



V2V EDTECH LLP

Online Coaching at an Affordable Price.

OUR SERVICES:

- Diploma in All Branches, All Subjects
- Degree in All Branches, All Subjects
- BSCIT / CS
- Professional Courses



+91 93260 50669



v2vedtech.com



V2V EdTech LLP



v2vedtech





Fy-Diploma (URJA) [LIVE] (Sem 2) only at 4999/- [BUY NOW](#)

Sy-Diploma (UMANG) [LIVE] (Sem 3 + sem 4) : only at 4999/- [BUY NOW](#)

Ty-Diploma (YUKTI) [LIVE] (Sem 3 + sem 4) : only at 4999/- [BUY NOW](#)

All Courses : [CHECK NOW](#) YOUTUBE : [SUBSCRIBE NOW](#) INSTA : [FOLLOW NOW](#)

Download V2V APP on Playstore for more [FREE STUDY MATERIAL](#)

Contact No : 9326050669 / 93268814282

MSBTE PAPERS
2 MARKS QUESTIONS
(WINTER 18)

1. Define the term algorithm

Ans: Algorithm is a stepwise set of instructions written to perform a specific task.

2. List any 4 applications of queue.

Ans: • In computer system to maintain waiting list for single shared resources such as printer, disk, etc.

• It is used as buffers on MP3 players, iPod playlist, etc.

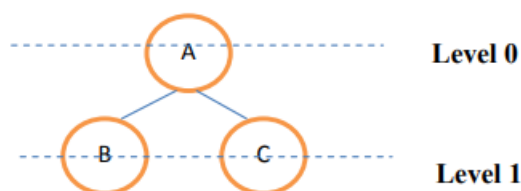
• Used for CPU scheduling in multiprogramming and time sharing systems.

• In real life, Call Center phone systems will use Queues, to hold people calling them in an order, until a service representative is free.

• Handling of interrupts in real-time systems.

3. Describe following terms w.r.to tree: (i) Leaf node (ii) Level of node

Ans: Example:



(i) Leaf node: A node without any child node is called as leaf node.

Nodes B and C are leaf node as shown in above example.

(ii) Level of node: Position of a node in the hierarchy of a tree is called as level of node.

Level of node B is 1 as shown in above example.

4. Differentiate between stack and queue.(Any two points)

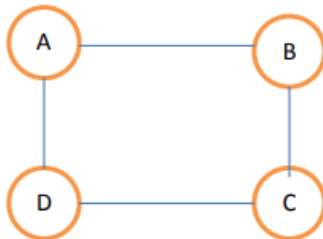
Ans:

Stack	Queue
1. Stack is a data structure in which insertion and deletion operations are performed at same end .	1. Queue is a data structure in which insertion and deletion operations are performed at different ends .
2. In stack an element inserted last is deleted first so it is called Last In First Out list .	2. In Queue an element inserted first is deleted first so it is called First In First Out list .
3. In stack only one pointer is used called as stack top	3. In Queue two pointers are used called as front and rear
4. Example: Stack of books	4. Example: Students standing in a line at fees counter
5. Application: <ul style="list-style-type: none"> • Recursion • Polish notation 	5. Application: <ul style="list-style-type: none"> • In computer system for organizing processes. • In mobile device for sending receiving messages.

5. Describe undirected graph with suitable example.

Ans Undirected graph: A graph in which the edges do not have any direction associated with them is known as undirected graph. In undirected graph, if an edge exists between two nodes A and B then the nodes can traverse from A to B as well as from B to A. Each edge is bidirectional.

Example:



6. Define the terms: Linear data structure and non-linear data structure.

Ans: Linear Data Structure: A data structure in which all data elements are stored in a particular sequence is known as linear data structure.

Example: stack, queue Non-Linear data structure: A data structure in which all data elements are not stored in any particular sequence is known as nonlinear data structure. Example: graph and tree.

7. convert infix expression into prefix expression:

$$(A+B)*(C/G)+F$$

Ans:

Infix expression	Read Character	Stack contents	Prefix expression
$(A+B)*(C/G)+F$	F	-	F
$(A+B)*(C/G)+$	+	+	F
$(A+B)*(C/G)$)	+))	F
$(A+B)*(C/G$	G	+))	GF
$(A+B)*(C/$	/	+)/)	GF
$(A+B)*(C$	C	+)/)	CGF
$(A+B)*$	(+)	/CGF
$(A+B)*$	*	+*)	/CGF
$(A+B)$)	+*)	/CGF
$(A+B$	B	+*)	B/CGF
$(A+$	+	+*)	B/CGF
$(A$	A	+*)	AB/CGF
$($	(+*)	+AB/CGF
			*+AB/CGF
			+*+AB/CGF

