
        {
            printf("\t%d",a[i]);
        }
        getch();
}
```

1 WORD QUESTION ANSWER

SR.NO.	QUESTION	ANSWER
1	HOW MANY PASSES DOES AN INSERTION SORT ALGORITHM CONSIST OF?	N-1
2	WHICH ALGORITHM IMPLEMENTATIONS IS SIMILAR TO THAT OF AN INSERTION SORT?	BINARY HEAP
3	WHAT IS THE AVERAGE CASE RUNNING TIME OF AN INSERTION SORT ALGORITHM?	O(N2)
4	ANY ALGORITHM THAT SORTS BY EXCHANGING ADJACENT ELEMENTS REQUIRE O(N2) ON AVERAGE.(TRUE OR FALSE)	TRUE
5	WHAT IS THE RUNNING TIME OF AN INSERTION SORT ALGORITHM IF THE INPUT IS PRE-SORTED?	O(N)
6	WHAT WILL BE THE NUMBER OF PASSES TO SORT THE ELEMENTS USING INSERTION SORT? 14, 12,16, 6, 3, 10	5
7	WHICH SORTING ALGORITHM IS BEST SUITED IF THE ELEMENTS ARE ALREADY SORTED?	INSERTION SORT
8	IN C, WHAT ARE THE BASIC LOOPS REQUIRED TO PERFORM AN INSERTION SORT?	FOR AND WHILE

Quick sort



- It is widely used sorting techniques which uses divide and conquer (also known as partition exchange sort)mechanism.
- The quick sort algorithm works by portioning the array to be sorted. And each partition is internally sorted recursively.
- In the quick sort mechanism, first of all we have to select middle element form the list and is known as pivot element.
- After that, the sort is divides the list into two sub lists.
- First list contains the elements that are less than the pivot elements and a second list contains elements that are greater than pivot elements.

Algorithm for Quick sort

- Let "a" be an array of n numbers. "temp" is a temporary variable for swapping the position of the numbers.

Step 1: initialize low=first, high=last
pivot=(low+high)/2

Step 2 repeat this step till low,=high

```
while(a[low]<pivot)
        low++;
while(a[high]>pivot)
        high--;
if(low<=high)
        temp=a[low]
        a[low]=a[high]
        a[high]=temp
```

Step 3: if(first<high)

```
        quicksort(a,first,high)
```

Step 4: if(low<last)

```
        quicksort(a,low,last)
```

Step 5: Exit

Program for quick sort

```
#include<stdio.h>
#include<conio.h>
void quicksort(int [],int,int);//function declaration
void main()
{
    int a[100],n,i;
    clrscr();
    printf("\nenter array size:");
    scanf("%d",&n);
        for(i=0;i<n;i++)
        {
            printf("\n enter array elements a[%d]",i);
            scanf("%d",&a[i]);
        }
        quicksort(a,0,n-1);//function calling
        for(i=0;i<n;i++)
        {
            printf("\n sorted array elements %d",a[i]);
        }
        getch();
    }
```

2 – Vaishalinagar
Nr. Amrapali Under Bridge
Raiya Road
Rajkot – 360001
Ph.No–(0281)2440478,2472590

3 – Vaishalinagar
Nr. Amrapali Under Bridge
Raiya Road
Rajkot - 360001
Ph.No–(0281)2471645

```
void quicksort(int a[], int first, int last)//function definition
{
        int low, high, temp, pivot, i;
        low=first;
        high=last;
        pivot=(low+high)/2;
        while(low<=high)
        {
                while(a[low]<a[pivot])
                        low++;
                while(a[high]>a[pivot])
                        high--;
                if(low<=high)
                {
                        temp=a[low];
                        a[low]=a[high];
                        a[high]=temp;
                        low++;
                        high--;
                }
}
if(first<high)
        quicksort(a,first,high);
if(low<last)
        quicksort(a,low,last);
}
```

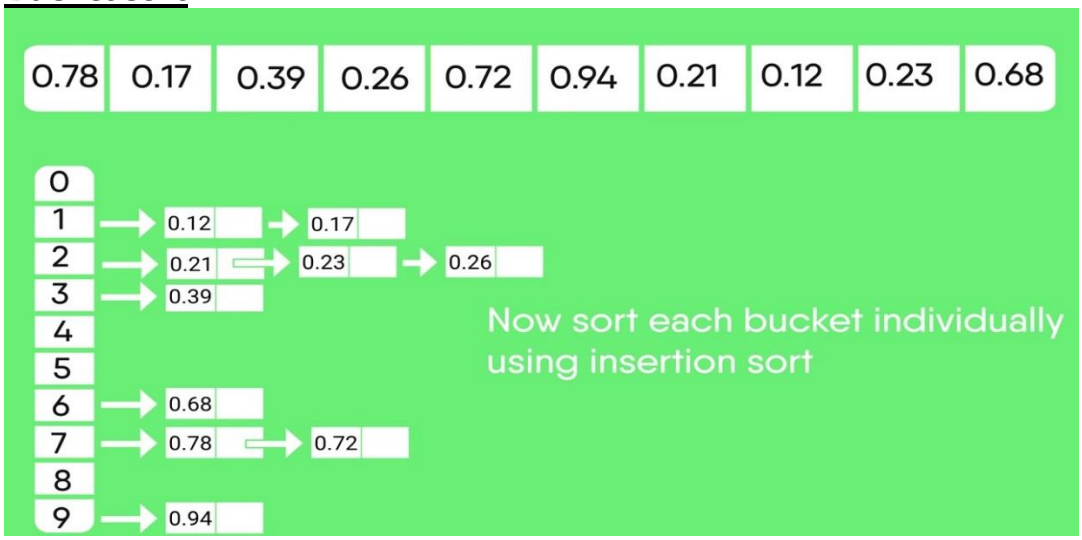
2 – Vaishalinagar
Nr. Amrapali Under Bridge
Raiya Road
Rajkot – 360001
Ph.No–(0281)2440478,2472590

3 – Vaishalinagar
Nr. Amrapali Under Bridge
Raiya Road
Rajkot - 360001
Ph.No–(0281)2471645

1 WORD QUESTION ANSWER

SR.NO.	QUESTION	ANSWER
1	WHICH SORTING ALGORITHMS IS THE FASTEST?	QUICK SORT
2	QUICK SORT FOLLOWS DIVIDE-AND-CONQUER STRATEGY.(TRUE OR FALSE)	TRUE
3	WHAT IS THE WORST CASE TIME COMPLEXITY OF A QUICK SORT ALGORITHM?	O(N2)
4	FIND THE PIVOT ELEMENT FROM THE GIVEN INPUT USING MEDIAN-OF-THREE PARTITIONING METHOD. 8, 1, 4, 9, 6, 3, 5, 2, 7, 0.	6
5	WHICH IS THE SAFEST METHOD TO CHOOSE A PIVOT ELEMENT?	CHOOSING A RANDOM ELEMENT AS PIVOT
6	WHAT IS THE AVERAGE RUNNING TIME OF A QUICK SORT ALGORITHM?	O(N LOG N)
7	WHICH SORTING ALGORITHMS IS USED ALONG WITH QUICK SORT TO SORT THE SUB ARRAYS?	INSERTION SORT
8	QUICK SORT USES JOIN OPERATION RATHER THAN MERGE OPERATION.(TRUE OR FALSE)	TRUE

Bucket sort



- Bucket sort is a sorting method that can be used to sort a list of numbers by its base.
- If we want to sort list of English words where base is 26, then 26 buckets is used to sort the words.
- To sort array of decimal numbers where base is 10 we need 10 buckets and it can be numbered as 0,1,2,3,4,5,6,7,8,9.
- On the basis of the largest number's digit that many passes are required.
- Mechanism includes comparison of the first position of digit with the digit of bucket and place it. (recursive)

Algorithm for Bucket sort

Step 1: Input n number of elements in array a.
Step 2: Find out the largest element and the digit of the largest element
Step 3: Initialize i=1 and repeat steps 4 and 5 until (i<digitcnt)
Step 4: Initialize the buckets j=0 and repeat the steps (a) until (j<n)
(a)Compare i the position of each element of the array with the bucket number and place it int the corresponding bucket.
Step 5: Read the elements of the bucket from 0^{th} bucket to 9^{th} bucket and from first position to higher one to generate new array a.
Step 6: Display the sorted array a
Step 7: Exit

Program for Bucket sort

```
#include<stdio.h>
#include<conio.h>
void main()
{
    int a[100],n,lrg,i,j,k,p,digcnt,divsr,r;
    int bucktcnt[10],buckt[10][10];
    clrscr();
    printf("\nEnter array size:");
    scanf("%d",&n);
    for(i=0;i<n;i++)
    {
        printf("\nEnter array elements a[%d]:",i);
        scanf("%d",&a[i]);
    }
    i=0;
    lrg=a[i];
    while(i<n)/* find the largest element in array */
    {
        if(a[i]>lrg)
            lrg=a[i];
            i++;
    }
    /*count the no of digit in the largest no*/
    digcnt=0;
    while(lrg>0)
    {
        digcnt++;
        lrg=lrg/10;
    }
    i=1;
    divsr=1;
    while(i<=digcnt)
    {
        j=0;
        while(j<10)
```

```
        {
            bucktcnt[j]=0;
            j++;
        }
        j=0;
        while(j<n)
        {
            r=(a[j]/divsr)%10;
            buckt[r][bucktcnt[r]]=a[j];
            bucktcnt[r]++;
            j++;
        }
        /*collect all elements in order*/
        j=0;
        p=0;
        while(j<10)
        {
            k=0;
            while(k<bucktcnt[j])
            {
                a[p]=buckt[j][k];
                p++;
                k++;
            }
            j++;
        }
        i++;
        divsr=divsr*10;
    }
    printf("\nSorted Elements:");
    for(i=0;i<n;i++)
    {
        printf("\t%d",a[i]);
    }
    getch();
}
```

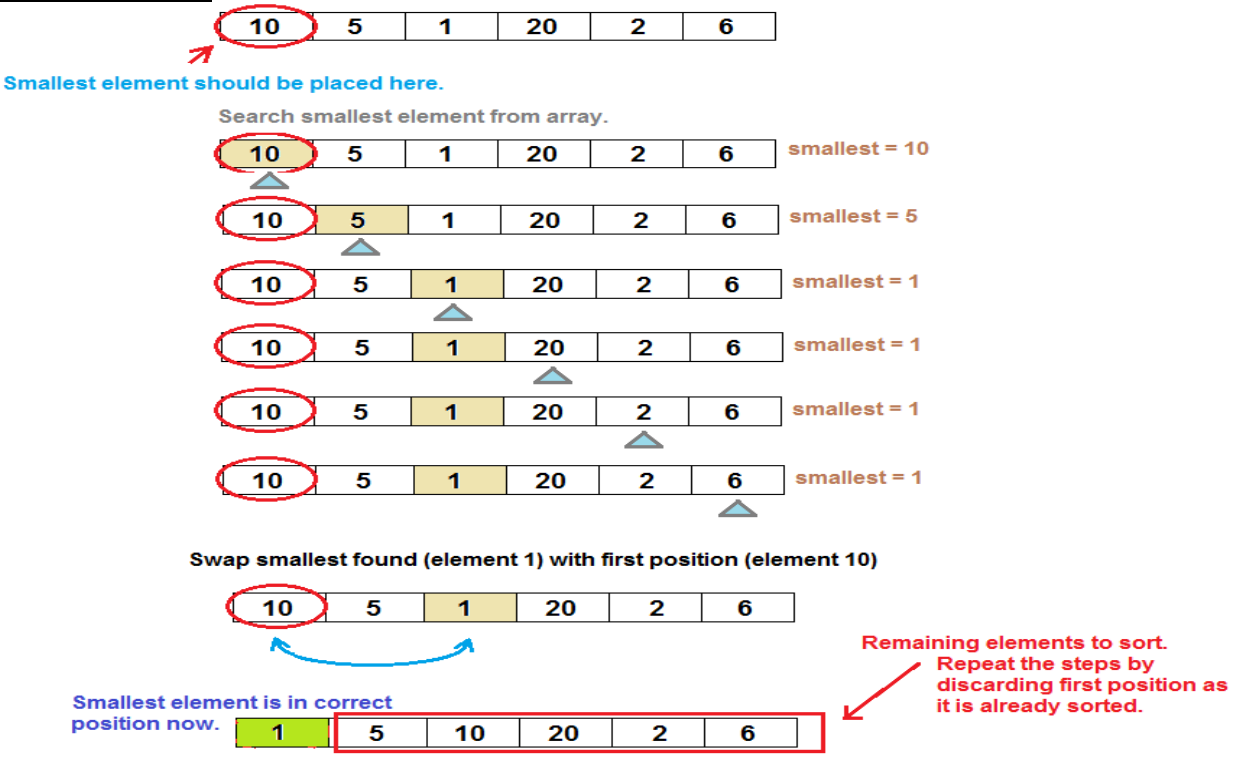
1 WORD QUESTION ANSWER

SR.NO.	QUESTION	ANSWER
1	HOW MANY COMPARISONS WILL BE MADE TO SORT THE ARRAY ARR={1, 5, 3, 8, 2} USING BUCKET SORT?	0
2	WHAT IS THE ALTERNATE NAME OF BUCKET SORT?	BIN SORT
3	WHICH NON-COMPARISON SORT CAN ALSO BE CONSIDERED AS A COMPARISON BASED SORT?	BUCKET SORT
4	WHAT IS THE WORST CASE TIME COMPLEXITY OF BUCKET SORT (K = NUMBER OF BUCKETS)?	O(N2)
5	WHAT IS THE BEST TIME COMPLEXITY OF BUCKET SORT (K= NUMBER OF BUCKETS)?	O(N + K)
6	BUCKET SORT IS AN IN PLACE SORTING ALGORITHM. (TRUE OR FALSE)	FALSE
7	WHAT IS THE WORST SPACE COMPLEXITY OF BUCKET SORT (K = NUMBER OF BUCKETS)?	O(N.K)

2 – Vaishalinagar
Nr. Amrapali Under Bridge
Raiya Road
Rajkot – 360001
Ph.No–(0281)2440478,2472590

3 – Vaishalinagar
Nr. Amrapali Under Bridge
Raiya Road
Rajkot - 360001
Ph.No–(0281)2471645

Selection sort



- Selection Sort algorithm is a simple sorting algorithm which specially is an in-place comparison sorts. It is a technique to arrange the data in proper order.
- This type of sorting is called “Selection sort” because it works by repeatedly selecting smallest element.
- If we want to sort array in increasing order(i.e smallest element at the beginning of the array and the largest element at the end.) then find the minimum element and place it in the first position (recursion).

Algorithm for Selection sort

Step 1: Input n number of elements in array a.
Step 2: Initialize i TO 0 (i=0)
Step 3: Repeat through step 8 while i<n-1(i=0,1,2….,n-1)
Step 4: Initialize min TO i (min=i)
Step 5: Initialize j=i+1
Step 6: Repeat through j=j+1 while j<n(j=i+1,i+2,….)
If(a[j]<a[min])
min = j (min=j)
Step 7: if(min!=i)
temp = a[i]
a[i] = a[min]
a[min] = temp
Step 8: i=i+1
Step 9: Exit

Program for Selection sort

```
#include<stdio.h>
#include<conio.h>
void main()
{
    int a[100],n,i,j,temp,min;
    clrscr();
    printf("\nEnter array size:");
    scanf("%d",&n);
    for(i=0;i<n;i++)
    {
        printf("\nEnter array elements a[%d]:",i);
        scanf("%d",&a[i]);
    }
    for(i=0;i<n-1;i++)
    {
        min=i;
        for(j=i+1;j<n;j++)
        {
            if(a[j]<a[min])
```

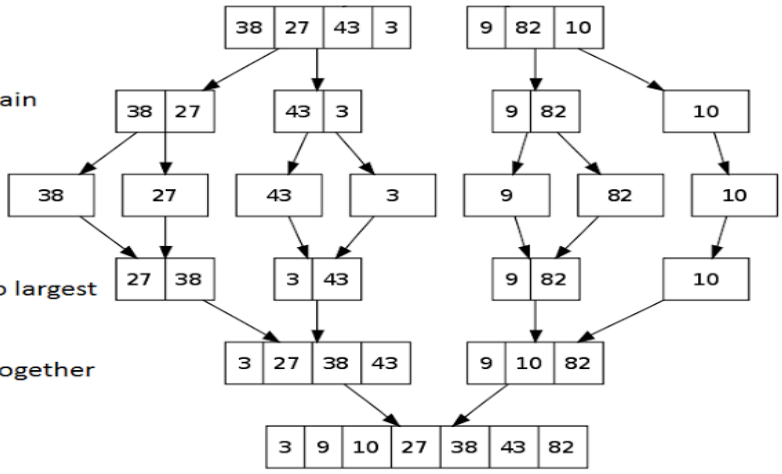
```
                    min=j;
                }
                if(min!=i)
                {
                    temp=a[i];
                    a[i]=a[min];
                    a[min]=temp;
                }

            }
            printf("\n\nSorted array:");
            for(i=0;i<n;i++)
            {
                printf("\t%d",a[i]);
            }
            getch();
}
```

1 WORD QUESTION ANSWER

SR.NO.	QUESTION	ANSWER
1	WHAT IS THE WORST CASE COMPLEXITY OF SELECTION SORT?	O(N2)
2	WHAT IS THE BEST CASE COMPLEXITY OF SELECTION SORT?	O(N2)
3	THE GIVEN ARRAY IS ARR = {3,4,5,2,1}. THE NUMBER OF ITERATIONS IN BUBBLE SORT AND SELECTION SORT RESPECTIVELY ARE,	5 AND 4
4	WHAT IS THE AVERAGE CASE COMPLEXITY OF SELECTION SORT?	O(N2)

Merge sort

1. Divide the array into two parts
2. Divide the array into two parts again
3. Break each element into single parts
4. Sort the elements from smallest to largest
5. Merge the divided sorted arrays together
6. The array has been sorted



- It is widely used sorting technique which uses divide & conquer mechanism.
- In this type of method the problems which can be broken into smaller problems, solve the smallest problems and then merge them to get the final answer. It means that we continue dividing till only one element is left.
- In this techniques, our large list is broken down into smaller lists and then after merge together.
- It is the process of combining two or more sorted array into third sorted array.
- When we are using these techniques different lists available are:
    1. First divide the list into half
    2. Then sort the left half
    3. Then sort the right half
    4. Then merge the two sorted halves into one sorted list.

Algorithm for Merge sort

Mergesort(n,list1,m,list2);

n:- represent number of elements in first list
list1:- represent list of elements (first list)
m:- represent number of elements in second list

list2: represent list of elements (second list)

Step 1: first, the array is divided into two parts. i.e. mid is determined between low index and high index.

midsort(low, high);
low: low index
high: high index
mid=(low+high)/2

step 2: then first part(from low to mid) and second part (mid+1 to high) are sorted by calling the function midsort

midsort(low,mid);
midsort(mid+1,high)

step 3: then, above two sorted parts are merged by calling mergesort function

step 4: initialize i low, j mid+1, k high(low)

step 5: repeat this step till i<=mid and j<=high

```
if(a[i]>=a[j])
        temp[k]=a[j];
        k++;
        j++;
else
        temp[k]=a[i];
        k++;
        i++
```

step 6: repeat this step till i<=mid.