(SUMMER-19)

1. List any four operations on data structure.

Ans: Operations on data structure:

- Insertion
- Deletion
- Searching
- Sorting
- Traversing
- Merging

2. Enlist queue operation condition.

Ans: 1. Queue Full

2. Queue Empty

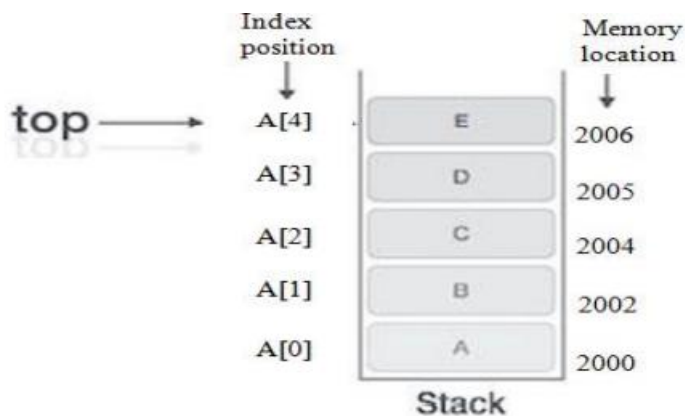
3. Define: (i) Binary tree (ii) Binary search tree

Ans: (i) Binary tree: It is a nonlinear data structure in which each non-leaf node can have maximum two child nodes as left child and right child.

(ii) Binary search tree: It is a nonlinear data structure in which left child of root node is less than root and right child of root node is greater than root.

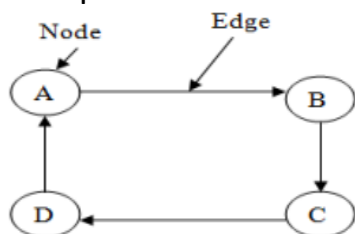
4. Show the memory representation of stack using array with the help of a diagram.

Ans: Consider stack contains five integer elements represented with an array A in which each element occupies 2 bytes memory. Array starts with base address of 2000



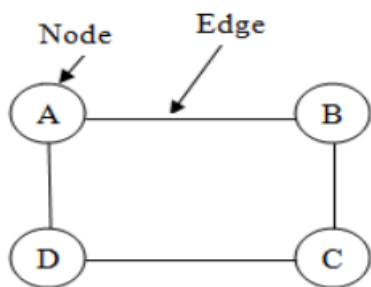
5. Define given two types of graph and give example. (i) Direct graph (ii) Undirected graph
 Ans: (i) Direct graph: A graph in which direction is associated with each edge is known as directed graph.

Example:



(ii) Undirected graph: A graph in which the edges do not have any direction associated with them is known as undirected graph.

Example:-



6. Differentiate between linear and non-linear data structures on any two parameters.
 Ans:

Sr. No.	Linear data structure	Non-linear data structure
1	A data structure in which all data elements are stored in a sequence is known as linear data structure.	A data structure in which all data elements are not stored in a sequence is known as non-linear data structure.
2	All elements are stored in contiguous memory locations inside memory.	All elements may stored in non-contiguous memory locations inside memory.
3	Example:- stack, queue	Example:- tree, graph

7. Convert the following infix expression to its prefix form using stack $A + B - C * D/E + F$

Ans:

Infix Expression	Read Character	Stack contents	Prefix Expression
A+B-C*D/E+F	F		F
A+B-C*D/E+	+	+	F
A+B-C*D/E	E	+	EF
A+B-C*D/	/	/ +	EF
A+B-C*D	D	/ +	DEF
A+B-C*	*	* +	/DEF
A+B-C	C	* +	C/DEF
A+B-	-	- +	+*C/DEF
A+B	B	- +	B+*C/DEF
A+	+	+ +	-B+*C/DEF
A	A	+ +	A-B+*C/DEF
			+A-B+*C/DEF

(WINTER-19)

1. Write any four operations that can be performed on data structure.

Ans 1. Data structure operations (Non-Primitive)