```
temp[k]=a[i];
k++;
i++;
```

step 7: repeat this step till j<=high

```
temp[k]=a[j];
k++;
j++;
```

step 8: repeat this step till i<=high

```
for(i=low;i<=high;i++)          copying final array to a
        a[i]=temp[i]
```

step 9: STOP

2 – Vaishalinagar
Nr. Amrapali Under Bridge
Raiya Road
Rajkot – 360001
Ph.No–(0281)2440478,2472590

3 – Vaishalinagar
Nr. Amrapali Under Bridge
Raiya Road
Rajkot - 360001
Ph.No–(0281)2471645

Program for Merge sort

```
#include<stdio.h>
#include<conio.h>
int a[100];
void m_sort(int,int);
void merge_sort(int,int,int);
void main()
{
    int n,i;
    clrscr();
    printf("\nEnter array size:");
    scanf("%d",&n);
    for(i=0;i<n;i++)
    {
        printf("\nEnter array elements a[%d]:",i);
        scanf("%d",&a[i]);
    }
    printf("\n\nUnsorted array");
    for(i=0;i<n;i++)
    {
        printf("\t%d",a[i]);
    }
    m_sort(0,n-1);
    printf("\n\nSorted array");
    for(i=0;i<n;i++)
    {
        printf("\t%d",a[i]);
    }
    getch();
}
void m_sort(int low,int high)
{
    int mid;
    if(low!=high)
    {
        mid=(low+high)/2;
```

2 – Vaishalinagar
Nr. Amrapali Under Bridge
Raiya Road
Rajkot – 360001
Ph.No–(0281)2440478,2472590

3 – Vaishalinagar
Nr. Amrapali Under Bridge
Raiya Road
Rajkot - 360001
Ph.No–(0281)2471645

```
            m_sort(low,mid);
            m_sort(mid+1,high);
            merge_sort(low,mid,high);
        }
}
void merge_sort(int low,int mid,int high)
{
        int i,j,k,temp[100];
        i=low;
        j=mid+1;
        k=low;
        do
        {
            if(a[i]>=a[j])
                    temp[k++]=a[j++];
            else
                    temp[k++]=a[i++];
        }while((i<=mid)&&(j<=high));
        while(i<=mid)
            temp[k++]=a[i++];
        while(j<=high)
            temp[k++]=a[j++];
        for(i=low;i<=high;i++)
        {
            a[i]=temp[i];
        }
}
```

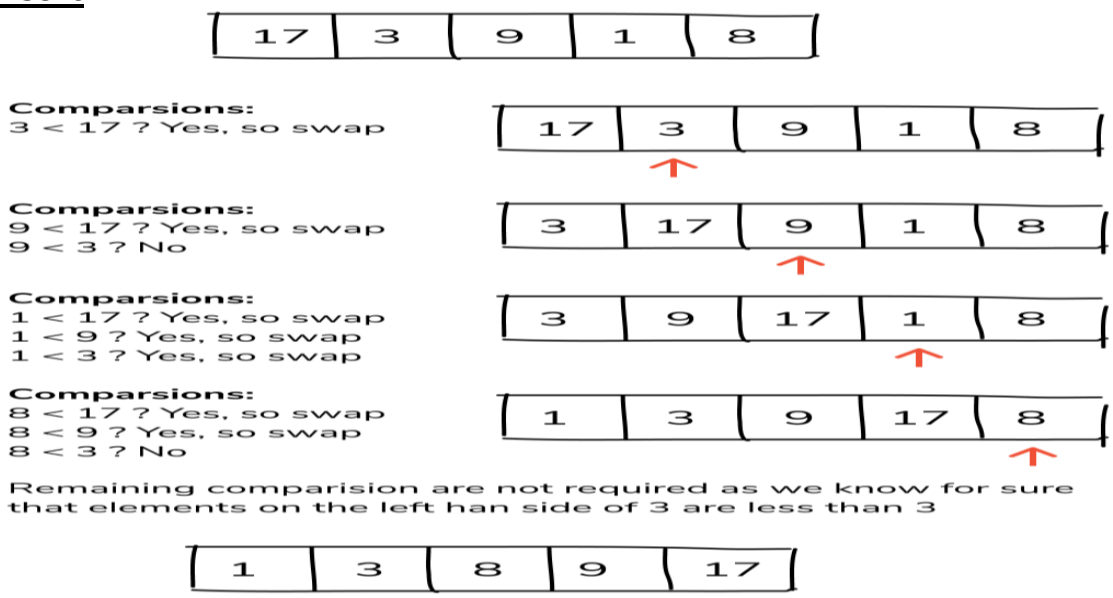
2 – Vaishalinagar
Nr. Amrapali Under Bridge
Raiya Road
Rajkot – 360001
Ph.No–(0281)2440478,2472590

3 – Vaishalinagar
Nr. Amrapali Under Bridge
Raiya Road
Rajkot - 360001
Ph.No–(0281)2471645

1 WORD QUESTION ANSWER

SR.NO.	QUESTION	ANSWER
1	MERGE SORT USES WHICH TECHNIQUE TO IMPLEMENT SORTING?	DIVIDE AND CONQUER
2	WHAT IS THE AVERAGE CASE TIME COMPLEXITY OF MERGE SORT?	O(N LOG N)
3	MERGE SORT CAN BE IMPLEMENTED USING O(1) AUXILIARY SPACE.(TRUE OR FALSE)	TRUE
4	WHAT IS THE WORST CASE TIME COMPLEXITY OF MERGE SORT?	O(N LOG N)
5	WHICH METHOD IS USED FOR SORTING IN MERGE SORT?	MERGING
6	WHAT WILL BE THE BEST CASE TIME COMPLEXITY OF MERGE SORT?	O(N LOG N)
7	MERGE SORT IS PREFERRED FOR ARRAYS OVER LINKED LISTS.(TRUE OR FALSE)	FALSE

Shell sort



- Shell sort is introduced to improve the efficiency of simple insertion sort.
- Shell sort is also called **diminishing increment sort**

Algorithm for Shell sort

Step 1: Input n number of elements in array a.
Step 2: Initialize i= 0 and repeat through step 6 if(i<x)
Step 3: span=incr[i]
Step 4: Initialize j=span and repeat through step 6 if (j<n)
temp=a[j]
Step 5: Initialize k=j-span and repeat through step 5 if(k>=0) and (temp<a[k])
a[k+span]=a[k]
Step 6: a[k+span]=temp
Step 7: Exit

Program for Shell sort

```
#include<stdio.h>
#include<conio.h>
void main()
{
    int a[100],n,i,num,k,j;
```

2 – Vaishalinagar
Nr. Amrapali Under Bridge
Raiya Road
Rajkot – 360001
Ph.No–(0281)2440478,2472590

3 – Vaishalinagar
Nr. Amrapali Under Bridge
Raiya Road
Rajkot - 360001
Ph.No–(0281)2471645

```
    clrscr();
    printf("\nEnter array size:");
    scanf("%d",&n);
    for(i=0;i<n;i++)
    {
        printf("\nEnter array elements a[%d]:",i);
        scanf("%d",&a[i]);
    }
    printf("\nEnter maximum number (odd value):");
    scanf("%d",&num);
    while(num>=1)
    {
        for(j=num;j<n;j++)
        {
                k=a[j];
                for(i=j-num;i>=0 && k<a[i];i=i-num)
                        a[i+num]=a[i];
                a[i+num]=k;
        }
        printf("\nIncrement=%d\n",num);
        for(i=0;i<n;i++)
                printf("%d\t",a[i]);
                printf("\n");
                num=num-2;
    }
    printf("\nSorted array");
    for(i=0;i<n;i++)
    {
        printf("%d\t",a[i]);
    }
    getch();
}
```

2 – Vaishalinagar
Nr. Amrapali Under Bridge
Raiya Road
Rajkot – 360001
Ph.No–(0281)2440478,2472590

3 – Vaishalinagar
Nr. Amrapali Under Bridge
Raiya Road
Rajkot - 360001
Ph.No–(0281)2471645

1 WORD QUESTION ANSWER

SR.NO.	QUESTION	ANSWER
1	WHAT IS THE OTHER NAME FOR A SHELL SORT ALGORITHM?	DIMINISHING INCREMENT SORT
2	THE WORST CASE RUNNING TIME OF SHELL SORT, USING SHELL'S INCREMENTS IS?	O(N2)
3	WHO INVENTED THE SHELL SORT ALGORITHM?	DONALD SHELL
4	SHELL SORT ALGORITHM IS AN EXAMPLE OF?	INTERNAL SORTING

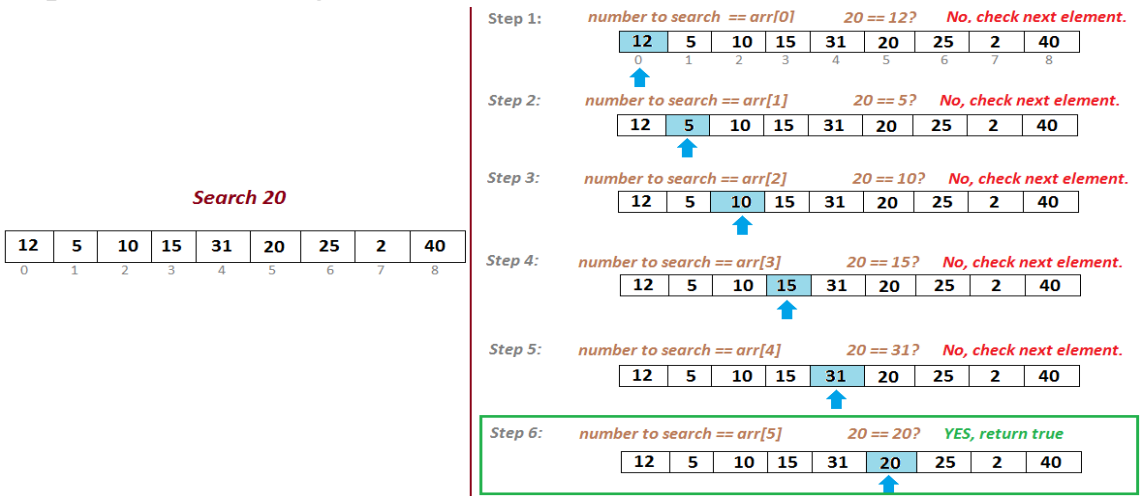
2 – Vaishalinagar
Nr. Amrapali Under Bridge
Raiya Road
Rajkot – 360001
Ph.No–(0281)2440478,2472590

3 – Vaishalinagar
Nr. Amrapali Under Bridge
Raiya Road
Rajkot - 360001
Ph.No–(0281)2471645

SEARCHING METHOD

- Traversing an array to locate a particular item
- It requires the user to specify the target item
- If the target item is found, its index is returned
- If the target item is NOT found, -1 is returned
- Two searching algorithms
  - Linear Search (works with any array)
  - Binary Search (works if the searched array is sorted)

1. Sequential searching



- To search (locate, find) an element from the unsorted array list we are using this simplest technique.
- It simply traverses from top to bottom in the array and searches for the key value from the list and displays output as well. It is called sequential searching method.

Algorithm for sequential searching

a ➡ represents array
n ➡ represent number of elements in array
ele ➡ represents element to be searched

Step 1: Initialize flag=0
Step 2: Initialize i=0 and repeat till if(i<n)

Get array elements (&a[i])

Step 3: Repeat step 4 for i=0,1,2….n-1

Step 4:if(a[i]==ele)

Output “successful searching”

flag=1

Step 5: if flag==0

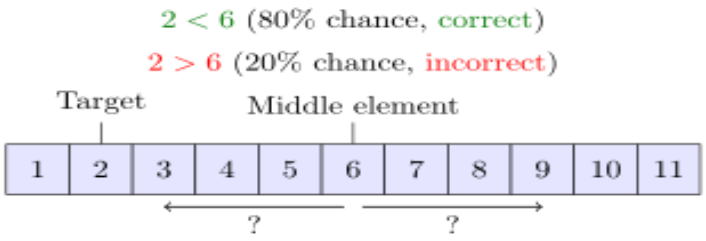
Output “unsuccessful search”

Step 6: STOP

Program for sequential searching

```
#include<stdio.h>
#include<conio.h>
void main()
{
    int a[100],n,ele,i,flag=0;
    clrscr();
    printf("\nEnter array size:");
    scanf("%d",&n);
    for(i=0;i<n;i++)
    {
        printf("\nEnter array elements a[%d]:",i);
        scanf("%d",&a[i]);
    }
    printf("\nEnter the element you want to search:");
    scanf("%d",&ele);
    for(i=0;i<n;i++)
    {
        if(a[i]==ele)
        {
            printf("\nYour element %d is at position %d",ele,i);
            flag=1;
        }
    }
    if(flag==0)
        printf("\nNo such element avaialable in array");
    getch();
}
```

Binary searching



Goal: Search for the index of the target value so that there is a probability of p that the index is correct. p is specified before the search.

- The main characteristics of binary search is that it works only on sorted array lists and so it becomes easy to find any information very fast.

Algorithm for binary searching

a ➡ represents array
n ➡ represent number of elements in array
ele ➡ represents element to be searched

Step 1: Initialize flag=0,start:
Step 2: Initialize i=0 and repeat through step 3 till if(i<n)
Get array elements (&a[i])
Step 3:if(i>0)
if(a[i-1]>a[i])
break;
goto start; //label
Step 4: Repeat step 4 for i=0,1,2….n-1
Step 5:if(a[i]==ele)
Output "successful searching"
flag=1
Step 6: if flag==0
Output "unsuccessful search"
Step 7: STOP

Program for binary searching