2. Inserting: Adding a new data in the data structure is referred as insertion.

3. Deleting: Removing a data from the data structure is referred as deletion.

4. Sorting: Arranging the data in some logical order (ascending or descending, numerically or alphabetically).

5. Searching: Finding the location of data within the data structure which satisfy the searching condition.

6. Traversing: Accessing each data exactly once in the data structure so that each data item is traversed or visited.

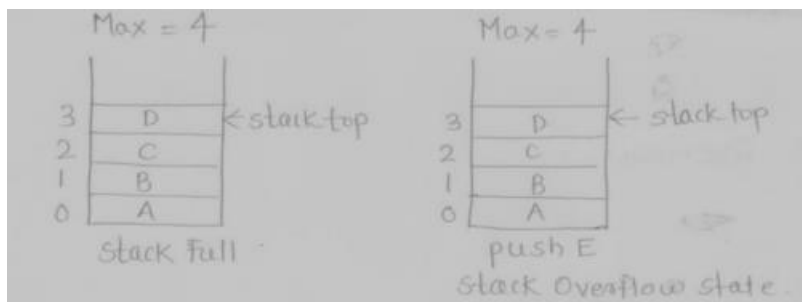
7. Merging: Combining the data of two different sorted files into a single sorted file.

8. Copying: Copying the contents of one data structure to another.

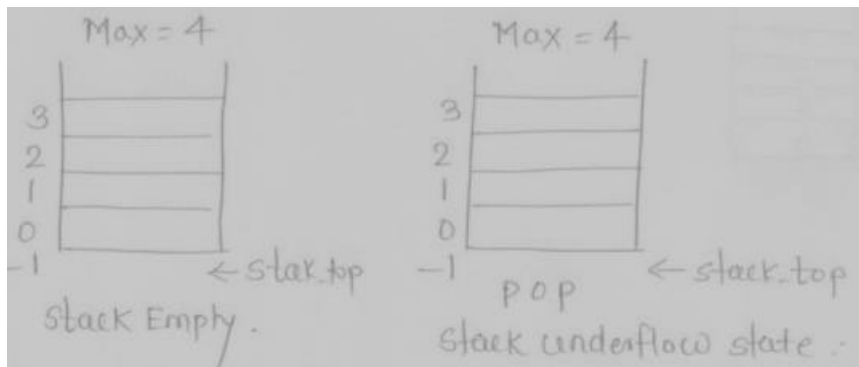
9. Concatenation: Combining the data from two or more data structure.

2. Define the term overflow and underflow with respect to stack.

Ans: Stack overflow: When a stack is full and push operation is performed to insert a new element, stack is said to be in overflow state



Stack underflow: When there is no element in a stack (stack empty) and pop operation is called then stack is said to underflow state.



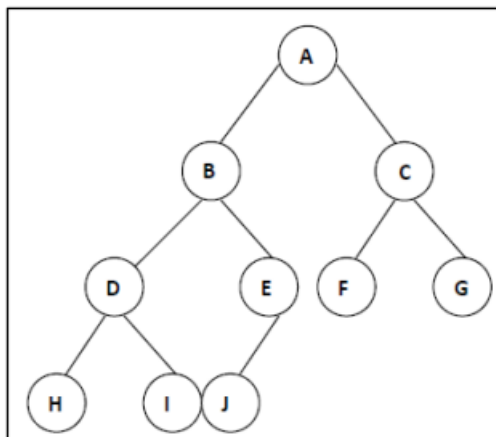
3. Define the following term w.r.t. tree: (i) In-degree (ii) out-degree.

Ans: In -degree: Number of edges coming towards node is in-degree of node.

For e.g. : In degree of node B is 1

Out -degree: Number of edges going out from node is out -degree of node.

For e.g. Out Degree of is node D is 2



4. Evaluate the following arithmetic expression P written in postfix notation:

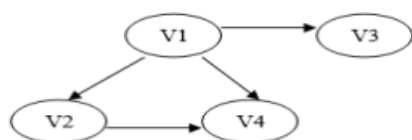
P : 4, 2, ^, 3, *, 3, -, 8, 4, /, +

Ans:

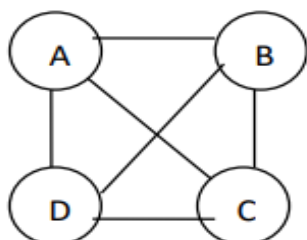
Sr. No.	Symbol Scanner	STACK
1	4	4
2	2	4, 2
3	^	16
4	3	16, 3
5	*	48
6	3	48,3
7	-	45
8	8	45,8
9	4	45,8,4
10	/	45,2
11	+	47

5. Describe directed and undirected graph.

Ans: Direct Graph: A directed graph is defined as the set of ordered pair of vertices and edges where each connected edge has assigned a direction.