```
#include<stdio.h>
#include<conio.h>
void main()
{
```

```
    int a[100],n,i,ele,flag=0;
    clrscr();
    printf("\nEnter array size:");
    scanf("%d",&n);
    for(i=0;i<n;i++)
    {
            printf("\nEnter array element a[%d]:",i);
            scanf("%d",&a[i]);
            if(i>0)
            {
                    if(a[i-1]>a[i])
                    {
                            printf("\nnumber should be greater than
previous value");
                            getch();
                            break;
                    } }}
    printf("\nEnter element to be searched");
    scanf("%d",&ele);
    for(i=0;i<n;i++)
    {
            if(a[i]==ele)
            {
                    printf("\nYour element %d is at position %d",ele,i);
                    flag==1;
            } }
    if(flag==0)
            printf("\nElement not found");
    getch();
}
```

1 WORD QUESTION ANSWER

SR.NO.	QUESTION	ANSWER
1	WHAT IS THE BEST CASE FOR LINEAR SEARCH?	O(1)
2	WHAT IS THE WORST CASE FOR LINEAR SEARCH?	O(N)
3	WHAT IS THE BEST CASE AND WORST CASE COMPLEXITY OF ORDERED LINEAR SEARCH?	O(1), O(N)
4	WHAT IS THE WORST CASE RUNTIME OF LINEAR SEARCH(RECURSIVE) ALGORITHM?	O(N)
5	LINEAR SEARCH(RECURSIVE) ALGORITHM USED IN _____	WHEN THE SIZE OF THE DATASET IS LOW
6	WHAT IS THE WORST CASE COMPLEXITY OF BINARY SEARCH USING RECURSION?	O(LOGN)
7	WHAT IS THE AVERAGE CASE TIME COMPLEXITY OF BINARY SEARCH USING RECURSION?	O(LOGN)
8	WHAT IS THE TIME COMPLEXITY OF BINARY SEARCH WITH ITERATION?	O(LOGN)

SUMMARY

- ✓ The process of arranging the data or information in some logical order is known as **sorting**
- ✓ The logical order may be ascending or descending based on our requirement.
- ✓ There are mainly two types of sorting available are.
  - Internal sorting
  - External sorting
- ✓ There are mainly five sorting techniques

- Bubble sort
- Insertion sort
- Quick sort
- Bucket sort
- Merge sort
- Selection sort
- Shell sort

✓ The process of "looking up" specific records from the database is known as **searching**

✓ There are mainly two types:

- Binary
- Sequential

2 – Vaishalinagar
Nr. Amrapali Under Bridge
Raiya Road
Rajkot – 360001
Ph.No–(0281)2440478,2472590

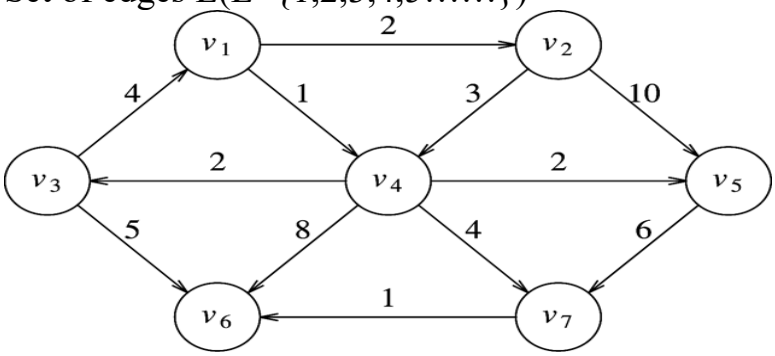
3 – Vaishalinagar
Nr. Amrapali Under Bridge
Raiya Road
Rajkot - 360001
Ph.No–(0281)2471645

UNIT - 2 PART – 2 - GRAPH

- Introduction
- Adjacency matrix and adjacency lists
- Graph traversal
  - Depth first search (dfs) Implementation
  - Breadth first search (bfs) Implementation
- Shortest path problem
- Minimal spanning tree

1. Introduction

- A graph is a general tree with no parent-child relationship.
- Graphs have many applications in computer science and other fields of science such as mapping, transportation, geography and so on. Graphs are used in various types of modeling. For example graphs can be used to represent connecting roads between cities.
- In general graphs represent a relatively less restrictive relationship between the data items. It is a way of representing relationships that exists between pairs of objects.
- A graph consist of
  - **i.** Set of vertices V(V={v1,v2,v3……})
  - **ii.** Set of edges E(E={1,2,3,4,5……})



2. Adjacency Matrix and Adjacent Lists

- Graph is a mathematical structure and finds its application in many areas, where the problem is to be solved by computers.

- The problem related to graph G must be represented in computer memory using any suitable data structure to solve the same.
- There are two standard ways of maintaining a graph G in the memory of computer.
  - **i.** Sequential representation of a graph using adjacent(i.e. Adjacency matrix representation)
  - **ii.** Linked representation of graph using linked list(i.e. Adjacency list representation)

1 WORD QUESTION ANSWER

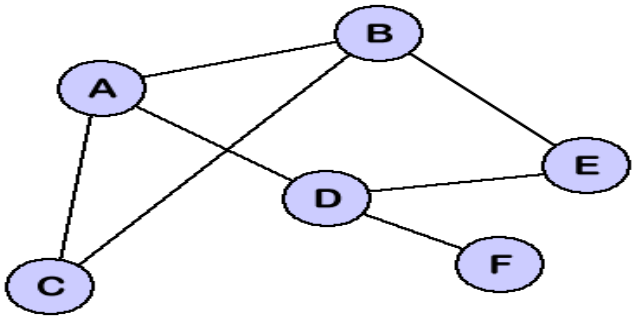
SR.NO.	QUESTION	ANSWER
1	IN GRAPH EVERY PATH IS TRAIL PATH? (TRUE OR FALSE)	**TRUE**
2	WHAT IS THE NUMBER OF EDGES PRESENT IN A COMPLETE GRAPH HAVING N VERTICES?	(N*(N-1))/2
3	IN A SIMPLE GRAPH, THE NUMBER OF EDGES IS EQUAL TO TWICE THE SUM OF THE DEGREES OF THE VERTICES.	FALSE
4	A GRAPH MAY CONTAIN MANY EDGES AND NO VERTICES.(TRUE OR FALSE)	TRUE
5	THE NUMBER OF ELEMENTS IN THE ADJACENCY MATRIX OF A GRAPH HAVING 7 VERTICES IS _____	49
6	ADJACENCY MATRIX OF ALL GRAPHS ARE SYMMETRIC.(TRUE OR FALSE)	FALSE

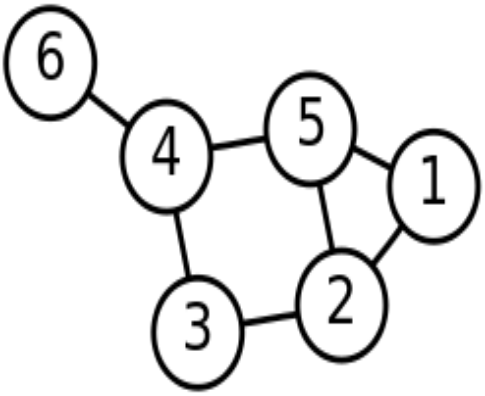
3. Graph Traversal

- A graph can be traversed using two methods as below:

a. Depth First Search(DFS)-Implementation

- When a graph is traversed by visiting the nodes in the forward direction as long as possible the traversal is called depth-first traversal.
- The algorithm repeatedly searches deeper by visiting unvisited vertices and whenever an unvisited vertex is not found, it backtracks to previous vertex to find out whether there are still unvisited vertices.



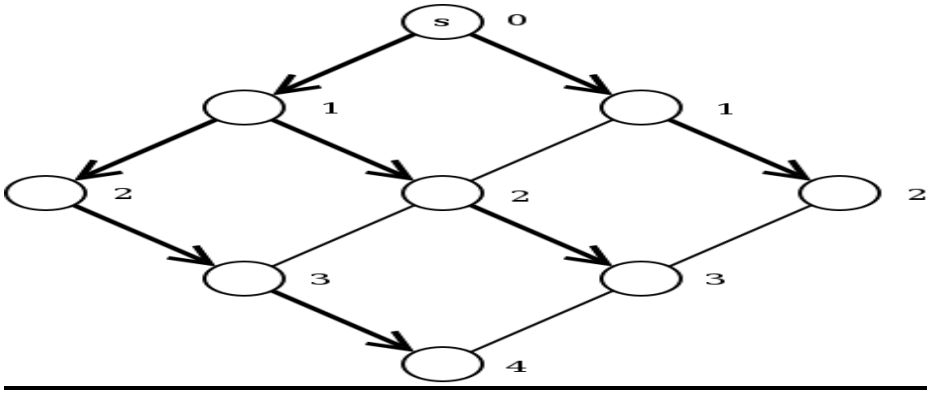


b. Breadth First Search(BFS)-Implementation

- When a graph is traversed by visiting all the adjacent nodes/vertices of node/vertex first, the traversal is called breadth first traversal
- When breath first is applied the vertices of the graph are divided into two categories.
- The vertices which are visited as part of search and those vertices which are not visited as part of the search.

2 – Vaishalinagar
Nr. Amrapali Under Bridge
Raiya Road
Rajkot – 360001
Ph.No–(0281)2440478,2472590

3 – Vaishalinagar
Nr. Amrapali Under Bridge
Raiya Road
Rajkot - 360001
Ph.No–(0281)2471645



1 WORD QUESTION ANSWER

SR.NO.	QUESTION	ANSWER
1	BREADTH FIRST SEARCH IS EQUIVALENT TO WHICH OF THE TRAVERSAL IN THE BINARY TREES?	LEVEL-ORDER TRAVERSAL
2	TIME COMPLEXITY OF BREADTH FIRST SEARCH IS? (V – NUMBER OF VERTICES, E – NUMBER OF EDGES)	O(V + E)
3	THE DATA STRUCTURE USED IN STANDARD IMPLEMENTATION OF BREADTH FIRST SEARCH IS?	QUEUE
4	THE BREADTH FIRST SEARCH TRAVERSAL OF A GRAPH WILL RESULT INTO?	TREE
5	WHEN THE BREADTH FIRST SEARCH OF A GRAPH IS UNIQUE?	WHEN THE GRAPH IS A LINKED LIST
6	THE DATA STRUCTURE USED IN STANDARD IMPLEMENTATION OF DFS IS?	STACK
7	WHAT WILL BE THE TIME COMPLEXITY OF THE ITERATIVE DEPTH FIRST TRAVERSAL CODE(V=NO. OF VERTICES E=NO.OF EDGES)?	O(V+E)

8	WHAT IS THE SPACE COMPLEXITY OF STANDARD DFS(V: NO. OF VERTICES E: NO. OF EDGES)?	O(V)
9	DEPTH FIRST SEARCH IS EQUIVALENT TO WHICH OF THE TRAVERSAL IN THE BINARY TREES?	PRE-ORDER TRAVERSAL
10	THE DEPTH FIRST SEARCH TRAVERSAL OF A GRAPH WILL RESULT INTO?	TREE

4. Shortest Path Problem

- As we ever that when we want to reach from one station to another at that time driver takes shortest possible route to reach destination. There are many instances to find the shortest path for traveling from one place to another. That is to find which route can reach as quick as possible of a route for which the traveling cost is minimum.
- In a graph finding shortest path is the most important problem.
- Dijkstra's Algorithm is used to find the shortest path.
- The single source shortest path algorithm is based on assumption that no edges have negative weights.

DIJKSTRA'S ALGORITHM

- It is invented by Dutch computer scientist E.W. Dijkstra's which solves the problem of finding the shortest path from a point in a graph to destination with non-negative weight edge.

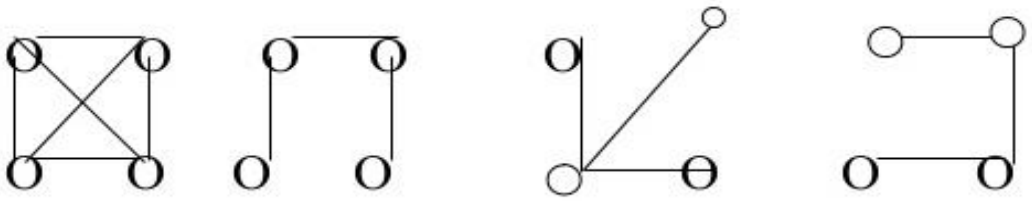
2 – Vaishalinagar
Nr. Amrapali Under Bridge
Raiya Road
Rajkot – 360001
Ph.No–(0281)2440478,2472590

3 – Vaishalinagar
Nr. Amrapali Under Bridge
Raiya Road
Rajkot - 360001
Ph.No–(0281)2471645

5. Minimal Spanning Tree

- Definition : Spanning tree : Let G = (V,E) be an un-directed connected graph.

  A minimal connected sub-graph of G which includes all the vertices of G is a spanning tree of G; (a) is a complete graph and(b),(c),(d) are three of A's spanning trees



(a) (b) (c) (d)

- A spanning tree of a graph is just a sub graph that contains all the vertices and is a tree(with no cycle).
- A graph may have many spanning trees.
- A minimum spanning tree(MST) for graph is a subgraph of graph that contains all the vertices of G(graph).
- If a graph G is not a connected graph, then it cannot have any spanning tree.
- In this case, it will have a spanning forest. Suppose a graph(G) with n vertices then the MST will have (n-1) edges, assuming that the graph is connected.
- To obtain minimum spanning tree of connected weighted and undirected graph, different algorithms are use which are listed as under:
  i. Krukal's Algorithm
  ii. Prim's Algorithm
  iii. Sollin's Algorithm

2 – Vaishalinagar
Nr. Amrapali Under Bridge
Raiya Road
Rajkot – 360001
Ph.No–(0281)2440478,2472590

3 – Vaishalinagar
Nr. Amrapali Under Bridge
Raiya Road
Rajkot - 360001
Ph.No–(0281)2471645

1 WORD QUESTION ANSWER

SR. NO.	QUESTION	ANSWER
1	DIJKSTRA'S ALGORITHM IS USED TO SOLVE _____ PROBLEMS.	SINGLE SOURCE SHORTEST PATH
2	WHICH IS THE MOST COMMONLY USED DATA STRUCTURE FOR IMPLEMENTING DIJKSTRA'S ALGORITHM?	MIN PRIORITY QUEUE
3	PRIM'S ALGORITHM IS A _____	GREEDY ALGORITHM
4	CONSIDER A COMPLETE GRAPH G WITH 4 VERTICES. THE GRAPH G HAS ____ SPANNING TREES.	13
5	TO IMPLEMENT DIJKSTRA'S SHORTEST PATH ALGORITHM ON UNWEIGHTED GRAPHS SO THAT IT RUNS IN LINEAR TIME, THE DATA STRUCTURE TO BE USED IS:	QUEUE

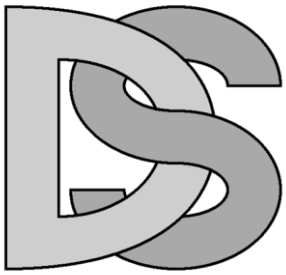
SUMMARY

- ✓ In mathematics and computer science, graph theory is the study of *graphs*, which are mathematical structures used to model pairwise relations between objects.
- ✓ A "graph" in this context is made up of "vertices" or "nodes" and lines called *edges* that connect them.
- ✓ A graph may be *undirected*, meaning that there is no distinction between the two vertices associated with each edge, or its edges may be *directed* from one vertex to another; see graph (mathematics) for more detailed definitions and for other variations in the types of graph that are commonly considered. Graphs are one of the prime objects of study in discrete mathematics.

2 – Vaishalinagar
Nr. Amrapali Under Bridge
Raiya Road
Rajkot – 360001
Ph.No–(0281)2440478,2472590

3 – Vaishalinagar
Nr. Amrapali Under Bridge
Raiya Road
Rajkot - 360001
Ph.No–(0281)2471645

UNIT-3 - PART - 1- INTRODUCTION TO DATA STRUCTURE



- Introduction
- Primitive and simple structures
- Linear and nonlinear structures file organization.

➡ Some Important Terms

1. **DATA**

Information which is input to a computer system and is then processed by mathematical and logical operation. So that it can ultimately be output in a sensible form. It usually has numbers facts letter or system that refer to or describe an object idea, condition, situation relationship or other type of information

2. **DATA-TYPE**

Data type is the set of permitted data values and certain operation on data.

DATA TYPE = PERMITTED DATA VALUES + OPERATION

3. **DATA-STRUCTURE**

Data structure is the possible ways of organizing data items that defines how the data items are stored in memory and relationship with each other. Data structure is the possible ways that defines the relationship between data items.

DATA STRUCTURE = ORGANIZED DATA + ALLOWEDOPE

4. **CELLS**

Cells is the memory area that used to store elementary data items, it can be refered as a single bit, byte, group of bytes.

5. **FIELDS**

Field is a smallest piece of information that can be reference by a programming language.

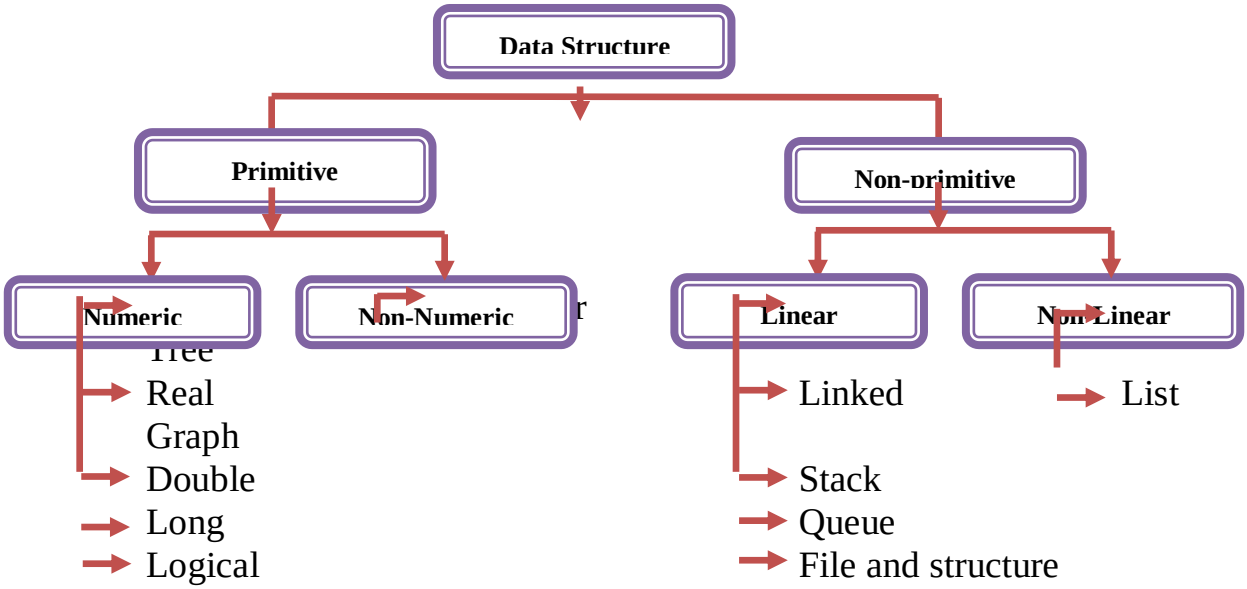
TYPES OF DATA STRUCTURE

- The collection of organized data is known as data structure.

Data structure=Organized Data + Allowed operation

There are main two types of data structure:

1) Primitive data structure
2) Non-primitive data structure



Non-Primitive data structure:-

- Non-primitive data structure means the data structure constructed by using primitive data structure.
- Non-primitive data structure is also known as composite types.
- Non-primitive data structure is classified into two parts as shown above:

a) Linear data structure:-

- In this type of data structure, the elements are arranged in sequence like an array.
- i) Array:
  - An array is the collection of structured set that holds fix number of data elements.

- An array is set of homogeneous elements (having same data type).
- There are mainly 2 types of array:
  - One-dimensional
  - Two-dimensional

ii) Stack:

- Stack is one type of data structure where information is based on LIFO (Last In First Out).
- All the insertions and deletions take place at only one end which is known as Top Of Stack(TOS).
- There are two types of stack:
  - Static stack (using array)
  - Dynamic stack (using structure or linked list).

iii) Queue:

- Queue is one type of data structure where information is based on
  FIFO(First In First Out).
- All the insertions and deletion takes place at only end known as front end.
- There are two types of queue:
  - Static Queue (using array)
  - Dynamic Queue (using structure or linked list).

iv) Linked List:-

- Linked list is defined as the collection of nodes and each node contain two parts:
  - Information part
  - Pointer to next node.
- Information part contains the data and it may consists of one or more fields.
- Pointer to next node contains the location where next information is stored.

Node

Information part Pointer to next node

- There are two types of linked list:
  - Singly linked list
  - Doubly linked list

v) File and structure:-

- File is the collection of data that is available to program whenever needed.
- There are various operations that can be performed on file.
  - Read
  - Write
  - Append
  - Copy
- Structure is the collection of data elements that may or may not have same data type (non-homogeneous data type).

b) Non-linear data structure

- In this type of data structure, data elements are not arranged in the sequence.

i) Tree:

- It is the most important non-linear data structure which stores data as branches.
- There are different types of trees like binary tree etc.

ii) Graph:

- It is also non-linear data structure which contains two main pair:- Vertices and edges.
- Depending on nature of representation, there are different types of graph.

2 – Vaishalinagar
Nr. Amrapali Under Bridge
Raiya Road
Rajkot – 360001
Ph.No–(0281)2440478,2472590

3 – Vaishalinagar
Nr. Amrapali Under Bridge
Raiya Road
Rajkot - 360001
Ph.No–(0281)2471645

1 WORD QUESTION ANSWER

SR.NO.	QUESTION	ANSWER
1	AN ALGORITHM SHOULD HAVE _____ FEATURES.?	FREE OF AMBIGUITYCONCISE CONCISE EFFICIENT
2	IF ELEMENTS OF THE DATA STRUCTURE FORM A SEQUENCE OF THE LIST THEN IT IS CALLED AS _____.	LINEAR DATA STRUCTURE
3	WHICH DATA STRUCTURES ARE INDEXED STRUCTURES?	LINEAR ARRAYS
4	OPERATIONS ON A DATA STRUCTURE MAY BE ….	CREATION DESTRUCTION SELECTION
5	WHICH DATA STRUCTURE IS NON-LINEAR TYPE?	TREE
6	FINDING THE LOCATION OF THE ELEMENT WITH A GIVEN VALUE IS:	SEARCH
7	WHICH DATA STRUCTURE CAN'T STORE THE NON-HOMOGENEOUS DATA ELEMENTS?	ARRAYS

SUMMARY

- ✓ A collection of organized data is known as DATA STRUCTURE
- ✓ A data structure mainly divides into two parts
  - Primitive data structure
  - Non primitive data structure
    - Linear data structure
    - Non-linear data structure
- ✓ There are mainly three types of array:
  - 1 D Array
  - 2 D Array
  - Multi dimensional array

2 – Vaishalinagar
Nr. Amrapali Under Bridge
Raiya Road
Rajkot – 360001
Ph.No–(0281)2440478,2472590

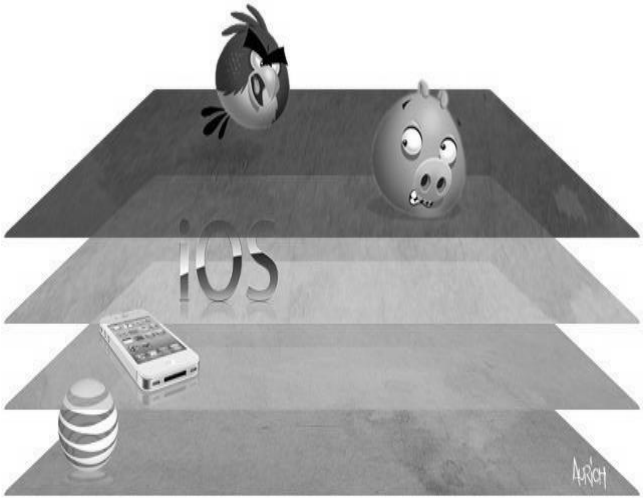
3 – Vaishalinagar
Nr. Amrapali Under Bridge
Raiya Road
Rajkot - 360001
Ph.No–(0281)2471645

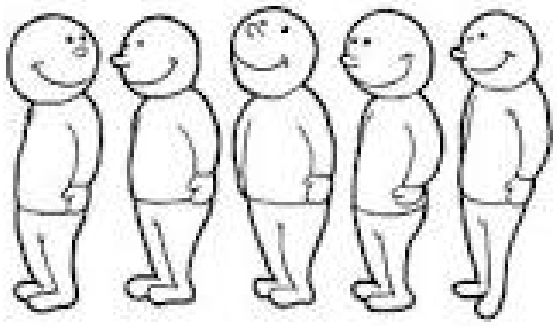
- ✓ An array and structure are related with each other in two way:
  - Array as member of structure.
  - Array of structure means structure variable as an array.
- ✓ Any data structure whose size is fixed at compile time is known as static data structure(allocation)
- ✓ Any data structure whose size is not fixed means during the program executing, but it expands at run time is known as dynamic data structure(allocation)
- ✓ When we using dynamic memory allocation, it allocates memory at run time that's why it saves memory space.
- ✓ There are mainly four types of dynamic memory allocation functions are as below:
  - malloc: it allocates memory for the requested byte of size
    p=(structure *) malloc (size * size of(structure))
  - calloc: it allocates memory for elements of an array. It initializes value to zero
    p=(structure *)calloc(n,element size)
  - free: it removes previously allocated memory
    free(p);
  - realloc: it modifies the size of previously allocated memory
    p=realloc(p,newsize)

2 – Vaishalinagar
Nr. Amrapali Under Bridge
Raiya Road
Rajkot – 360001
Ph.No–(0281)2440478,2472590

3 – Vaishalinagar
Nr. Amrapali Under Bridge
Raiya Road
Rajkot - 360001
Ph.No–(0281)2471645

UNIT-3 PART – 2 - ELEMENTARY DATA STRUCTURE





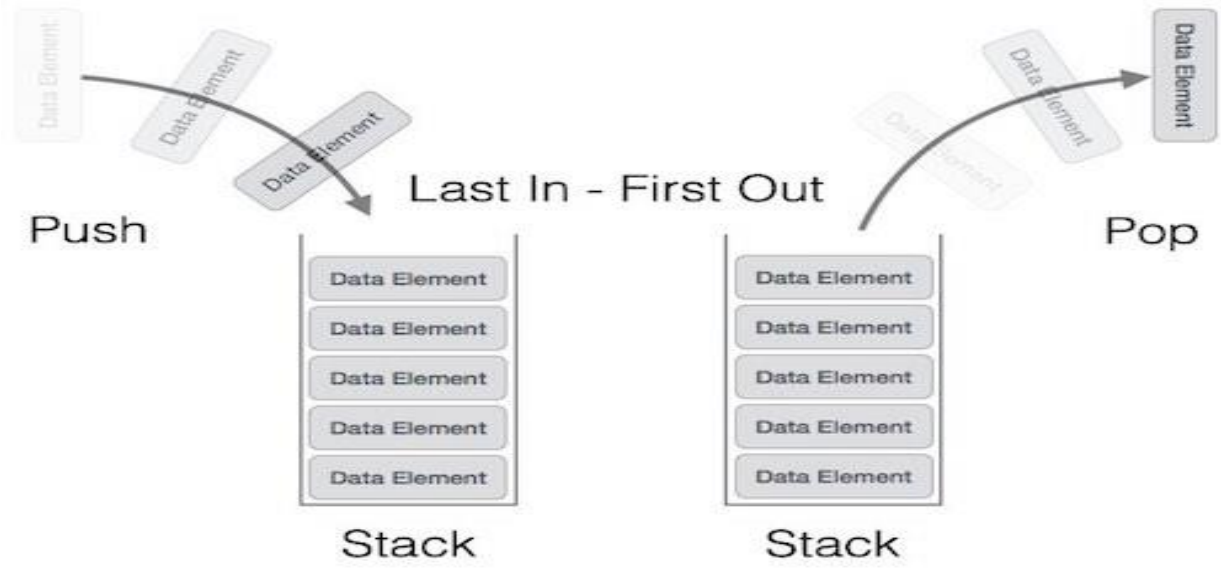
- Introduction
- Stack
  - Definition
  - Operations on stack
  - Implementation of stacks using arrays
  - Function to insert an element into the stack
  - Function to delete an element from the stack
  - Function to display the items
- Recursion and stacks
- Evaluation of expressions using stacks
  - Postfix expressions
  - Prefix expression
- Queue
  - Introduction
  - Array implementation of queues
  - Function to insert an element into the queue
  - Function to delete an element from the queue
- Circular queue

- Function to insert an element into the queue
- Function for deletion from circular queue
- Circular queue with array implementation

- Deques
- Priority Queues

What is stack?

- Stack is a linear data structure in which insertions and deletions of an element are done at one end which is known as TOS (Top Of Stack).
- Stack supports LIFO(Last In First Out)



- **Application of Stacks**
  - Stack is used in the mobile. Message sent by one user and another message sent by same user after some time, in that case the message that was sent last by the user arrives in the inbox first
  - Consider a stack of plates placed on counter in a restaurant, During dinner time, plates are taken from the top of place(stack) and waiter puts the washed plates on the top of position which is known as TOS
  - The most important application of stack is recursion.
  - It is also used in memory management and in operating systems
- **Types of stack/implementation of stack**
  - Static Stack/Stack(array)
  - Dynamic Stack(linked list)

- **Static stack:**

a. Push operation

- To insert an element on the stack we are using push operation
- This operation inserts the element only on the TOS
- Algorithm (we have to define size first)
- **Step-1**:First we have to check for stack overflow

  If(tos>=size)

  Stack is full
- **Step-2:**Tos=tos+1(increment the pointer value by 1)
- **Step-3:**Stack[tos]=ele; where ele is an element that is to be inserted
- **Step-4:** End

b. Pop operation

- To delete an element from the stack we are using pop operation

- This operation removes or deletes the top most element from stack
- Algorithm
- **Step-1:**First we have to check for stack underflow
  If(tos<0)
  Stack is empty
- **Step-2:** tos=tos-1
- **Step-3:** Print stack[tos]
- **Step-4:** End

c. Peep operation

- If we want to access some information stored at some location in stack then peep operation is required.
- The index value is subtracted from tos
- Algorithm
- **Step-1:** If tos<0---stack is empty
- **Step-2:**Input the position of the element that you want to read
- **Step-3:**If(element<0 || element>tos+1)
  Out of range
- **Step-4:** Else Print peeped element
  Stack[tos-pos+1]
- **Step-5:** End

d. Display operation

- **Step-1:** If(tos<0)—stack is empty
- **Step-2:** else
  for(i=tos;i>=0;i--)
  print element that is stack[i]
- **Step-3:** End

e. Update operation

- **Step-1:** If tos<0—stack is empty

- **Step-2:** Input the position of the element that you want to change
- **Step-3:** if(pos<0 || pos>tos+1)
  out of range
- **Step-4:**else
  Enter new data and print the new data [tos-pos+1]
- **Step-5:** End

Program:

```
#include<stdio.h>
#include<conio.h>
#define size 100
int tos=-1;
int stack[size];
void push(int);
void display();
void peep();
void pop();
void update();
void main()
{
   clrscr();
   push(10);
   push(11);
   push(12);
   display();
   peep();
   update();
   display();
   pop();
   display();
   getch();
}
void push(int ele)
{
   if(tos>=size)
```

```
            printf("\nStack is full");
    else
            tos++;
            stack[tos]=ele;
}
void display()
{
    int i;
    if(tos<0)
            printf("\nstack is empty");
    else
    {
            for(i=tos;i>=0;i--)
printf("\nElement at position %d is %d",i,stack[i]);
    }
}
void peep()
{
    int pos;
    if(tos<0)
            printf("\nstack is empty");
    else
            printf("\nEnter value of pos:");
            scanf("%d",&pos);
            if(pos<0 || pos>tos+1)
            printf("\nout of range");
            else
printf("\nPeeped element is %d",stack[tos-pos+1]);
}
void pop()
{
    if(tos<0)
            printf("\nStack is empty");
    else
            printf("\nelement deleted");
            tos--;
```

```
}
void update()
{
   int pos;
   if(tos<0)
         printf("\nstack is empty");
   else
   {
         printf("\nEnter  postion:");
         scanf("%d",&pos);
         if(pos<=0 || pos>tos+1)
         printf("\nout of range");
         else
         {
   printf("\nenter new value:");
   scanf("%d",&stack[tos-pos+1]);
   //printf("\nelement=%d",stack[tos-pos+1]);
         }
   }
}
```

- **Dynamic stack:**
  - **a.** Push operation
    - **Step-1**:Allocate the memory to the node (node *t)
      t=(node *)malloc(sizeof(node));
    - **Step-2:**Assign data part and next part of node
      t->info=ele;
      t->next=tos;
      top=t;
    - **Step-3:** End
  - **b.** Pop operation
    - **Step-1:**Check whether stack is empty or not.
      If(tos==NULL)
      Stack is empty
    - **Step-2:** If the stack is not empty then
      t=tos

```
tos=t->next
free(t)
```

- **Step-3:** End

c. Peep operation

- **Step-1:** Enter the position from user
- **Step-2:**Initialize i=0,j=1;
- **Step-3:**node *temp

```
for(temp=tos; temp!=NULL;temp=temp->next)
        i++;
```

- **Step-4:**if(pos>i)—stack is full
- **Step-5:**else

```
temp=tos;
for(j=1;j<pos;j++)
    temp=temp->next;
printf("%d",temp->info)
```

- **Step-5:** End

d. Display operation

- **Step-1:** while(temp!=NULL)

```
printf("%d",temp->info)
temp=temp->next;
```

- **Step-2:** End

e. Update operation

- **Step-1:** Enter the position from user
- **Step-2:**Initialize i=0,j=1;
- **Step-3:**node *temp

```
for(temp=tos; temp!=NULL;temp=temp->next)
        i++;
```

- **Step-4:**if(pos>i)—stack is full
- **Step-5:**else

```
temp=tos;
```

```
new1=(node *)malloc(sizeof(node)
scanf("%d",new1->info)
for(j=1;j<pos;j++)
        temp=temp->next;
temp->info=new1->info
```

- **Step-5:** End

Program

```
#include<stdio.h>
#include<conio.h>
struct list
{
   int info;
   struct list *next;
};
typedef struct list node;
struct list *new1;
node *tos;
void push(int);
void pop();
void peep();
void display();
void update();
void main()
{
   tos=NULL;
   clrscr();
   push(10);
   push(11);
   push(12);
   push(13);
   push(14);
   display();
   pop();
   display();
   peep();
   update();
```

```
    display();
    getch();
}
void push(int ele)
{
    node *t;
    t=(node *)malloc(sizeof(node));
    t->info=ele;
    t->next=tos;
    tos=t;
}
void display()
{
    node *temp;
    temp=tos;
    while(temp!=NULL)
    {
            printf("\nStack Element is:%d",temp->info);
            temp=temp->next;
    }
}
void pop()
{
    node *temp;
    if(tos==NULL)
            printf("\nStack is empty");
    else
            temp=tos;
            tos=temp->next;
            free(temp);
}
void peep()
{
    int pos,i,j;
    node *temp;
    printf("\nEnter position:");
```

```
    scanf("%d",&pos);
    for(temp=pos;temp!=NULL;temp=temp->next)
    {
            i++;
    }
    if(pos>i)
            printf("\nStack is empty");
    else
    {
            temp=tos;
            for(j=1;j<pos;j++)
            {
                    temp=temp->next;
            }
            printf("\nValue at %d is %d",pos,temp->info);
    }
}
void update()
{
    int pos,i=0,j;
    node *temp;
    printf("\nEnter position:");
    scanf("%d",&pos);
    for(temp=pos;temp!=NULL;temp=temp->next)
    {
            i++;
    }
    if(pos>i)
            printf("\nStack is full");
    else
    {
            new1=(node *)malloc(sizeof(node));
            printf("\nenter new value");
            scanf("%d",&new1->info);
            temp=tos;
            for(j=1;j<pos;j++)
```

```
            {
                temp=temp->next;
            }
            temp->info=new1->info;
        }
    }
```

Recursion & Evaluation of expression using Stack

♦ The main applications of stack are:

1) Polish Notation
2) Recursion

1) Polish Notation:- The process of writing the operators either before or after the operands in known as polish notation. It has 3 categories:
    1. Infix : Operators are written between two operands.
    2. Prefix: operators are written before their operands.
    3. Postfix: Operators are written after their operands.

Example:

Infix: a+b*c
Prefix: +a*bc
Postfix: abc*+

Lets see how the stack is used in polish notation.
Step 1: Suppose we want to convert infix to prefix then we have to start the infix statement from back. When the operator comes then they should not be pushed in stack.

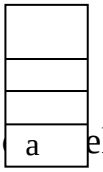
Infix: a+b*c

❖ After the insertion of b in the stack, operator + comes so now the precedence should be checked. As the precedence of + is less than * so now elements should be popped.

- The first element that will be popped is b then c. Then * operator is written. So we get: *bc
- Now the stack is empty as b and c are popped.

Operator
+

- The pushed in the stack.

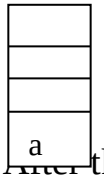


Operator
+

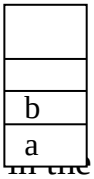
S element a is popped. Now we get
+a *bc which is our prefix expression.

Step 2: Suppose we want to convert infix to postfix then we have to start the infix statement from front. When the operator comes then they should not be pushed in stack.

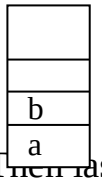
Infix: a+b*



Operator
+

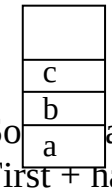


- After the insertion of b in the stack, operator * comes so now the precedence should be checked. As the precedence of * is more than + so it remains as it is.



Operator
+,*

- Then last element c is pushed.



Operator
+,*

- So as the expression is complete, the elements are popped.
- First + have low precedence so it is placed at last then * operator. After that elements are popped and are written from back that is as c is popped first then c*+ then b is popped so bc*+ and then a is popped so abc*+. Finally we get

abc*+ which is our postfix expression

2) **Recursion**:- The process in which function calls itself is known as recursion.

Example:

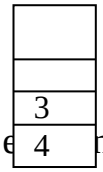
```
int fact(int n)
{
    if(n==1)
        return 1;
    else
        return n*fact(n-1);
}
```

Stack in recursion is used in following way:

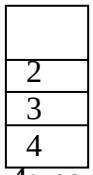
- Suppose we want to find factorial of 4. So n=4.and fact() will be recursively called.

Step 1: 4*fact(4-1).

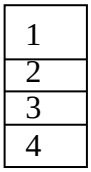
Step 2: now n=3 , so 4* 3*fact(3-1)



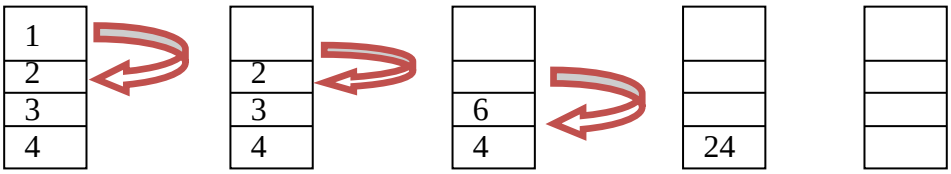
Step 3: now n=2, so 4*3*2*fact(2-1)



step 4: now n=1, so it will return 1.



Now the elements are popped in following way:



2 – Vaishalinagar
Nr. Amrapali Under Bridge
Raiya Road
Rajkot – 360001
Ph.No–(0281)2440478,2472590

3 – Vaishalinagar
Nr. Amrapali Under Bridge
Raiya Road
Rajkot - 360001
Ph.No–(0281)2471645

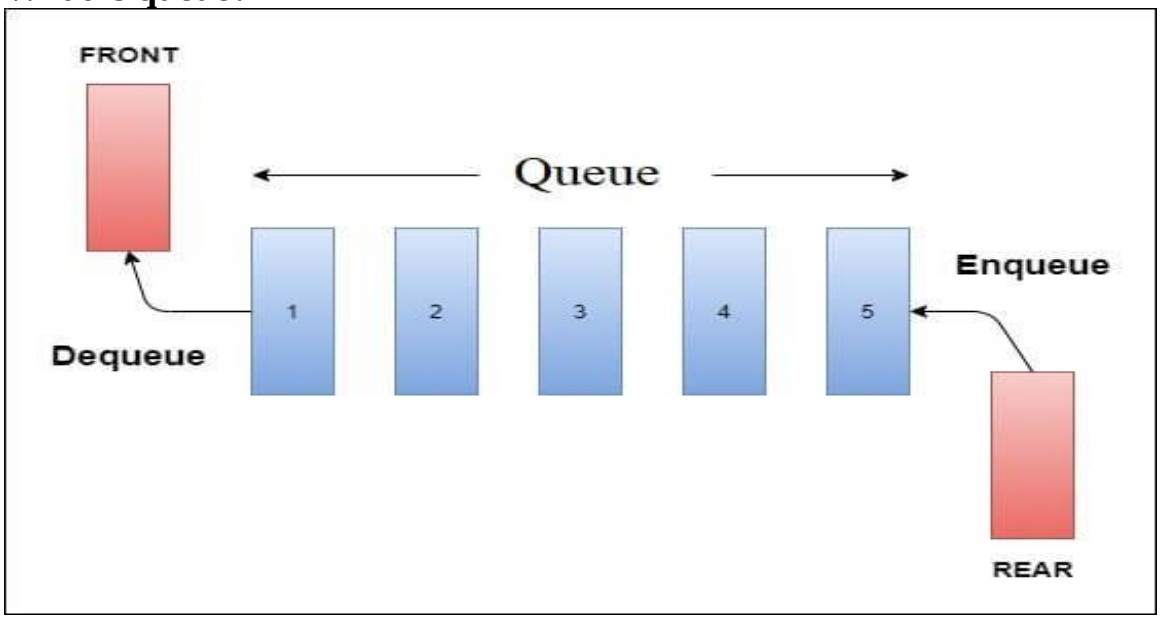
1 WORD QUESTION ANSWER

SR.NO.	QUESTION	ANSWER
1	PROCESS OF INSERTING AN ELEMENT IN STACK IS CALLED _____	PUSH
2	PROCESS OF REMOVING AN ELEMENT FROM STACK IS CALLED _____	POP
3	IN A STACK, IF A USER TRIES TO REMOVE AN ELEMENT FROM AN EMPTY STACK IT IS CALLED _____	UNDERFLOW
4	PUSHING AN ELEMENT INTO STACK ALREADY HAVING FIVE ELEMENTS AND STACK SIZE OF 5, THEN STACK BECOMES _____	OVERFLOW
5	FULL FORM OF LIFO_____	LAST IN FIRST OUT
6	THE DATA STRUCTURE REQUIRED TO CHECK WHETHER AN EXPRESSION CONTAINS A BALANCED PARENTHESIS IS?	STACK
7	WHAT DATA STRUCTURE WOULD YOU MOSTLY LIKELY SEE IN NON RECURSIVE IMPLEMENTATION OF A RECURSIVE ALGORITHM?	STACK
8	THE PROCESS OF ACCESSING DATA STORED IN A SERIAL ACCESS MEMORY IS SIMILAR TO MANIPULATING DATA ON A _____	STACK
9	WHICH DATA STRUCTURE IS USED FOR IMPLEMENTING RECURSION?	STACK
10	WHICH DATA STRUCTURE IS NEEDED TO CONVERT INFIX NOTATION TO POSTFIX NOTATION?	STACK
11	THE POSTFIX FORM OF A*B+C/D IS?	AB*CD/+
12	THE TYPE OF EXPRESSION IN WHICH OPERATOR SUCCEEDS ITS OPERANDS IS?	POSTFIX EXPRESSION

2 – Vaishalinagar
Nr. Amrapali Under Bridge
Raiya Road
Rajkot – 360001
Ph.No–(0281)2440478,2472590

3 – Vaishalinagar
Nr. Amrapali Under Bridge
Raiya Road
Rajkot - 360001
Ph.No–(0281)2471645

What is queue?



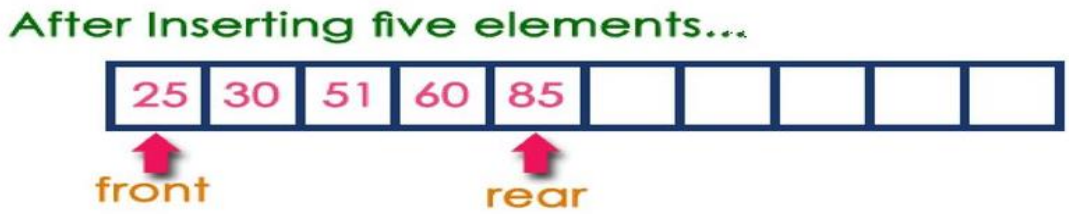
- Queue is the linear data structure in which information is based on FIFO(First In First Out) or FCFS(First Come First Server)
- In queue insertion of an element is performed at one end known as back or rear end and deletion is performed at another end known as front end.

- Insertion operation is known as **enqueue** and deletion operation is known as **dqueue**

STACK	QUEUE
In stack both the insertion and deletion takes place at only one end know as TOS (top of stack)	In queue insertion takes place at rear and deletion takes place at front
In stack we have to keep track of only TOS	In queue we have to keep track of both the ends that is front and rear

- **Types of queue/implementation of queue**
  - Static Queue/Queue(array)

Queue after inserting 25, 30, 51, 60 and 85.



Operations on a Queue

The following operations are performed on a queue data structure...

- enQueue(value) - (To insert an element into the queue)
- deQueue() - (To delete an element from the queue)
- display() - (To display the elements of the queue)

  - Dynamic Queue(linked list)
  - Circular Queue

- **Static queue:**
  - **a.** Insert operation
    - Algorithm (we have to define size first)
    - **Step-1**:If rear>=size
      Output “queue overflow”

- **Step-2:**rear = rear +1
- **Step-3:**q[rear]=element
- **Step-4:** End

b. Delete operation

- Algorithm
- **Step-1:**If front <0
  Output "queue underflow"
- **Step-2:** front ++;
- **Step-3:** End

c. Display operation

- Algorithm
- **Step-1:** If fornt<0---queue underflow
- **Step-2:**for(i=front;i<=rear;i++)
  Printf("%d",queue[i]);
- **Step-3:**End

Program:

```
#include<stdio.h>
#include<conio.h>
#define size 100
int front=-1,rear=-1;
int queue[size];
void insert(int);
void deleted();
void display();
void main()
{
   clrscr();
   insert(10);
   insert(11);
   insert(12);
   insert(13);
   insert(14);
```

2 – Vaishalinagar
Nr. Amrapali Under Bridge
Raiya Road
Rajkot – 360001
Ph.No–(0281)2440478,2472590

3 – Vaishalinagar
Nr. Amrapali Under Bridge
Raiya Road
Rajkot - 360001
Ph.No–(0281)2471645

```
    display();
    printf("\n\n");
    deleted();
    display();
    getch();
}
void insert(int ele)
{
    if(rear>size)
            printf("\nQueue is full");
    else
    {
            rear++;
            queue[rear]=ele;
            if(rear==0)
                    front=0;
    }
}
void deleted()
{
    if(front<0)
            printf("\nQueue is empty");
    else
    {
            if(front==rear)
            {
                    front=-1;
                    rear=-1;
            }
            else
                    front++;
    }
}
void display()
{
    int i;
```