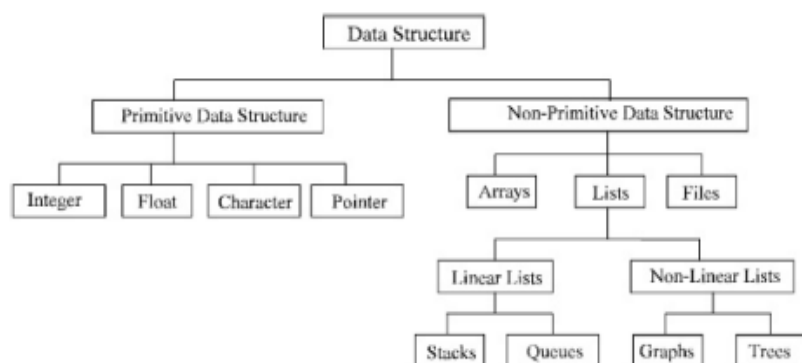


Undirected Graph : An undirected graph G is a graph in which each edge e is not assigned a direction



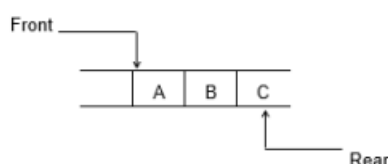
6. Give classification of data structure.

Ans:



7. Define queue. State any two applications where queue is used.

Ans: A Queue is an ordered collection of items. It has two ends, front and rear. Front end is used to delete element from queue. Rear end is used to insert an element in queue. Queue has two ends; the element entered first in the queue is removed first from the queue. So it is called as FIFO list



APPLICATIONS OF QUEUES:

1. Round Robin Technique for processor scheduling is implemented using queues.
2. All types of customer service (like railway ticket reservation) center software's are designed using queues to store customer's information.
3. Printer server routines are designed using queues. A number of users share a printer using printer server (a dedicated computer to which a printer is connected), the printer server then spools all the jobs from all the users, to the server's hard disk in a queue. From here jobs are printed one-by-one according to their number in the queue

(SUMMER-22)

1. Define linear data structure and non-linear data structure.

Ans: In a linear data structure, data elements are arranged in a linear order where each and every element is attached to its previous and next adjacent.

In a non-linear data structure, data elements are attached in hierarchically manner.

2. Enlist operations on stack.

Ans:

(c) Define : (i) General tree (ii) Binary tree

(d) Draw the diagram of circular queue with front and rear pointers.

(e) Describe given two types of graphs: Directed and Undirected graph.

(f) Define Abstract Data Type.

(g) State any four applications of queue.



4 MARKS QUESTION
(WINTER-18)

1. Describe working of linear search with example.

Ans: In linear search, search element is compared with each element from the list in a sequence. Comparison starts with first element from the list and continues till number is found or comparison reaches to the last element of the list. As each element is checked with search element, the process of searching requires more time. Time complexity of linear search is $O(n)$ where n indicates number of elements in list. Linear search on sorted array: - On sorted array search takes place till element is found or comparison reaches to an element greater than search element.

Example: - Using array representation Input list 10, 20, 30, 40, 50 and Search element 30, Index = 0

Iteration 1

10	20	30	40	50
----	----	----	----	----



$10 \neq 30$

Index = Index + 1

Iteration 2

10	20	30	40	50
----	----	----	----	----



$20 \neq 30$

Index = Index + 1

Iteration 3

10	20	30	40	50
----	----	----	----	----



$30 = 30$

Number found

2. Describe the concept of linked list with the terminologies: node, next Pointer, null pointer and empty list.

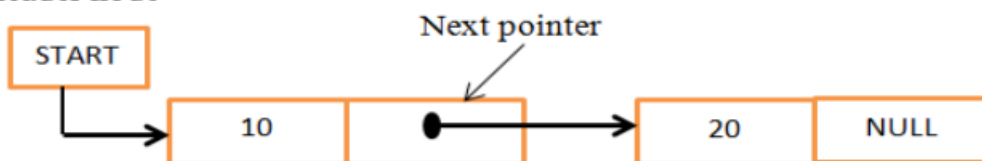
Ans Node: Each data element in a linked list is represented as a node. Node contains two parts one is info (data) and other is next pointer (address). Info part stores data and next pointer stores address of next node.

Node



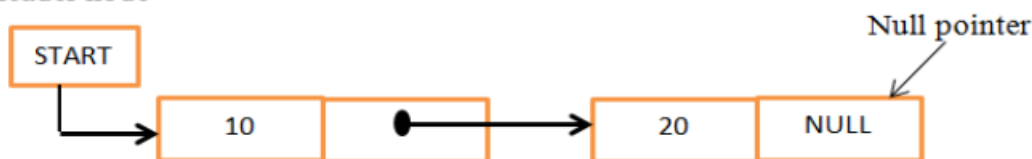
Next pointer: It is a pointer that holds address of next node in the list i.e. next pointer points to next node in the list

Header node



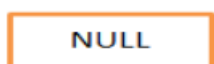
Null pointer: It is a pointer that does not hold any memory address i.e. it is pointing to nothing. It is used to specify end of the list. The last element of list contains NULL pointer to specify end of list.

Header node



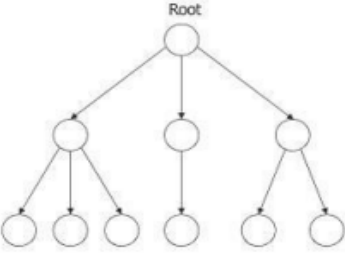
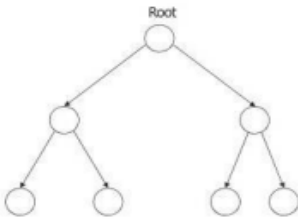
Empty list: Each linked list has a header node. When header node contains NULL value, then that list is said to be empty list.

Header node



3. Describe queue full and queue empty operation conditions on linear queue with suitable diagrams.

Ans: Queue full:-A queue is full when its rear pointer points to max -1 position. Max is maximum number of elements in a queue. If rear pointer is not equal to max-1 then a new element can be added to a queue. If queue is full then new element cannot be added to a queue.

Sr. no	General Tree	Binary Tree
1	A general tree is a data structure in which each node can have infinite number of children	A Binary tree is a data structure in which each node has at most two nodes i.e. left and right
2	In general tree, root has in-degree 0 and maximum out-degree n .	In binary tree, root has in-degree 0 and maximum out-degree 2 .
3	In general tree, each node have in-degree one and maximum out-degree n .	In binary tree, each node have in-degree one and maximum out-degree 2 .
4	Height of a general tree is the length of longest path from root to the leaf of tree. $\text{Height}(T) = \{ \max(\text{height}(\text{child1}), \text{height}(\text{child2}), \dots, \text{height}(\text{child-n}) + 1) \}$	Height of a binary tree is : $\text{Height}(T) = \{ \max(\text{Height}(\text{Left Child}), \text{Height}(\text{Right Child}) + 1) \}$
5	Subtree of general tree are not ordered	Subtree of binary tree is ordered .
6	<p style="text-align: center;">General tree</p> 	<p style="text-align: center;">Binary Tree</p> 

5. Write a C program for deletion of an element from an array.

Ans:

```
#include <stdio.h>
int main()
{
    int array[100], position, c, n;
    printf("Enter number of elements in array\n");
    scanf("%d", &n);
    printf("Enter %d elements\n", n);
    for (c = 0; c < n; c++)
        scanf("%d", &array[c]);
    printf("Enter the location where you wish to delete element\n");
    scanf("%d", &position);
    if (position >= n+1)
        printf("Deletion not possible.\n");
    else
    {
        for (c = position - 1; c < n - 1; c++)
            array[c] = array[c+1];

        printf("Resultant array:\n");

        for (c = 0; c < n - 1; c++)
            printf("%d\n", array[c]);
    }
    return 0;
}
```

6. Convert following expression into postfix form. Give stepwise procedure.

$A+B \uparrow C*(D/E)-F/G$.

Ans:

Scanned Symbol	Operation stack	Postfix Expression
((
A	(A
+	(+	A
B	(+	AB
↑	(+↑	AB
C	(+↑	ABC
*	(+*	ABC↑
((+*(ABC↑
D	(+*(ABC↑D
/	(+*(/	ABC↑D
E	(+*(/	ABC↑DE
)	(+*	ABC↑DE/
-	(-	ABC↑DE/*+
F	(-	ABC↑DE/*+F
/	(-/	ABC↑DE/*+F
G	(-/	ABC↑DE/*+FG
)	EMPTY	ABC↑DE/*+FG/-

POSTFIX EXPRESSION: ABC↑DE/*+FG/-

7. Find the position of element 29 using binary search method in an array 'A' given below. Show each step. A={11,5,21,3,29,17,2,43}

Ans:

An array which is given $A[] = \{11,5,21,3,29,17,2,43\}$ is not in sorted manner, first we need to sort them in order; So an array will be $A[] = \{2,3,5,11,17,21,29,43\}$ and the value to be searched is $VAL = 29$. The binary search algorithm will proceed in the following manner.