```
    if(front<0)
        printf("\nQueue empty");
    else
    {
        for(i=front;i<=rear;i++)
        {
printf("\nElement at position %d is %d",i,queue[i]);
        }
    }
}
```

- **Dynamic queue:**
  - **a.** Insert operation
    - **Step-1**:Allocate the memory to the node (node *q)
      q=(node *)malloc(sizeof(node));
    - **Step-2:**Assign data part and next part of node
      q->info=ele;
      q->next=NULL;
    - **Step-3:**If queue is empty then:
      if(front==NULL)
      front=rear=q
    - **Step-4:**If queue is not empty
      rear-next=q;
      rear=q;
    - **Step-5:** End

  - **b.** Delete operation
    - **Step-1:**Check whether queue is empty or not.
    - **Step-2:** If the queue is empty then
      If(front==NULL)---queue empty
    - **Step-3:** If queue is not empty then
      node *q
      q=front
      front=q->next

- **Step-4:**free(q)
- **Step-5:**End

c. Display operation

- **Step-1:** node *q
  q=front
- **Step-2:**Check if queue is empty or not
  If(q==NULL)---queue empty
- **Step-3:** If queue is not empty then
  While(q!=NULL)
  Printf("%d",q->info);
  q=q->next
- **Step-**4:END

Program

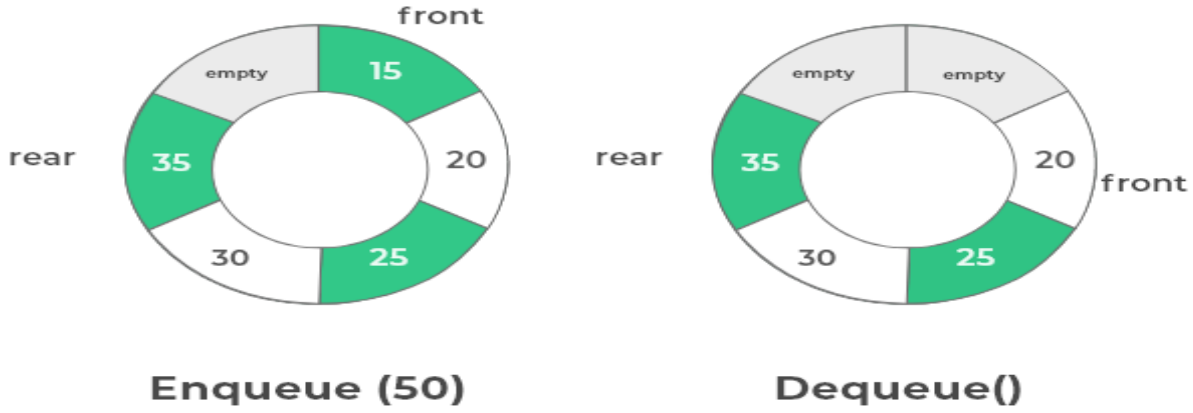
```
#include<stdio.h>
#include<conio.h>
struct list
{
   int info;
   struct list *next;
};
typedef struct list node;
node *front,*rear;
void insert(int);
void deleted();
void display();
void main()
{
   front=rear=NULL;
   clrscr();
   insert(10);
   insert(11);
   insert(12);
   display();
```

```
    deleted();
    display();
    getch();
}
void insert(int ele)
{
    node *t;
    t=(node *)malloc(sizeof(node));
    t->info=ele;
    t->next=NULL;
    if(front==NULL)
    {
            front=rear=t;
    }
    else
    {
            rear->next=t;
            rear=t;
    }
}
void deleted()
{
    node *temp;
    temp=front;
    if(temp==NULL)
            printf("\nQueue is empty");
    else
            front=temp->next;
            free(temp);
}
void display()
{
    node *temp;
    int i=0;
    temp=front;
    if(temp==NULL)
```

```
            printf("\nQueue is empty");
        else
        {
            while(temp!=NULL)
            {
                printf("\nElement is %d",temp->info);
                temp=temp->next;
            }
        }
    }
```

- **Circular queue:**
  - The problem with simple queue is that once we insert the elements and when we are removing elements using front pointer these elements becomes blank. As the elements become blank we cannot insert any elements in that blank space.
  - So to overcome this problem there is technique available known as circular queue. In this technique when rear reaches the queue's size the first element will become the queue's new rear

  - **NOTE:** When the queue is full, the first element of queue becomes the rear if and only if front has moved forward otherwise it will be again in a "queue full(overflow)" state.




```
Enqueue()    rear = (rear + 1) % SIZE;
Dequeue()    front = (front + 1) % SIZE;
```

a. Insert operation

- **Step-1**:If front==1 and rear==size (output queue overflow)
- **Step-2:**else if (front==0)
  Front=rear=1
  q[rear]=element
- **Step-3:**else if (rear==size)
  rear=1
  q[rear]=element
- **Step-4:**else
  rear=rear+1
  q[rear]=element
- **Step-5:** End

b. Delete operation

- **Step-1:**If front==0 (output queue is empty)
- **Step-2:** If front==rear
  front=rear=0
- **Step-3:** else if (front==size)
  front=1
- **Step-4:**else

front=front+1

- **Step-5:** End

c. Display operation

- **Step-1:** If front==0 (output queue is empty)
- **Step-2:** else if (front >rear)

```
for(i=1;i<=rear;i++)
    printf("%d",queue[i])
for(i=front;i<=size;i++)
    printf("%d",queue[i])
```

- **Step-3:** else

```
for(i=front;i<=rear;i++)
        printf("%d",queue[i])
```

- **Step-**4:END

Program

```
#include<stdio.h>
#include<conio.h>
#define size 5
int queue[size];
int front=0,rear=0;
void insert(int);
void deleted();
void display();

void main()
{
   clrscr();
   insert(10);
   insert(20);
   insert(30);
   insert(40);
   insert(50);
   display();
   printf("\n\n");
```

2 – Vaishalinagar
Nr. Amrapali Under Bridge
Raiya Road
Rajkot – 360001
Ph.No–(0281)2440478,2472590

3 – Vaishalinagar
Nr. Amrapali Under Bridge
Raiya Road
Rajkot - 360001
Ph.No–(0281)2471645

```
    deleted();
    display();
    printf("\n\n");
    insert(60);
    display();
    getch();
}
void insert(int ele)
{
    if(front==1 && rear==size)
            printf("\nQueue is full");
    else if(front==0)
    {
            front=rear=1;
            queue[rear]=ele;
    }
    else if(rear==size)
    {
            rear=1;
            queue[rear]=ele;
    }
    else
    {
            rear++;
            queue[rear]=ele;
    }
}
void deleted()
{
    if(front==0)
            printf("\nQueue is empty");
    else if(front==rear)
            front=rear=0;
    else if(front==size)
            front=1;
    else
```


2 – Vaishalinagar
Nr. Amrapali Under Bridge
Raiya Road
Rajkot – 360001
Ph.No–(0281)2440478,2472590
3 – Vaishalinagar
Nr. Amrapali Under Bridge
Raiya Road
Rajkot - 360001
Ph.No–(0281)2471645

```
            front++;
}
void display()
{
    int i;
    if(front==0)
            printf("\nQueue is empty");
    else if(front>rear)
    {
            for(i=1;i<=rear;i++)
            {
    printf("\nElement at %d is %d",i,queue[i]);
            }
            for(i=front;i<=size;i++)
            {
    printf("\nElement at %d is %d",i,queue[i]);
            }
    }
    else
    {
            for(i=front;i<=rear;i++)
            {
    printf("\nElement at %d is %d",i,queue[i]);
            }
    }
}
```

SHREE H. N. SHUKLA COLLEGE OF I.T. & MGMT.

(AFFILIATED TO SAURASHTRA UNIVERSITY)

2 – Vaishalinagar
Nr. Amrapali Under Bridge
Raiya Road
Rajkot – 360001
Ph.No–(0281)2440478,2472590

3 – Vaishalinagar
Nr. Amrapali Under Bridge
Raiya Road
Rajkot - 360001
Ph.No–(0281)2471645

1 WORD QUESTION ANSWER

SR.NO.	QUESTION	ANSWER
1	A LINEAR LIST OF ELEMENTS IN WHICH DELETION CAN BE DONE FROM ONE END (FRONT) AND INSERTION CAN TAKE PLACE ONLY AT THE OTHER END (REAR) IS KNOWN AS _____	QUEUE
2	THE DATA STRUCTURE REQUIRED FOR BREADTH FIRST TRAVERSAL ON A GRAPH IS?	QUEUE
3	A QUEUE FOLLOWS _____ PRINCIPLE	FIFO (FIRST IN FIRST OUT
4	FULL FORM OF FIFO_____	FIRST IN FIRST OUT
5	CIRCULAR QUEUE IS ALSO KNOWN AS _____	RING BUFFEI
6	IF THE ELEMENTS “A”, “B”, “C” AND “D” ARE PLACED IN A QUEUE AND ARE DELETED ONE AT A TIME, IN WHAT ORDER WILL THEY BE REMOVED?	ABCD
7	A DATA STRUCTURE IN WHICH ELEMENTS CAN BE INSERTED OR DELETED AT/FROM BOTH ENDS BUT NOT IN THE MIDDLE IS?	DEQUEUE
8	A NORMAL QUEUE, IF IMPLEMENTED USING AN ARRAY OF SIZE MAX_SIZE, GETS FULL WHEN?	REAR = MAX_SIZE – 1
9	WITH WHAT DATA STRUCTURE CAN A PRIORITY QUEUE BE IMPLEMENTED?	TREE
10	WHAT IS THE TIME COMPLEXITY TO INSERT A NODE BASED ON KEY IN A PRIORITY QUEUE?	O(N)
11	WHAT IS THE TIME COMPLEXITY TO INSERT A NODE BASED ON POSITION IN A PRIORITY QUEUE?	O(N)

2 – Vaishalinagar
Nr. Amrapali Under Bridge
Raiya Road
Rajkot – 360001
Ph.No–(0281)2440478,2472590

3 – Vaishalinagar
Nr. Amrapali Under Bridge
Raiya Road
Rajkot - 360001
Ph.No–(0281)2471645

SUMMARY

- ✓ Stack is a linear data structure.
- ✓ It has LIFO structure means that last inputted element is the first out from the stack.
- ✓ The place from insertion of elements and deletions of elements are take place is known as TOS(Top of the stack)
- ✓ It supports various operations for the manipulation of elements. If we want to insert elements then push operations is used.
- ✓ If we want to deleted elements then **pop** operation is used.
- ✓ If we want to know the top of the stack element then **peep** operation is used.
- ✓ If we want to know the size of stack elements then we have to create **size()** function through which we find out the size of it.
- ✓ If we want to know the stack is full or empty and based on that appropriated message is display then we have to create **isFull()** and **isEmpty()**function for that.
- ✓ Stack is very important part and very useful in large applications
- ✓ For example,it is used in memory management and in operating system.
- ✓ Another example is the tennis balls in their container(we cannot remove 2 balls at the same time)
- ✓ There are two ways through stack is implemented are static stack and dynamic stack.
- ✓ Static stack uses array to perform all the operations on stack and dynamic stack uses structure and pointer to perform all the operations
- ✓ Queue is a linear data structure which works FCFS (first come first serve) basis.
- ✓ A queue insert element at one end is known as rear pointer variable whereas it delete element
- ✓ From one end is known as front pointer variable. It means that insertion of elements and
- ✓ Deletion of it occurs at different end of a queue.
- ✓ To insert elements at rear end a queue performs enqueue operations. To delete elements from the front end a queue performs dequeue operations and we can retrieve elements from the both end also.

- ✓ We can implement static queue as well as dynamic queue in a data structure.
- ✓ There are lots many application In which we are using queues. Consider one example of time sharing computer. System in which many users share the system simultaneously. The procedure is designed with the use of round robin technique. The railway reservation counter is also an example of queue where the people collect their tickets on FIFO or FCFS based.
- ✓ There is a main problem in simple queue is that our array size is fixed and there fore when rear pointer variable is equal to array size (i.e. rear=size) we can't insert any elements into queue.
- ✓ With the use of front pointer variable we can delete elements from the queue. Each time the front pointer variable is increased by 1 when we delete elements from it. But the main problem is that when our front= rear (i.e. front variable value becomes equal to rear pointer variable) at that time previously deleted element's memory space is free though we can't insert elements into it. Because we can only insert elements at rear end but rear=front that's why not possible.
- ✓ To overcome this problem fortunately it has another type which is known as circular queue.
- ✓ So in this when 'rear' reaches the queue's size the first element will become the queue's new 'rear'.
- ✓ Once the queue is full the 'first' element of the queue becomes the 'rear' most element, if and only if the 'front' has moved forward; otherwise it will again be in a 'queue overflow' state.
- ✓ In a circular queue when rear = n, if we insert and element then this element is assigned to Q [1] instead of increasing rear to n + 1 . Suppose queue contains only one element that is front = rear != NULL and suppose that the element is removed then the front and rear pointers are now assigned NULL to indicate that the queue is empty.
- ✓ There is another technique also available which is known as double ended queue (DEQUE). It works same like as simple queue but insertion

2 – Vaishalinagar
Nr. Amrapali Under Bridge
Raiya Road
Rajkot – 360001
Ph.No–(0281)2440478,2472590

3 – Vaishalinagar
Nr. Amrapali Under Bridge
Raiya Road
Rajkot - 360001
Ph.No–(0281)2471645

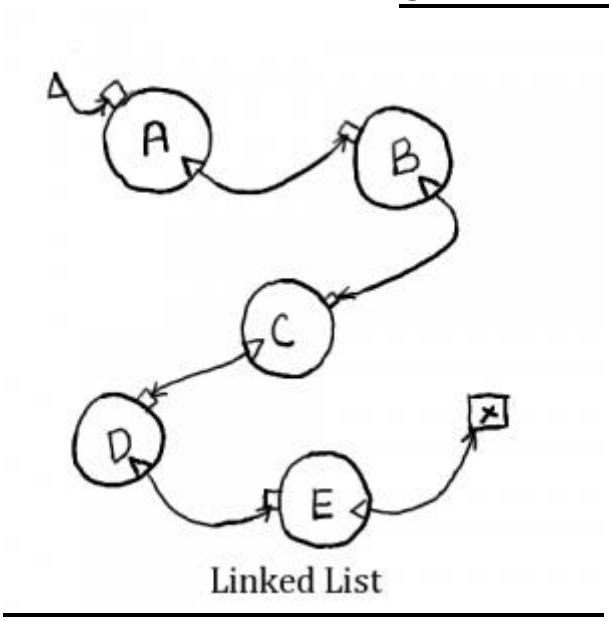
and deletions are possible at either end. Elements can be inserted or deleted from the front or rear end. It is a general form of stack and queue.

- ✓ There are mainly two types of de queue:
  - **Input restricted dequeue:**
    - **It allows insetions at only one end of the list whereas deletion can occur either both the ends.**
  - **Output restricted dequeue:**
    - **It allows deletions at only one end of the list whereas insertion can occur either both the ends of list.**

2 – Vaishalinagar
Nr. Amrapali Under Bridge
Raiya Road
Rajkot – 360001
Ph.No–(0281)2440478,2472590

3 – Vaishalinagar
Nr. Amrapali Under Bridge
Raiya Road
Rajkot - 360001
Ph.No–(0281)2471645

UNIT -4 - LINK LIST



Linked List

- Introduction
- Singly linked lists.
  - Implementation of linked list
  - Insertion of a node at the beginning
  - Insertion of a node at the end
  - Insertion of a node after a specified node
  - Traversing the entire linked list
  - Deletion of a node from linked list
- Concatenation of linked lists
- Merging of linked lists
- Reversing of linked list
- Doubly linked list.
  - Implementation of doubly linked list
- Circular linked list

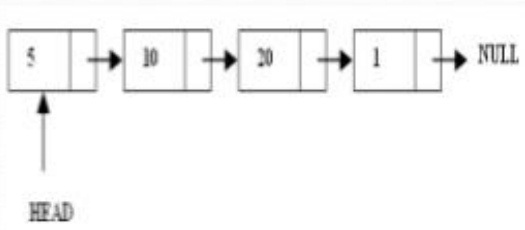
Applications of the linked lists

2 – Vaishalinagar
Nr. Amrapali Under Bridge
Raiya Road
Rajkot – 360001
Ph.No–(0281)2440478,2472590

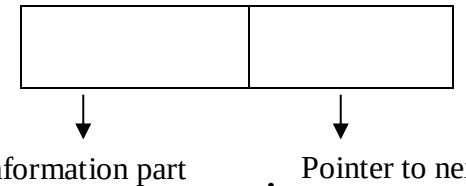
3 – Vaishalinagar
Nr. Amrapali Under Bridge
Raiya Road
Rajkot - 360001
Ph.No–(0281)2471645

What is Linked list?

- A linked list is a linear data structure.
- Nodes make up linked lists.
- Nodes are structures made up of data and a pointer to another node.
- Usually the pointer is called next.



- A linked list is defined as collection (sequence) of nodes. Each node has two parts
  - Information
  - Pointer to next node
- **Information**
  - Information part may consist of one or more than one fields.
- **Pointer to next node**
  - Pointer to next node contains the address of location where next information is stored
  - The last node of the list contains NULL in the pointer field



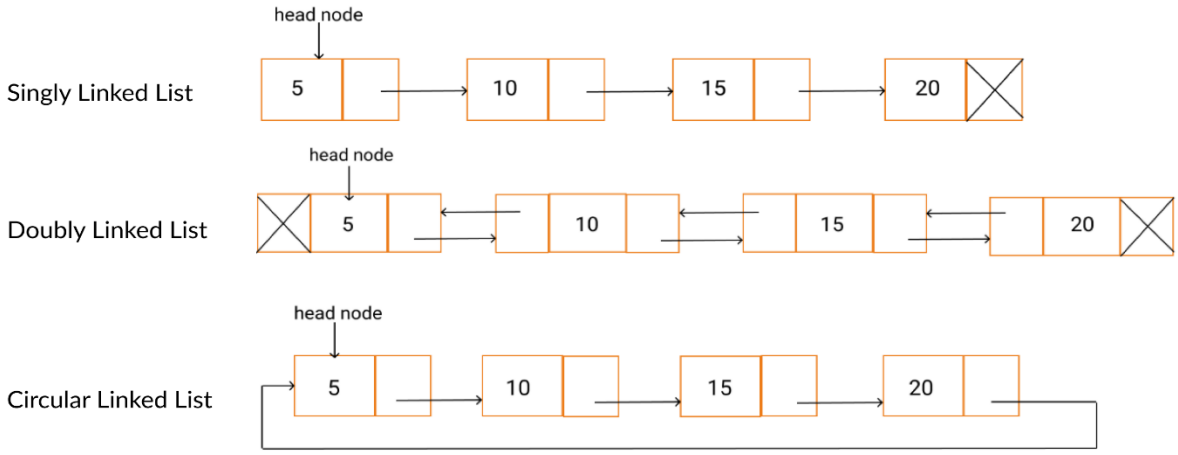
Note: Linked list is also kn[illegible]**ntial** data type because it contains pointer that points to same data type

Advantage: A link list is a dynamic data structure and so the size of linked list can be changed during execution of program.

Disadvantage: The limitation of linked list is that it consumes extra space when compared to array because each node must also contain the address of next item.

Types of linked list are:

1. Singly Linked List
2. Doubly Linked List
3. Circular Linked List (Singly & Doubly)
4. Header Linked List



Applications of linked list are:

1. **In line editor:**
   - One interesting use of linked list is line editor. We can keep a link list of lines nodes. Each containing line number, a line of text and pointer to next line information node.
2. **In string manipulation:**
   - Variable string length can be represented as linked list. A string may be declared as a record that contains a string count & pointer to linked list of characters.

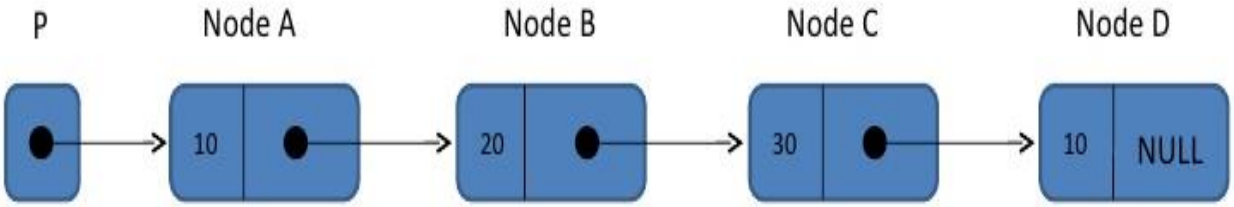
3. **In implementation of space matrix:**
    - A space matrix is a table which relatively with few non-zero elements.
4. **In operating system:**
    - The allocation of memory space may be managed by doubly linked list of various size block of memory. In multi-user system the operating system may keep track of user jobs waiting to execute through linked list queue.
5. **Implementing stack and queue:**
    - It is easy to implement stack & queue operation using linked list rather than array implementations of stack & queue.
6. **Polynomial manipulation:**
    - A linked list uses as a typical term of polynomial. The common operation performs on polynomial are addition, subtraction, multiplication, division, integration and differentiation.
7. **Linked dictionary:**
    - An important part of any compiler is the construction and maintenance of a dictionary containing name and their associated values, such dictionary is also called a symbol table

2 – Vaishalinagar
Nr. Amrapali Under Bridge
Raiya Road
Rajkot – 360001
Ph.No–(0281)2440478,2472590

3 – Vaishalinagar
Nr. Amrapali Under Bridge
Raiya Road
Rajkot - 360001
Ph.No–(0281)2471645

- **Singly Linked List:**

- A list implemented by each item having a link to the next item.
- Head points to the first node.
- Last node points to NULL.



a. Singly linked list is the most basic of all the linked data structure
b. A singly linked list is a collection of nodes and each node contains the pointer to next element
c. In singly link list we can move from left to right that is only in one dimension but we cannot return back.
d. Each node represents structure, containing variables for information and a structure pointer to itself.