A[0]	A[1]	A[2]	A[3]	A[4]	A[5]	A[6]	A[7]
2	3	5	11	17	21	29	43

Iteration 1:

BEG = 0, END = 7, MID = $(0 + 7)/2 = 3$

Now, VAL = 29 and A[MID] = A[3] = 11

A[3] is less than VAL, therefore, we now search for the value in the second half of the array.

So, we change the values of BEG and MID.

Iteration 2:

Now, BEG = MID + 1 = 4, END = 7, MID = $(4 + 7)/2 = 11/2 = 5$; VAL = 29 and A [MID] = A [5] = 21

A[5] is less than VAL, therefore, we now search for the value in the second half of the segment.

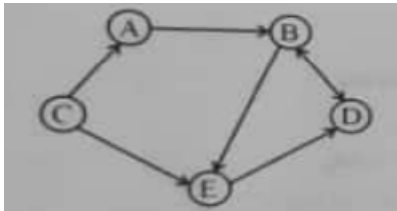
So, again we change the values of BEG and MID.

Iteration 3:

Now, BEG = MID + 1 = 6, END = 7, MID = $(6 + 7)/2 = 6$ Now, VAL = 29 and A [MID] = A [6]=29

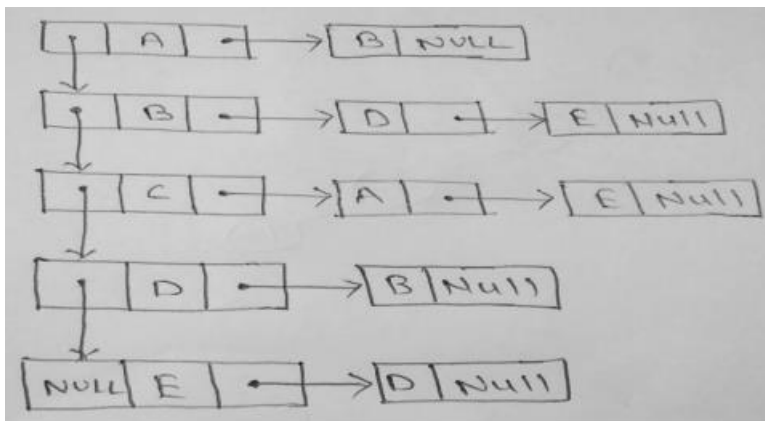
So, Element 29 is found at 6th location in given array A[]={2,3,5,11,17,21,29,43}

8. give adjacency list and adjacency matrix for given graph.



Ans:

Adjacency List: (Using Linked List) Here, we use doubly linked list for storing header node list and singly linked list for storing respective adjacent node to it.



9. Describe working of bubble sort with example.

Ans:

Bubble sort is a simple sorting algorithm. This sorting algorithm is comparison-based algorithm in which each pair of adjacent elements is compared and the elements are swapped if they are not in order. This algorithm is not suitable for large data sets as its average and worst case complexity is of $O(n^2)$ where n is the number of items.

Bubble Sort Working:

We take an unsorted array for our example as $A[] = \{19, 2, 27, 3, 7, 5, 31\}$.

Bubble sort takes $O(n^2)$ time so we're keeping it short and precise.