```
struct list
{
        int info;
        struct node *next;
};
typedef  struct list node;
node *p;
```

e. Create operation

- Algorithm
- **Step-1**Check whether any node exists in the list or not
- **Step-2:**If list is Empty then create node

```
if(q==NULL)
{
        q=(node *)malloc(sizeof(node));
        q->data=info;
        q->next=NULL;
}
```

- **Step-3:** If the list is not Empty then create the node after the last node

```
while(q->next!=NULL)
{
        q=q->next;
}
q->next=(node *)malloc(sizeof(node));
q->next->data=info;
q->next->next=NULL;
```

- **Step-4:** End

f. Display operation

- Algorithm
- **Step-1:**

```
while(q!=NULL)
{
        printf("\nElement is %d",q->info);
        q=q->next;
}
```

- **Step-2:** End

g. Add node at beginning operation

- Algorithm
- **Step-1:** Allocate the memory to the node P that is to be inserted in the beginning

```
p=(node *)malloc(sizeof(node))
```

- **Step-2:** Allocate the data and pointer to next node part

```
p->data=info
p->next=q;
```

- **Step-3:** End

h. Add node at after operation

- Algorithm
- **Step-1:** Take a temporary node which is to be inserted (node *temp)
- **Step-2:**for(i=1;i<pos;i++)

```
{
        q=q->next;
        if(q==NULL)
            printf("\nout of range");
}
```

- **Step-3:**Allocate memory to temporary node and data and next part

```
temp=(node *)malloc(sizeof(node))
temp->info=ele
temp->next=q->next;
q->next=temp;
```

- **Step-4:**End

i. Count operation

- Algorithm
- **Step-1:** Initialize one counter variable for counting total no of nodes (c=0)
- **Step-2:**while(q!=NULL)

```
{
    q=q->next
}
return c;
```

- **Step-3:**End

Program:

```
#include<stdio.h>
#include<conio.h>
struct list
{
   int info;
   struct list *next;
```


2 – Vaishalinagar
Nr. Amrapali Under Bridge
Raiya Road
Rajkot – 360001
Ph.No–(0281)2440478,2472590

3 – Vaishalinagar
Nr. Amrapali Under Bridge
Raiya Road
Rajkot - 360001
Ph.No–(0281)2471645

```
};
typedef struct list node;
node *p;
void create(int,node *);
void display(node *);
void addbeg(int,node *);
void addafter(int,int,node *);
void deleted(int,node *);
int count(node *);
void sort(node *);
void search(int,node *);
void main()
{
   p=NULL;
   clrscr();
   create(10,p);
   create(20,p);
   create(5,p);
   display(p);
   addbeg(0,p);
   printf("\n\n");
   addafter(2,500,p);
   deleted(20,p);
   display(p);
   sort(p);
   printf("\n\n");
   display(p);
   search(15,p);
   getch();
}
void create(int ele,node *q)
{
   if(q==NULL)
   {
         p=(node *)malloc(sizeof(node));
         p->info=ele;
```

```
            p->next=NULL;
    }
    else
    {
            while(q->next!=NULL)
            {
                    q=q->next;
            }
            q->next=(node *)malloc(sizeof(node));
            q->next->info=ele;
            q->next->next=NULL;
    }
}
void display(node *q)
{
    while(q!=NULL)
    {
            printf("\nElement is %d",q->info);
            q=q->next;
    }
}
int count(node *q)
{
    int c=0;
    while(q!=NULL)
    {
            q=q->next;
            c++;
    }
    return (c);
}
void addbeg(int ele,node *q)
{
    p=(node *)malloc(sizeof(node));
    p->info=ele;
    p->next=q;
```


2 – Vaishalinagar
Nr. Amrapali Under Bridge
Raiya Road
Rajkot – 360001
Ph.No–(0281)2440478,2472590

3 – Vaishalinagar
Nr. Amrapali Under Bridge
Raiya Road
Rajkot - 360001
Ph.No–(0281)2471645

```
}
void addafter(int c,int ele,node *q)
{
    node *temp;
    int i;
    for(i=1;i<c;i++)
    {
            q=q->next;
            if(q==NULL)
            {
            printf("\nposition is out of range");
                    return;
            }
    }
    temp=(node *)malloc(sizeof(node));
    temp->info=ele;
    temp->next=q->next;
    q->next=temp;
}
void deleted(int ele,node *q)
{
    node *temp;
    if(q->info==ele)
    {
            p=q->next;
            free(q);
            return;
    }
    while(q->next->next!=NULL)
    {
            if(q->next->info==ele)
            {
                    temp=q->next;
                    q->next=q->next->next;
                    free(temp);
                    return;
```

2 – Vaishalinagar
Nr. Amrapali Under Bridge
Raiya Road
Rajkot – 360001
Ph.No–(0281)2440478,2472590

3 – Vaishalinagar
Nr. Amrapali Under Bridge
Raiya Road
Rajkot - 360001
Ph.No–(0281)2471645

```
            }
            q=q->next;
    }
}
void sort(node *q)
{
    int t;
    node *temp;
    while(q!=NULL)
    {
            temp=q->next;
            while(temp!=NULL)
            {
                    if(q->info > temp->info)
                    {
                            t=q->info;
                    q->info=temp->info;
                            temp->info=t;
                    }
                    temp=temp->next;
            }
            q=q->next;
    }
}
void search(int num,node *q)
{
    while(q!=NULL)
    {
            if(q->info==num)
            {
                    printf("\nSearch success");
printf("\nSearch element %d ",num);
                    return;
            }
            q=q->next;
    }
```

```
    printf("\nSearch unsuccessful");
    getch();
}
```

1 WORD QUESTION ANSWER

SR.NO.	QUESTION	ANSWER
1	A LINEAR COLLECTION OF DATA ELEMENTS WHERE THE LINEAR NODE IS GIVEN BY MEANS OF POINTER IS CALLED?	LINKED LIST
2	IN LINKED LIST EACH NODE CONTAINS A MINIMUM OF TWO FIELDS. ONE FIELD IS DATA FIELD TO STORE THE DATA SECOND FIELD IS?	POINTER TO NODE
3	WHAT KIND OF LINKED LIST IS BEST TO ANSWER QUESTIONS LIKE “WHAT IS THE ITEM AT POSITION N?”	ARRAY IMPLEMENTATION OF LINKED LIST
4	LINKED LISTS ARE NOT SUITABLE FOR THE IMPLEMENTATION OF _____	BINARY SEARCH
5	LINKED LIST IS CONSIDERED AS AN EXAMPLE OF _____ TYPE OF MEMORY ALLOCATION.	DYNAMIC
6	IN LINKED LIST IMPLEMENTATION, A NODE CARRIES INFORMATION REGARDING _____	LINK
7	LINKED LIST DATA STRUCTURE OFFERS CONSIDERABLE SAVING IN _____	SPACE UTILIZATION AND COMPUTATIONAL TIME

2 – Vaishalinagar
Nr. Amrapali Under Bridge
Raiya Road
Rajkot – 360001
Ph.No–(0281)2440478,2472590

3 – Vaishalinagar
Nr. Amrapali Under Bridge
Raiya Road
Rajkot - 360001
Ph.No–(0281)2471645

- **Doubly Linked List:**

Program

```
#include<stdio.h>
#include<conio.h>
struct list
{
   int info;
   struct list *next,*prev;
};
typedef struct list node;
node *p;
void create(int,node *);
void display(node *);
void addbeg(int,node *);
void addafter(int,int,node *);
void deleted(int,node *);
int count(node *);
void sort(node *);
void search(int,node *);
void main()
{
   p=NULL;
   clrscr();
   create(10,p);
   create(20,p);
   create(30,p);
   display(p);
   addbeg(0,p);
   printf("\n\n");
   addafter(2,500,p);
   deleted(20,p);
   display(p);
   sort(p);
```

```
    printf("\n\n");
    display(p);
    search(15,p);
    getch();

}
void create(int ele,node *q)
{
    node *temp;
    if(q==NULL)
    {
            p=(node *)malloc(sizeof(node));
            p->prev=NULL;
            p->info=ele;
            p->next=NULL;
    }
    else
    {
            while(q->next!=NULL)
            {
                    q=q->next;
            }
            temp=(node *)malloc(sizeof(node));
            temp->next=NULL;
            temp->info=ele;
            temp->prev=q;
            q->next=temp;
    }
}
void display(node *q)
{
    while(q!=NULL)
    {
            printf("\nElement is %d",q->info);
            q=q->next;
    }
```

```
}
int count(node *q)
{
    int c=0;
    while(q!=NULL)
    {
            q=q->next;
            c++;
    }
    return (c);
}
void addbeg(int ele,node *q)
{
    p=(node *)malloc(sizeof(node));
    p->prev=NULL;
    p->info=ele;
    p->next=q;
    q->prev=p;
}
void addafter(int c,int ele,node *q)
{
    node *temp;
    int i;
    for(i=1;i<c;i++)
    {
            q=q->next;
            if(q==NULL)
            {
            printf("\nposition is out of range");
                    return;
            }
    }
    temp=(node *)malloc(sizeof(node));
    temp->prev=q;
    temp->next=q->next;
    temp->info=ele;
```

2 – Vaishalinagar
Nr. Amrapali Under Bridge
Raiya Road
Rajkot – 360001
Ph.No–(0281)2440478,2472590

3 – Vaishalinagar
Nr. Amrapali Under Bridge
Raiya Road
Rajkot - 360001
Ph.No–(0281)2471645

```
    temp->next->prev=temp;
    q->next=temp;
    return;
}
void deleted(int ele,node *q)
{
    node *temp;
    if(q->info==ele)
    {
            p=q->next;
            q->next->prev=NULL;
            free(q);
            return;
    }
    while(q->next->next!=NULL)
    {
            if(q->next->info==ele)
            {
                    temp=q->next;
                    q->next=q->next->next;
                    q->next->prev=temp->prev;
                    free(temp);
                    return;
            }
            q=q->next;
    }
}
void sort(node *q)
{
    int t;
    node *temp;
    while(q!=NULL)
    {
            temp=q->next;
            while(temp!=NULL)
            {
```

2 – Vaishalinagar
Nr. Amrapali Under Bridge
Raiya Road
Rajkot – 360001
Ph.No–(0281)2440478,2472590
3 – Vaishalinagar
Nr. Amrapali Under Bridge
Raiya Road
Rajkot - 360001
Ph.No–(0281)2471645

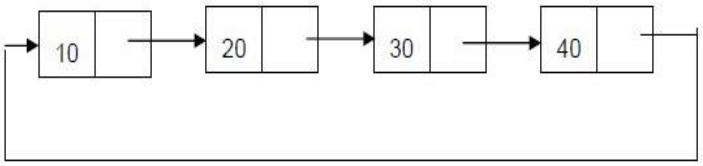
```
                    if(q->info>temp->info)
                    {
                        t=q->info;
                    q->info=temp->info;
                        temp->info=t;
                    }
                    temp=temp->next;
                }
                q=q->next;
    }
}
void search(int num,node *q)
{
    while(q!=NULL)
    {
            if(q->info==num)
            {
                    printf("\nSearch success");
printf("\nSearch element %d ",num);
                    return;
            }
            q=q->next;
    }
    printf("\nSearch unsuccessful");
    getch();
}
```

1 WORD QUESTION ANSWER

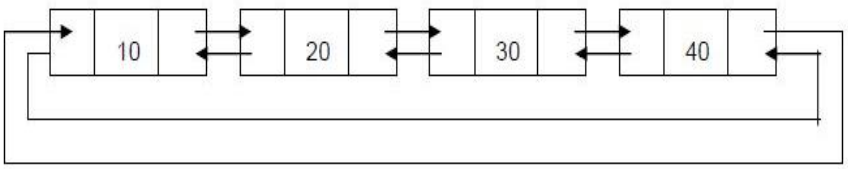
SR.NO.	QUESTION	ANSWER
1	HOW DO YOU CALCULATE THE POINTER DIFFERENCE IN A MEMORY EFFICIENT DOUBLE LINKED LIST?	POINTER TO PREVIOUS NODE XOR POINTER TO NEXT NODE
2	WHAT IS THE WORST CASE TIME COMPLEXITY OF INSERTING A NODE IN A DOUBLY LINKED LIST?	O(N)

- **Circular Singly Linked List:**

  - The linked list where the last node points the header node is called circular linked list.



Circular singly linked list



Circular doubly linked list

2 – Vaishalinagar
Nr. Amrapali Under Bridge
Raiya Road
Rajkot – 360001
Ph.No–(0281)2440478,2472590

3 – Vaishalinagar
Nr. Amrapali Under Bridge
Raiya Road
Rajkot - 360001
Ph.No–(0281)2471645

Program

```
#include<stdio.h>
#include<conio.h>
struct list
{
   int info;
   struct list *next;
};
typedef struct list node;
node *p;
void create(int,node *);
void display(node *);
void addbeg(int,node *);
void addafter(int,int,node *);
void deleted(int,node *);
int count(node *);
void sort(node *);
void search(int,node *);
void main()
{
     p=NULL;
   clrscr();
   create(10,p);
   create(11,p);
   create(12,p);
   display(p);
   printf("\n\n");
   addbeg(0,p);
   addafter(3,12,p);
   deleted(11,p);
   sort(p);
   search(5,p);
   display(p);
   getch();

}
```

2 – Vaishalinagar
Nr. Amrapali Under Bridge
Raiya Road
Rajkot – 360001
Ph.No–(0281)2440478,2472590

3 – Vaishalinagar
Nr. Amrapali Under Bridge
Raiya Road
Rajkot - 360001
Ph.No–(0281)2471645

```
void create(int ele,node *q)
{
   if(q==NULL)
   {
         p=(node *)malloc(sizeof(node));
         p->info=ele;
         p->next=p;
   }
   else
   {
         while(q->next!=p)
         {
               q=q->next;
         }
         q->next=(node *)malloc(sizeof(node));
         q->next->info=ele;
         q->next->next=p;
   }
}
void display(node *q)
{
   do
   {
         printf("\nElement is %d",q->info);
         q=q->next;
   }while(q!=p);
}
int count(node *q)
{
   int c=0;
   do
   {
         q=q->next;
         c++;
   }while(q!=p);
   return (c);
```

```
}
void addbeg(int ele,node *q)
{
   node *temp;
   temp=(node *)malloc(sizeof(node));
   temp->info=ele;
   temp->next=q;
   while(q->next!=p)
   {
         q=q->next;
   }
   q->next=temp;
   p=temp;
}
void addafter(int c,int ele,node *q)//pos is same as c var.
{
   node *temp;
   int i;
   for(i=1;i<c;i++)
   {
         q=q->next;
         if(q==p)
         {
         printf("\nposition is out of range");
               return;
         }
   }
   temp=(node *)malloc(sizeof(node));
   temp->info=ele;
   temp->next=q->next;
   q->next=temp;
}
void deleted(int ele,node *q)
{
   node *temp;
   if(q->info==ele)
```

2 – Vaishalinagar
Nr. Amrapali Under Bridge
Raiya Road
Rajkot – 360001
Ph.No–(0281)2440478,2472590

3 – Vaishalinagar
Nr. Amrapali Under Bridge
Raiya Road
Rajkot - 360001
Ph.No–(0281)2471645

```
    {
        do
        {
            q=q->next;
        }while(q->next!=p);
        q->next=p;
        p=p->next;
        return;
    }
    while(q->next->next!=p)
    {
        if(q->next->info==ele)
        {
            temp=q->next;
            q->next=temp->next;
            free(temp);
            return;
        }
        q=q->next;
    }
}
void sort(node *q)
{
    int t;
    node *temp;
    do
    {
        temp=q->next;
        while(temp!=p)
        {
            if(q->info>temp->info)
            {
                t=q->info;
            q->info=temp->info;
                temp->info=t;
            }
```

```
                temp=temp->next;
            }
            q=q->next;
        }while(q!=p);
    }
    void search(int num,node *q)
    {
        do
        {
            if(q->info==num)
            {
                printf("\nSearch success");
    printf("\nSearch element %d",num);
                return;
            }
            q=q->next;
        }while(q!=p);
        printf("\nSearch unsuccessful");
        getch();
    }
```

- **Circular Doubly Linked List:**

Program

```
#include<stdio.h>
#include<conio.h>
struct list
{
    int info;
    struct list *next,*prev;
};
typedef struct list node;
node *p;
void create(int,node *);
void display(node *);
void addbeg(int,node *);
```

```
void addafter(int,int,node *);
void deleted(int,node *);
int count(node *);
void sort(node *);
void search(int,node *);
void main()
{
   p=NULL;
   clrscr();
   create(10,p);
   create(5,p);
   create(15,p);
   addbeg(0,p);
//search(5,p);
//sort(p);
//deleted(15,p);
   display(p);
   getch();
}
void create(int ele,node *q)
{
   node *temp;
   if(q==NULL)
   {
         p=(node *)malloc(sizeof(node));
         p->prev=p;
         p->info=ele;
         p->next=p;
   }
   else
   {
         while(q->next!=p)
         {
               q=q->next;
         }
         temp=(node *)malloc(sizeof(node));
```

2 – Vaishalinagar
Nr. Amrapali Under Bridge
Raiya Road
Rajkot – 360001
Ph.No–(0281)2440478,2472590
3 – Vaishalinagar
Nr. Amrapali Under Bridge
Raiya Road
Rajkot - 360001
Ph.No–(0281)2471645

```
            temp->next=p;
            temp->info=ele;
            temp->prev=q;
            q->next=temp;
    }
}
void display(node *q)
{
    do
    {
            printf("\nElement is %d",q->info);
            q=q->next;
    }while(q!=p);
}
int count(node *q)
{
    int c=0;
    do
    {
            q=q->next;
            c++;
    }while(q!=p);
    return (c);
}
void addbeg(int ele,node *q)
{
    node *temp;
    while(q->next!=p)
    {
            q=q->next;
    }
    temp=(node *)malloc(sizeof(node));
    temp->prev=q;
    temp->info=ele;
    temp->next=p;
    p->prev=temp;
```


2 – Vaishalinagar
Nr. Amrapali Under Bridge
Raiya Road
Rajkot – 360001
Ph.No–(0281)2440478,2472590

3 – Vaishalinagar
Nr. Amrapali Under Bridge
Raiya Road
Rajkot - 360001
Ph.No–(0281)2471645

```
    q->next=temp;
    p=temp;

}
void addafter(int c,int ele,node *q)
{
    node *temp;
    int i;
    for(i=1;i<c;i++)
    {
            q=q->next;
            if(q==p)
            {
                    printf("\nposition is out of range");
                    return;
            }
    }
    temp=(node *)malloc(sizeof(node));
    temp->prev=q;
    temp->next=q->next;
    temp->info=ele;
    temp->next->prev=temp;
    q->next=temp;
    return;
}
void deleted(int ele,node *q)
{
    node *temp;
    if(q->info==ele)
    {
            temp=q;
            p=temp->next;
            while(q->next!=p)
            {
                    q=q->next;
            }
```


2 – Vaishalinagar
Nr. Amrapali Under Bridge
Raiya Road
Rajkot – 360001
Ph.No–(0281)2440478,2472590

3 – Vaishalinagar
Nr. Amrapali Under Bridge
Raiya Road
Rajkot - 360001
Ph.No–(0281)2471645

```
            p->prev=q;
            return;

    }
    do
    {
            if(q->next->info==ele)
            {
                    temp=q->next;
                    q->next=temp->next;
                    temp->next->prev=q;
                  // free(temp);
                    return;
            }
            q=q->next;
    }while(q!=p);
}
void sort(node *q)
{
    int t;
    node *temp;
    do
    {
            temp=q->next;
            while(temp!=p)
            {
                    if(q->info>temp->info)
                    {
                            t=q->info;
                            q->info=temp->info;
                            temp->info=t;
                    }
                    temp=temp->next;
            }
            q=q->next;
    }while(q!=p);
```

```
}
void search(int num,node *q)
{
   do
   {
         if(q->info==num)
         {
               printf("\nSearch success");
               printf("\nSearch element %d",num);
               return;
         }
         q=q->next;
   }while(q!=p);
   printf("\nSearch unsuccessful");
   getch();

}
```

NOTE: All the algorithms will be same only conditions will be changed.

1 WORD QUESTION ANSWER

SR.NO.	QUESTION	ANSWER
1	WHAT DIFFERENTIATES A CIRCULAR LINKED LIST FROM A NORMAL LINKED LIST?	YOU MAY OR MAY NOT HAVE THE 'NEXT' POINTER POINT TO NULL IN A CIRCULAR LINKED LIST
2	WHAT IS THE TIME COMPLEXITY OF SEARCHING FOR AN ELEMENT IN A CIRCULAR LINKED LIST?	O(N)
3	WHICH APPLICATION MAKES USE OF A CIRCULAR LINKED LIST?	ALLOCATING CPU TO RESOURCES

2 – Vaishalinagar
Nr. Amrapali Under Bridge
Raiya Road
Rajkot – 360001
Ph.No–(0281)2440478,2472590

3 – Vaishalinagar
Nr. Amrapali Under Bridge
Raiya Road
Rajkot - 360001
Ph.No–(0281)2471645

SUMMARY

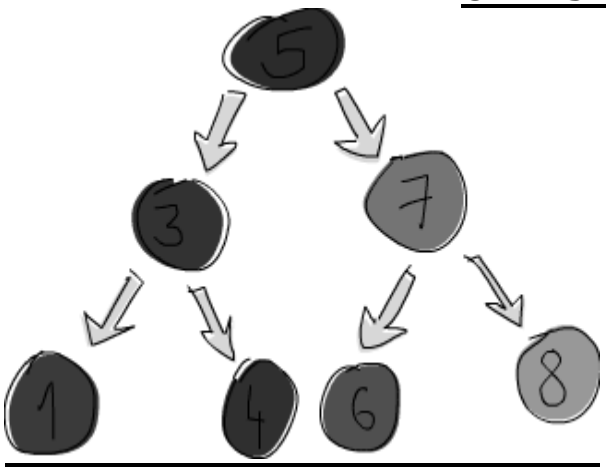
- ✓ Linked list is a **non linear** data structure which defines as a collection of nodes.
- ✓ A main limitation of an array (used in stack and queue) is that elements must be stored in ho-mogeneous manner and fixed amount of data is stored. Where as in linked list representation this limitation is overcome with the use of node.
- ✓ There are mainly five types of linked list which listed below:
  - Singly linked list contains information part and address of next node. In this last node contains NULL value in address part.
  - Doubly linked list contains information part and address of next node as well address of previous node. In this last node contains NULL value, in the next node address part.
  - Circular doubly linked list Is same like as doubly linked list but in the last node next address contains the address of first node. And first node previous part contains the address of last node.
  - Circular doubly linked list is same like as doubly linked list but in the last node next address contains the address of first node. And first node previous part contains the address of last node.
  - Header linked list contains a special type of information that how many number of nodes are there in linked list. There are two types of header linked lists:
- ✓ Grounded header linked list
- ✓ Circular header linked list
- ✓ Operations performed in a linked list are as below:

✓ Creation	✓ Insertion	✓ Deletion	✓ Find	✓ Append
✓ Sorting	✓ Display/view	✓ Count	✓	✓

2 – Vaishalinagar
Nr. Amrapali Under Bridge
Raiya Road
Rajkot – 360001
Ph.No–(0281)2440478,2472590

3 – Vaishalinagar
Nr. Amrapali Under Bridge
Raiya Road
Rajkot - 360001
Ph.No–(0281)2471645

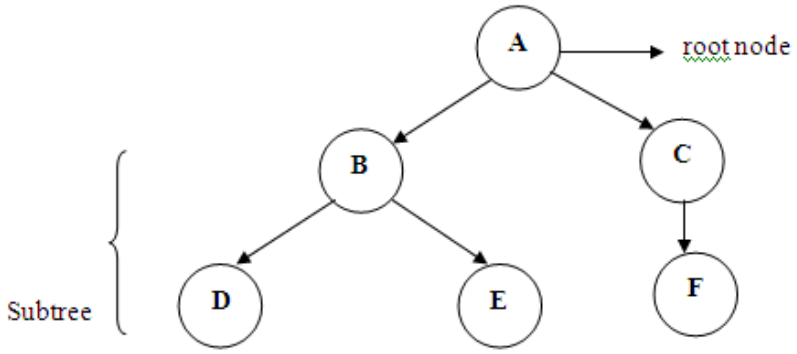
UNIT-5 - TREE



- Introduction
  - Objectives
  - Basic terminology
  - Properties of a tree
- Binary trees
  - Properties of binary trees
  - Implementation
  - Traversals of a binary tree
    - In order traversal
    - Post order traversal
    - Preorder traversal
- Binary search trees (bst)
  - Insertion in bst
  - Deletion of a node
  - Search for a key in bst
  - Height balanced tree
  - b-tree

Insertion & Deletion

1) What is tree? What is root of the node?
   Ans:
   - Tree is a non-linear data structure which is a set of one or more nodes such that:
     1) **There is specially designated node known as root of the tree**.
     2) The remaining nodes are divided into n disjoint se of nodes T1, T2…..Tn, each of which is a tree.



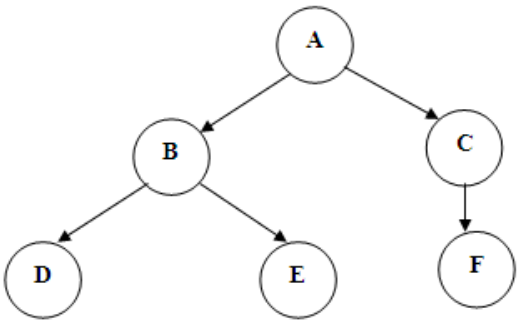
- The above figure shows tree which is divided into two disjoint sets.
  {B,D,E} and {C,F}

2) What are leaf and non-leaf nodes?

Ans:

- **Leaf nodes are the nodes that do not have any children.**
- Non-leaf nodes are the nodes that have children.

Example:



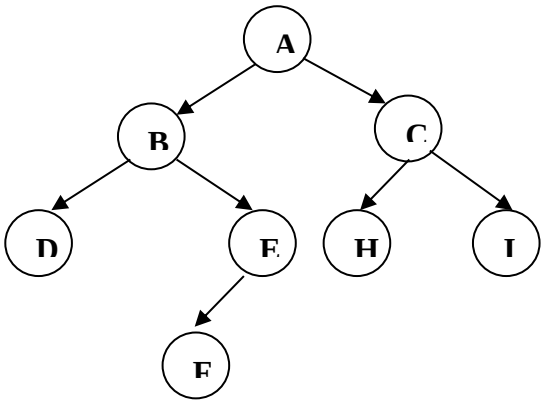
- In the above example, nodes D, E and F are leaf nodes and nodes A,B and c are non-leaf nodes.

3) What are ancestor and descendant?

Ans:

- Every node which is **parent to leaf and non-leaf nodes is known as ancestor**.

Example:



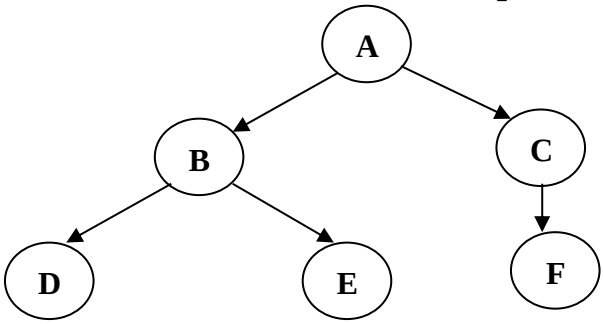
- In the above example, A is known as ancestor of E.
- All the nodes you can reach from the given node is known as descendant.
- In the above example, node H is known as descendant of node C.

4) What is the meaning of Siblings?

Ans:-

- Siblings are the **nodes that share the same parent node.**

Example:



- In the above example, nodes B and C are known as siblings.

2 – Vaishalinagar
Nr. Amrapali Under Bridge
Raiya Road
Rajkot – 360001
Ph.No–(0281)2440478,2472590

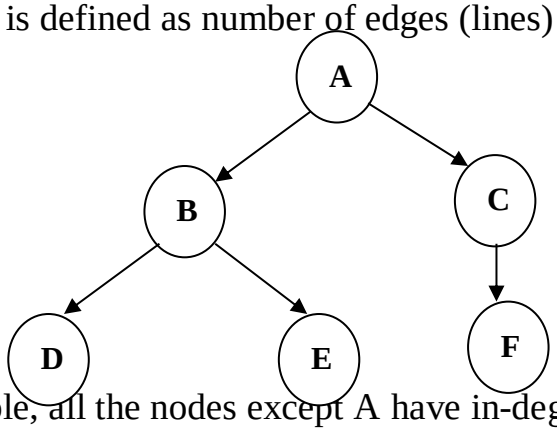
3 – Vaishalinagar
Nr. Amrapali Under Bridge
Raiya Road
Rajkot - 360001
Ph.No–(0281)2471645

5) What is the meaning of in-degree of vertex?

Ans:

- In-degree of vertex is defined as number of edges (lines) arriving at node.

Example:



- In the above example, all the nodes except A have in-degree 1.

Note: Root is the only tree which have in-degree 0.

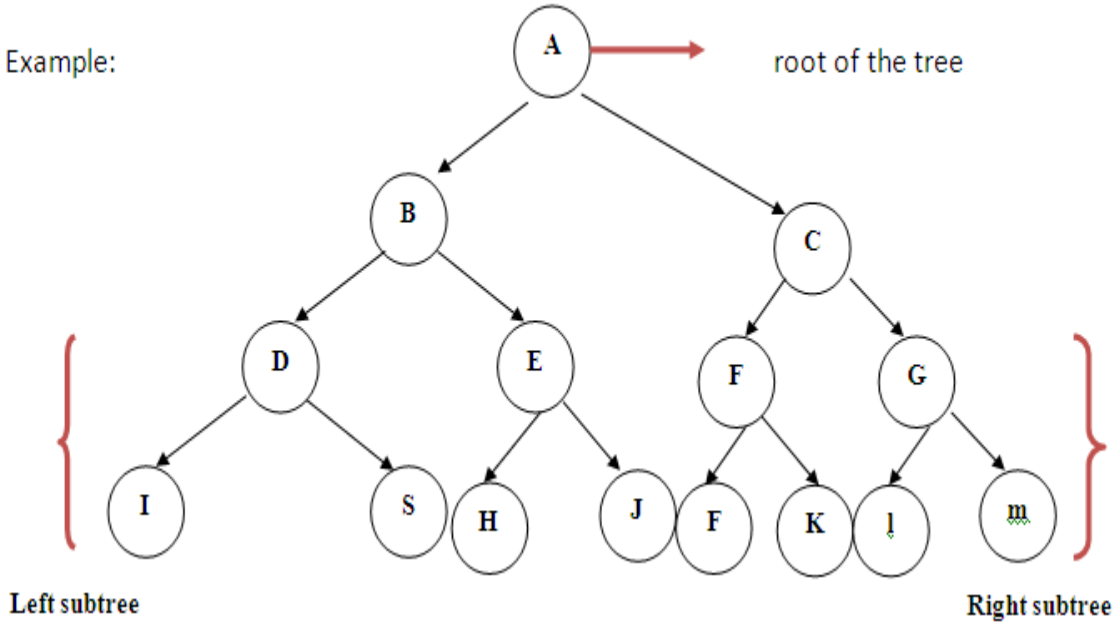
5) What is Binary tree?

Ans:

- Binary tree is the tree in which every node has maximum of two children.

OR

- Binary tree is the set of elements that is either empty or is divided into three disjoint subsets.
- The first subset contains single element called root of the tree.
- The other two subsets are binary tree themselves which are known as left and right sub tree. It can be empty.



- So in the above example, the depth of tree is 2.

6) Explain types of binary tree.

Ans:

There are 3 types of binary tree.

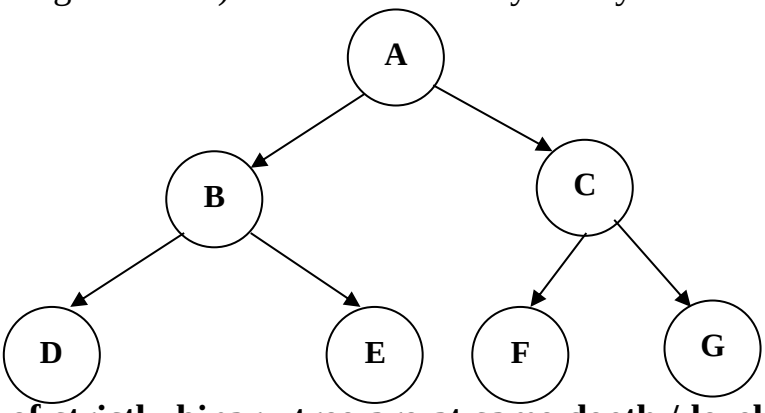
1) Strictly binary tree
2) Complete binary tree
3) Almost complete binary tree

2 – Vaishalinagar
Nr. Amrapali Under Bridge
Raiya Road
Rajkot – 360001
Ph.No–(0281)2440478,2472590

3 – Vaishalinagar
Nr. Amrapali Under Bridge
Raiya Road
Rajkot - 360001
Ph.No–(0281)2471645

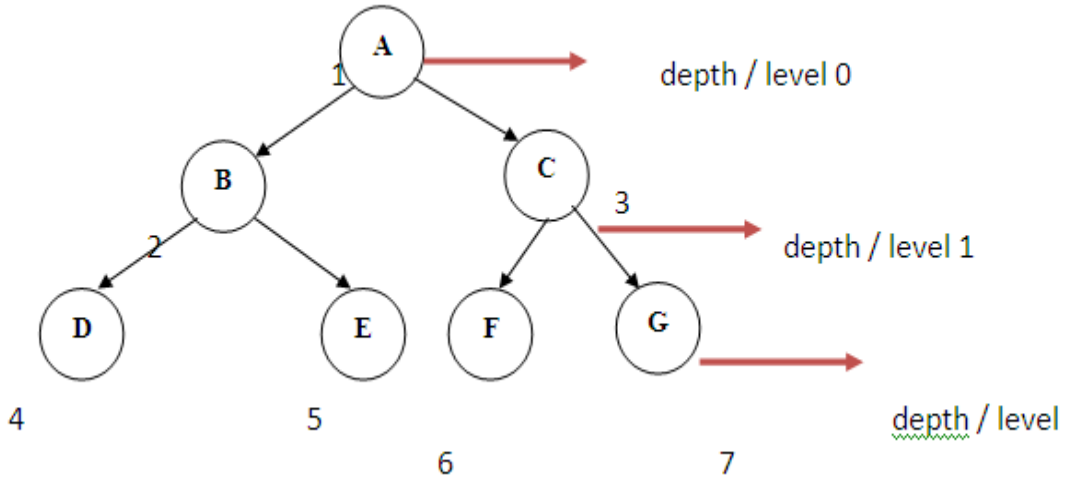
1) Strictly binary tree:-
   - **Every non-leaf node in binary tree has exactly two non-empty children** (left and right subtree) is known as strictly binary tree.

Example:



2) Complete binary tree:-
   - If **all the leaves of strictly binary tree are at same depth / level 'd' is known as complete binary tree.**

Example:



2 – Vaishalinagar
Nr. Amrapali Under Bridge
Raiya Road
Rajkot – 360001
Ph.No–(0281)2440478,2472590

3 – Vaishalinagar
Nr. Amrapali Under Bridge
Raiya Road
Rajkot - 360001
Ph.No–(0281)2471645

Advantage:

- The main advantage of complete binary tree is that we can easily find out parent or root node of any node and left and right sub tree of any node.

Example:

Suppose we want to find the parent or root node of node E then we have to write

floor(N/2)
=floor (5/2)
=floor(2.5)
=2
=B

- So, Parent node of node E is node B.
- Now, suppose that we want to find left and right sub tree of node B, so we have to write

Left Sub tree	Right Sub tree
2N	2N+1
=2*2	=2*2+1
=4	=5
=D	=E

- So, left and right sub tree of node B is D and E

- **Note: Total no. of nodes in the complete binary tree is calculated as $t=2^{(d+1)}-1$**

3) Almost complete binary tree:

- A binary tree of **depth / level d is known as almost complete binary tree if each node in tree is at level d or d-1 and for each node in tree is at level d or d-1** and for any node in a tree with right descendant at depth d, all the left descendant of this node are also at level d.

7) Explain difference between binary tree and strictly binary tree.

Binary Tree	*Strictly Binary Tree*
1) In binary tree, each node have maximum two nodes	1) In strictly binary tree, each node Has exactly two nodes.

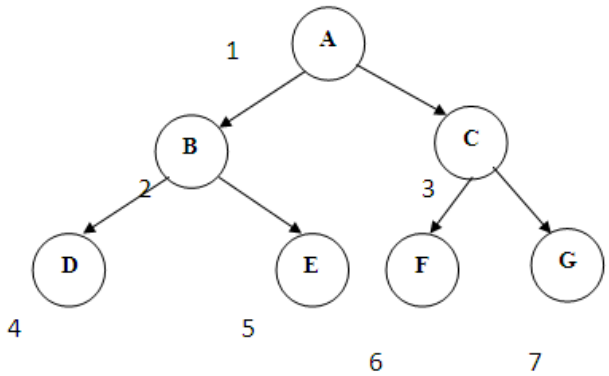
8) Explain implementation of binary tree.
Ans:

- Binary tree can be implemented by two methods:-
  1) Sequential representation (Linear)
  2) Linked List representation (Non-linear)

1) Sequential Representation:-

- In sequential representation, **nodes are stored in one dimension array**. Also assign level number to every node in tree.
- We can assign node numbers in such a way that root is assigned the number 1. Then, left sub tree must be assigned 2p+1 and right sub tree must be assigned 2p+2.

Example:

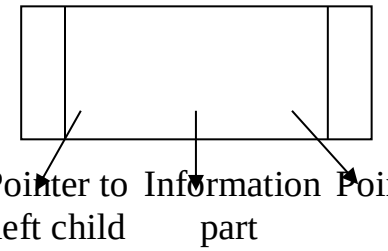


The above tree is represented in the array as:

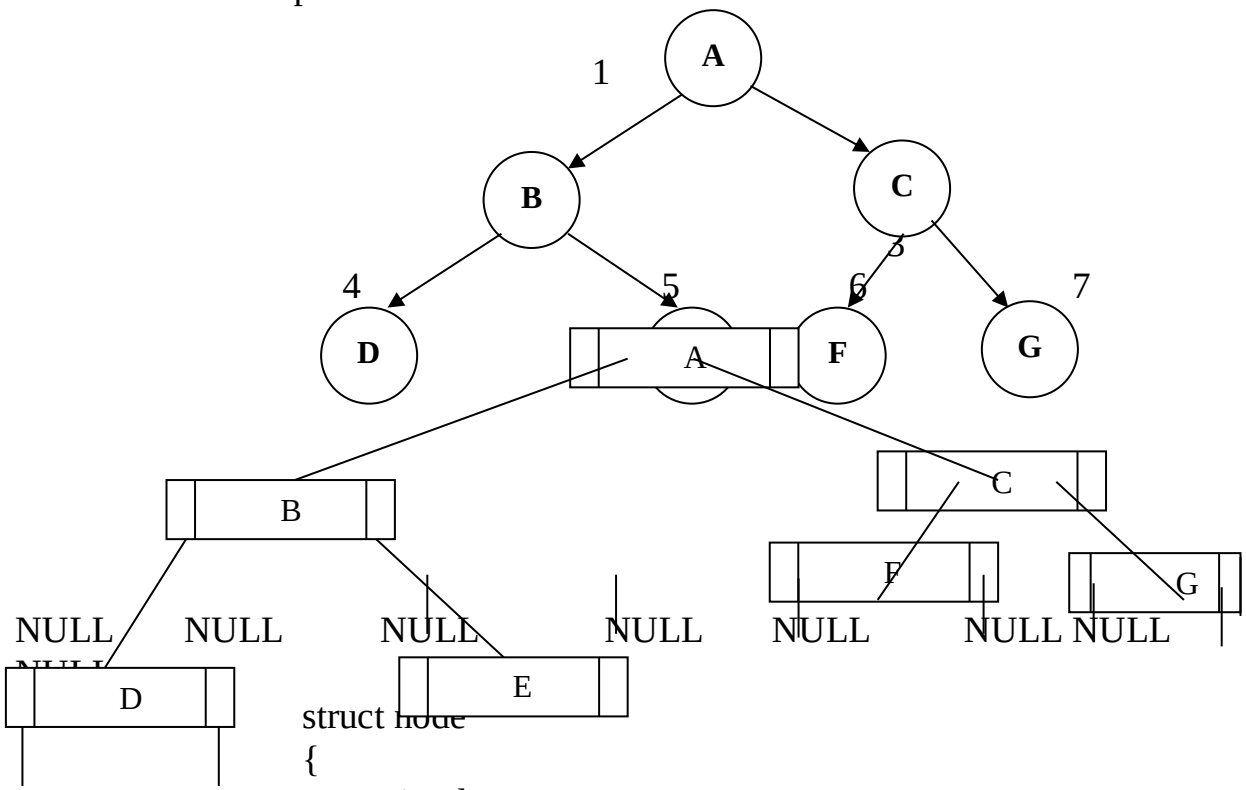
0	1	2	3	4	5	6
A	B	C	D	E	F	G

2) Linked List Representation:-

- In linked list representation, **every node is stored separately and keeps the address of other nodes.**



Example:-