{{**Note: Pass 4 onwards optional**}}

Pass 1: 2,19,27,3,7,5,31

2,19,27,3,7,5,31

2,19,3,27,7,5,31

2,19,3,7,27,5,31

2,19,3,7,5,27,31

Pass 1 Completed

Pass 2: 2,19,3,7,5,27,31

2,3,19,7,5,27,31

2,3,7,19,5,27,31

2,3,7,5,19,27,31

2,3,7,5,19,27,31

Pass 2 Completed

Pass 3: 2,3,7,5,19,27,31

2,3,7,5,19,27,31

2,3,5,7,19,27,31

Pass 3 Completed

Pass 4: 2,3,5,7,19,27,31

Pass 4 Completed

Pass 5: 2,3,5,7,19,27,31

Pass 5 Completed

Pass 6: 2,3,5,7,19,27,31

Pass 6 Completed

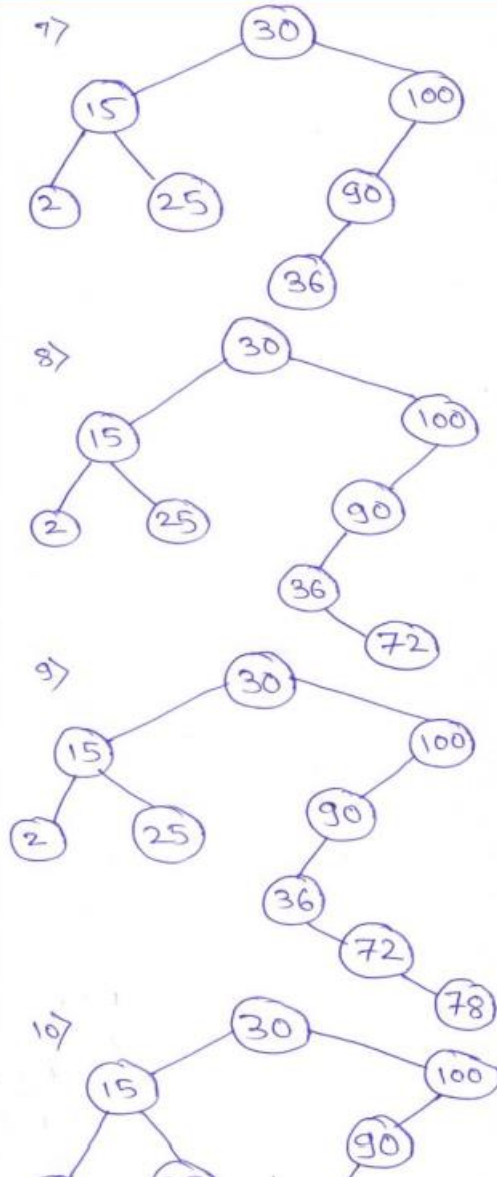
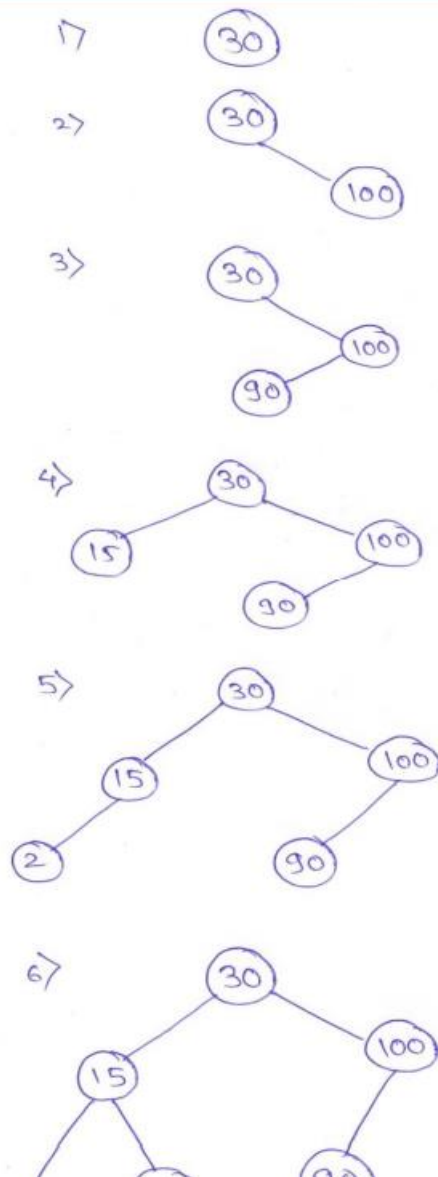
10. Construct a binary search tree for following elements:

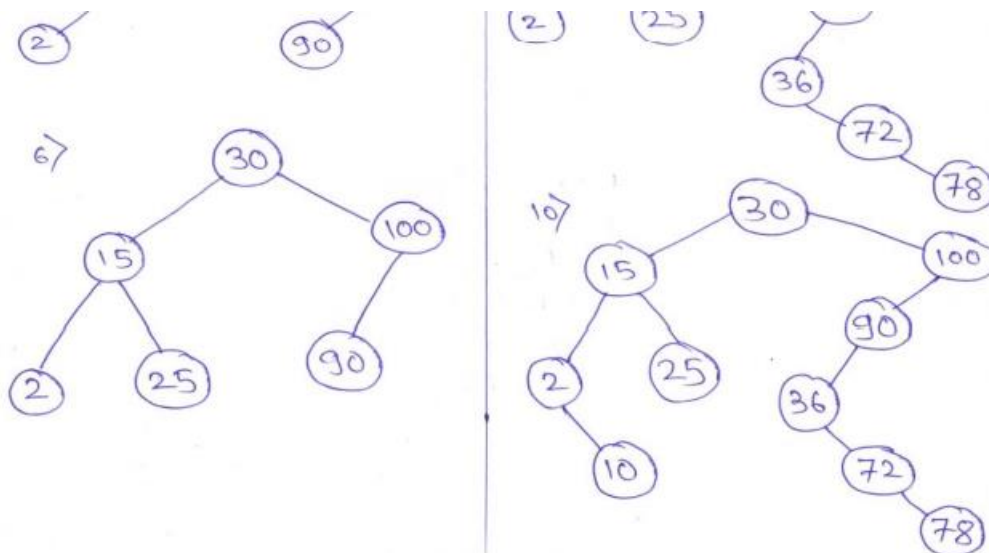
30,100,90,15,2,25,36,72,78,10 show each step of construction of BST.