```
struct node
{
        int data;
        struct node *lefttree;
        struct node *righttree;
};
```

9)Explain the steps for creating the binary tree.

Ans:

- The rule or method of creating the binary tree is as follows:

2 – Vaishalinagar
Nr. Amrapali Under Bridge
Raiya Road
Rajkot – 360001
Ph.No–(0281)2440478,2472590

3 – Vaishalinagar
Nr. Amrapali Under Bridge
Raiya Road
Rajkot - 360001
Ph.No–(0281)2471645

Step 1: Compare with the root element.

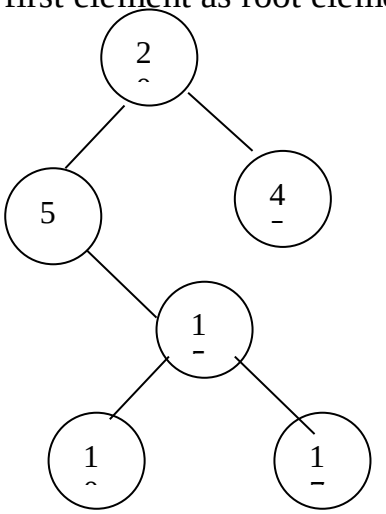
Step 2: If the element is less than root, move left side

Step 3: If the element is greater than root, move right side

Example:
Create the binary tree for following example

20 5 15 17 45 10

Taking first element as root element



2 – Vaishalinagar
Nr. Amrapali Under Bridge
Raiya Road
Rajkot – 360001
Ph.No–(0281)2440478,2472590

3 – Vaishalinagar
Nr. Amrapali Under Bridge
Raiya Road
Rajkot - 360001
Ph.No–(0281)2471645

1 WORD QUESTION ANSWER

SR.NO.	QUESTION	ANSWER
1	THE NUMBER OF EDGES FROM THE ROOT TO THE NODE IS CALLED _____ OF THE TREE.	DEPTH
2	THE NUMBER OF EDGES FROM THE NODE TO THE DEEPEST LEAF IS CALLED _____ OF THE TREE.	HEIGHT
3	WHAT IS A FULL BINARY TREE?	EACH NODE HAS EXACTLY ZERO OR TWO CHILDREN
4	WHAT IS THE AVERAGE CASE TIME COMPLEXITY FOR FINDING THE HEIGHT OF THE BINARY TREE?	H = O(LOG N)
5	HOW MANY CHILDREN DOES A BINARY TREE HAVE?	0 OR 1 OR 2
6	LEVEL ORDER TRAVERSAL OF A TREE IS FORMED WITH THE HELP OF	BREADTH FIRST SEARCH
7	WHAT IS THE SPECIALITY ABOUT THE INORDER TRAVERSAL OF A BINARY SEARCH TREE?	IT TRAVERSES IN AN INCREASING ORDER
8	WHAT ARE THE WORST CASE AND AVERAGE CASE COMPLEXITIES OF A BINARY SEARCH TREE?	O(N), O(LOGN)
9	WHAT WILL BE THE HEIGHT OF A BALANCED FULL BINARY TREE WITH 8 LEAVES?	4
10	THE BALANCE FACTOR OF A NODE IN A BINARY TREE IS DEFINED AS _____	HEIGHT OF LEFT SUBTREE MINUS HEIGHT OF RIGHT SUBTREE
11	A BINARY TREE IS BALANCED IF THE DIFFERENCE BETWEEN LEFT AND RIGHT SUBTREE OF EVERY NODE IS	1

	NOT MORE THAN ____	
12	WHICH TREE DATA STRUCTURES IS NOT A BALANCED BINARY TREE?	B-TREE
13	BALANCED BINARY TREE WITH N ITEMS ALLOWS THE LOOKUP OF AN ITEM IN ____ WORST-CASE TIME.	O(LOG N)
14	WHICH DATA STRUCTURES CAN BE EFFICIENTLY IMPLEMENTED USING HEIGHT BALANCED BINARY SEARCH TREE?	SETS AND PRIORITY QUEUE
15	ANOTHER NAME FOR THE DIRECTED GRAPH IS___	DIGRAPH
16	THE OPERATION OF PROCESSING EACH ELEMENT IN THE LIST IS KNOWN AS ……	TRAVERSAL
17	GRAPH G IS …….. IF FOR ANY PAIR U, V OF NODES IN G THERE IS A PATH FROM U TO V OR PATH FROM V TO U.	UNLITERALLY CONNECTED
18	A BINARY TREE WHOSE EVERY NODE HAS EITHER ZERO OR TWO CHILDREN IS CALLED …….	EXTENDED BINARY TREE
19	A TERMINAL NODE IN A BINARY TREE IS CALLED__	LEAF
20	IN EXTENDED-BINARY TREE NODES WITH 2 CHILDREN ARE CALLED ……..	INTERNAL NODE

10) What is tree traversal? Explain tree traversal techniques with example.
Ans:

- **Tree traversal is the method or technique to display data in binary tree.**
- In traversing method, t**ree is processed in such a way that each node is visited only once.**
- There are 3 main tree traversal techniques:
  - **Preorder traversal**
  - **Inorder traversal**
  - **Postorder traversal**

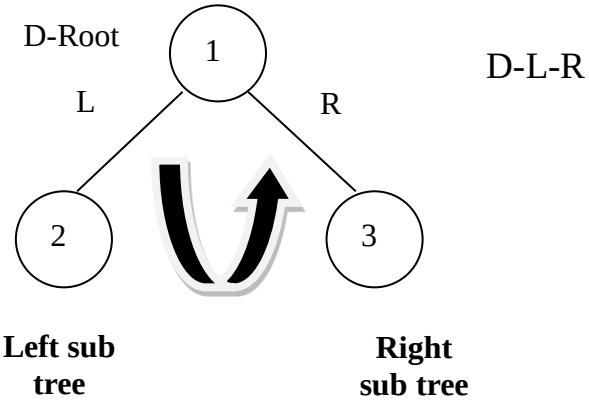
2 – Vaishalinagar
Nr. Amrapali Under Bridge
Raiya Road
Rajkot – 360001
Ph.No–(0281)2440478,2472590

3 – Vaishalinagar
Nr. Amrapali Under Bridge
Raiya Road
Rajkot - 360001
Ph.No–(0281)2471645

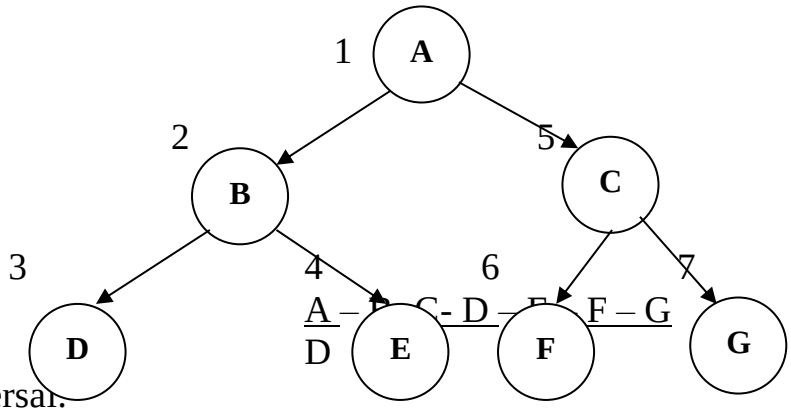
1) Preorder traversal:-

- It follows three rules:
  If the tree is not empty then:-
  1) **Visit the root node**
  2) **Traverse left sub tree-recursively**
  3) **Traverse right sub tree-recursively**

  In this, (D) is used for root node, (L) is used for left sub tree and (R) is used for right sub tree.



Example:

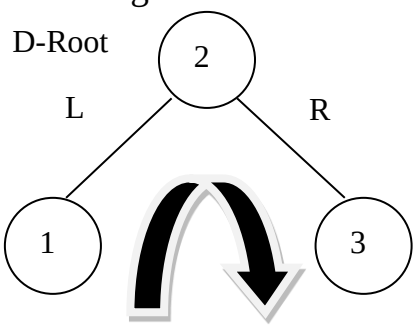


A – B- C- D – E – F – G
D

2) Inorder traversal:

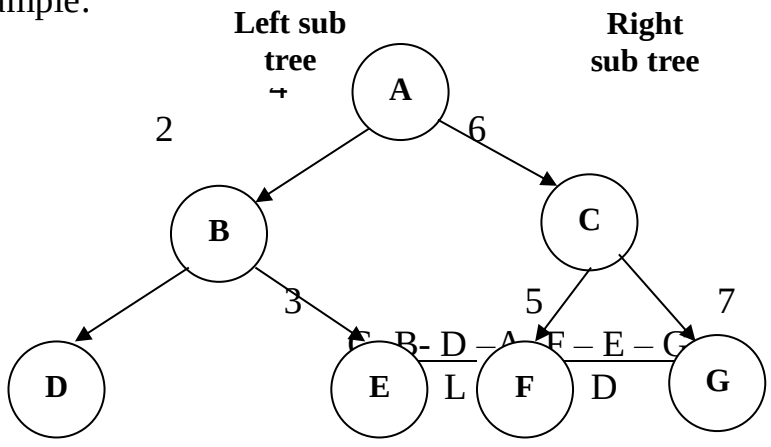
- It follows three rules:
  If the tree is not empty then:-
  1) **Traverse left sub tree-recursively**
  2) **Visit root node**
  4) **Traverse right sub tree-recursively**

  In this, (D) is used for root node, (L) is used for left sub tree and (R) is used for right sub tree.



L-D-R

Example:



D – B- D – A – F – E – G
L D

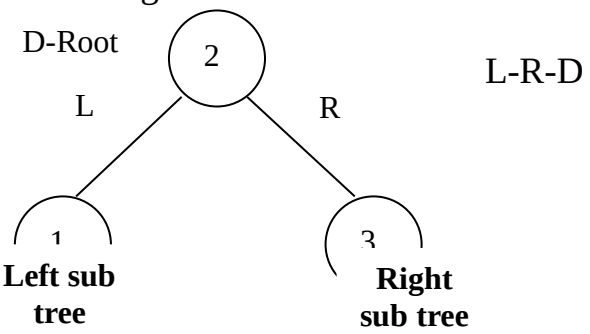
2 – Vaishalinagar
Nr. Amrapali Under Bridge
Raiya Road
Rajkot – 360001
Ph.No–(0281)2440478,2472590

3 – Vaishalinagar
Nr. Amrapali Under Bridge
Raiya Road
Rajkot - 360001
Ph.No–(0281)2471645

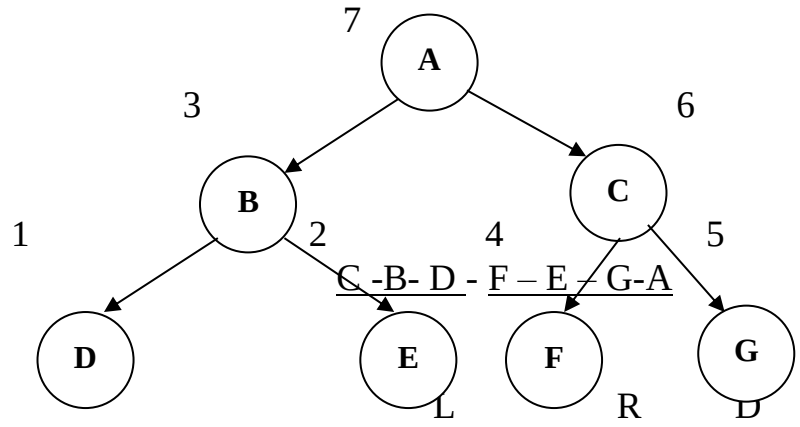
3) Postorder traversal:-

- It follows three rules:
  If the tree is not empty then:-
  1) **Traverse left sub tree-recursively**
  2) **Traverse right sub tree-recursively**
  3) **Visit root node.**

  In this, (D) is used for root node, (L) is used for left sub tree and (R) is used for right sub tree.



Example:



1 WORD QUESTION ANSWER

SR.NO.	QUESTION	ANSWER
1	WHAT IS THE TIME COMPLEXITY OF PRE-ORDER TRAVERSAL IN THE ITERATIVE FASHION?	O(N)
2	WHAT IS THE SPACE COMPLEXITY OF THE POST-ORDER TRAVERSAL IN THE RECURSIVE FASHION? (D IS THE TREE DEPTH AND N IS THE NUMBER OF NODES)	O(D)
3	TO OBTAIN A PREFIX EXPRESSION, WHICH OF THE TREE TRAVERSALS IS USED?	PRE-ORDER TRAVERSAL
4	IN POSTORDER TRAVERSAL OF BINARY TREE RIGHT SUBTREE IS TRAVERSED BEFORE VISITING ROOT. (TRUE OR FALSE)	TRUE
5	WHAT IS THE POSSIBLE NUMBER OF BINARY TREES THAT CAN BE CREATED WITH 3 NODES, GIVING THE SEQUENCE N, M, L WHEN TRAVERSED IN POST-ORDER.	5
6	THE POST-ORDER TRAVERSAL OF A	T Q O P S R

	BINARY TREE IS O P Q R S T. THEN POSSIBLE PRE-ORDER TRAVERSAL WILL BE _____	
7	A FULL BINARY TREE CAN BE GENERATED USING _____	POST-ORDER AND PRE-ORDER TRAVERSAL
8	THE MAXIMUM NUMBER OF NODES IN A TREE FOR WHICH POST-ORDER AND PRE-ORDER TRAVERSALS MAY BE EQUAL IS _____	1
9	THE STEPS FOR FINDING POST-ORDER TRAVERSAL ARE TRAVERSE THE RIGHT SUBTREE, TRAVERSE THE LEFT SUBTREE OR VISIT THE CURRENT NODE.(TRUE OR FALSE)	FALSE
10	THE PRE-ORDER AND IN-ORDER ARE TRAVERSALS OF A BINARY TREE ARE T M L N P O Q AND L M N T O P Q. WHICH OF FOLLOWING IS POST-ORDER TRAVERSAL OF THE TREE?	L N M O Q P T
11	FOR A BINARY TREE THE FIRST NODE VISITED IN IN-ORDER AND POST-ORDER TRAVERSAL IS SAME.(TRUE OR FALSE)	FALSE
12	WHAT IS THE SPACE COMPLEXITY OF THE IN-ORDER TRAVERSAL IN THE RECURSIVE FASHION? (D IS THE TREE DEPTH AND N IS THE NUMBER OF NODES)	O(D)
13	WHAT IS THE TIME COMPLEXITY OF LEVEL ORDER TRAVERSAL?	O(N)
14	WHICH GRAPH TRAVERSALS CLOSELY IMITATES LEVEL ORDER TRAVERSAL OF A BINARY TREE?	BREADTH FIRST SEARCH
15	IN A BINARY SEARCH TREE, WHICH TRAVERSALS WOULD PRINT THE NUMBERS IN THE ASCENDING ORDER?	IN-ORDER TRAVERSAL

2 – Vaishalinagar
Nr. Amrapali Under Bridge
Raiya Road
Rajkot – 360001
Ph.No–(0281)2440478,2472590

3 – Vaishalinagar
Nr. Amrapali Under Bridge
Raiya Road
Rajkot - 360001
Ph.No–(0281)2471645

SUMMARY

- ✓ A tree is hierarchical **non-liner** data structure that is used to store information.
- ✓ The tree data structure has the mainly three components: root (for a base), branches (for growth) and leaves (for existence).
- ✓ The **root node** of a tree is the node with no parents. There is at most one root node in a rooted tree.
- ✓ A **leaf node** has no children.
- ✓ A **directed edge** refers to the link from the parent to the child (the arrows in the picture of the tree).
- ✓ The binary tree creation follows a very simple principle – for the new element to be added, compare it with the current element in the tree.
- ✓ If its value is less than the current element in the tree.
- ✓ If its value is less than the current element in the tree then move towards the left side of that element.
- ✓ If its **value is greater than the current element** in the tree then move towards the right side of that element.
- ✓ If there is no sub tree on the left, make our new element as the left child of that current element or else compare it with the existing left child and follow the same rule.
- ✓ Exactly the same hs to be done for the case when our new element is greater than the current element in the tree but this time with the right child.
- ✓ The depth of a node n is the length of the path from the root to the node. The set of all nodes at a given depth is sometimes called a level of the tree. The root node is at depth zero. A tree with only a root node has a height of zero.
- ✓ We can also find out the level of a node in a binary tree. For example, root is at level '0' and level of any other node is one or more than the level of its ancestor.
- ✓ Siblings are nodes that share the same parent node.

2 – Vaishalinagar
Nr. Amrapali Under Bridge
Raiya Road
Rajkot – 360001
Ph.No–(0281)2440478,2472590

3 – Vaishalinagar
Nr. Amrapali Under Bridge
Raiya Road
Rajkot - 360001
Ph.No–(0281)2471645

- ✓ In-degree of a vertex is the number of edges arriving at that vertex.
- ✓ Root is the only node in the tree with in-degree=0.
- ✓ The trees will have their root at the top and leaves at the bottom.
- ✓ The direction from root to the leaves in 'down' and the opposite direction is 'up'.
- ✓ If going from the leaves to the root then it is known as 'climbing' the tree.
- ✓ If going from the root to the leaves then it is known as 'descending' the tree
- ✓ In a non-empty tree the root node is having no incoming edge (link).
- ✓ If there isn't any node in a tree then it is known as NULL tree.
- ✓ **Every nodes in a tree has a maximum of two children are known as binary trees.**
- ✓ Depends on he depth/level of binary tree, it can be classified into 3 types:
  - **Strictly binary tree**
  - **Complete binary tree**
  - **Almost complete binary tree**
- ✓ There are mainly two popular techniques that are used to maintain binary tree into memory.
- ✓ These are:
  - Sequential representation(Linear)
  - Linked list representation(Non-Linear)
- ✓ If we want to display data in a binary tree then we have to use traversal method.
- ✓ In the **traversing mechanism**, tree is processing such a way that, each node is visited only once (this is a procedure by which each node in the tree is processed exactly once).
- ✓ There are mainly three different ways available in a binary tree to visit each and every element of a tree. They are as follows:
  - Preorder traversal
  - Inorder traversal
  - Postorder traversal

2 – Vaishalinagar
Nr. Amrapali Under Bridge
Raiya Road
Rajkot – 360001
Ph.No–(0281)2440478,2472590

3 – Vaishalinagar
Nr. Amrapali Under Bridge
Raiya Road
Rajkot - 360001
Ph.No–(0281)2471645

- ✓ These traversal methods are limited to binary trees only and it's not implemented for any other tree.
- ✓ Here, given below short summary of 3 methods:

✓ **Pre order**	✓ Display info	✓ Go to left	✓ Go to right
✓ **In order**	✓ Go to left	✓ Display info	✓ Go to right