Ans: Stepwise construction of Binary search tree for following elements:

30,100,90,15,2,25,36,72,78,10 is as follows:

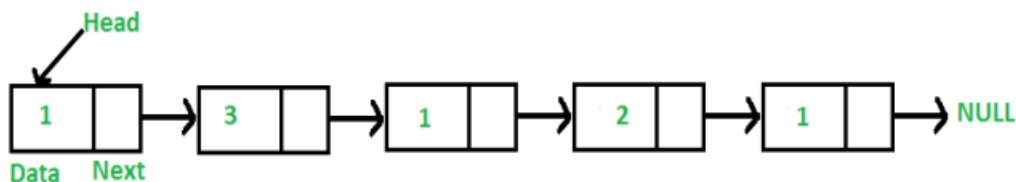






11. Write an algorithm to count number of nodes in singly linked list.

Ans Function to count number of nodes in a given singly linked list



Output: 5

For example, the function should return 5 for linked list 1->3->1->2->1.

Algorithm: Using Iterative Solution

- 1) Initialize count as 0
- 2) Initialize a node pointer, current = head.
- 3) Do following while current is not NULL
 - a) current = current -> next
 - b) count++;
- 4) Return count

12. Write a program in 'C' to insert an element in a linear queue.

Ans // C program to insert an element in a linear queue using array

```
#include
#include
#define n 5
void main()
{
int queue[n],ch=1,front=0,rear=0,i,j=1,x=n;
```



```
//clrscr();
printf("Queue using Array");
printf("\n1.Insertion \n2.Display \n3.Exit");
while(ch)
{
printf("\nEnter the Choice:");
scanf("%d",&ch); switch(ch)
{
case 1: if(rear==x)
printf("\n Queue is Full");
else
{
printf("\n Enter no %d:",j++);
scanf("%d",&queue[rear++]);
}
break;
case 2: printf("\n Queue Elements are:\n ");
if(front==rear)
printf("\n Queue is Empty");
else
{
for(i=front; i<rear; i++)
{
printf("%d",queue[i]);
printf("\n");
}
break;
case 3:
exit(0);
default:
printf("Wrong Choice: please see the options");
}
}
}
getch();
}
```

13. Describe circular linked list with suitable diagram. Also state advantage of circular linked list over linear linked list.

Ans Circular Linked List A circular linked list is a variation of linked list in which the last element is linked to the first element. This forms a circular loop.



A circular linked list can be either singly linked or doubly linked.

- for singly linked list, next pointer of last item points to the first item
- In doubly linked list, prev pointer of first item points to last item as well.

We declare the structure for the circular linked list in the same way as follows:

Struct node {

Int data;

Struct node *next;

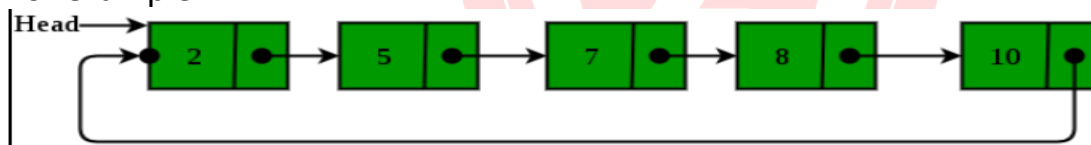
};

Typedef struct node *Node;

Node *start = null;

Node *last = null;

For example:



Advantages of Circular Linked Lists:

- 1) Any node can be a starting point. We can traverse the whole list by starting from any point. We just need to stop when the first visited node is visited again.
- 2) Useful for implementation of queue. Unlike this implementation, we don't need to maintain two pointers for front and rear if we use circular linked list. We can maintain a pointer to the last inserted node and front can always be obtained as next of last.
- 3) Circular lists are useful in applications to repeatedly go around the list. For example, when multiple applications are running on a PC, it is common for the operating system to put the running applications on a list and then to cycle through them, giving each of them a slice of time to execute, and then making them wait while the CPU is given to another application. It is convenient for the operating system to use a circular list so that when it reaches the end of the list it can cycle around to the front of the list.

4) Circular Doubly Linked Lists are used for implementation of advanced data structures like Fibonacci Heap.

(SUMMER-19)

1. Explain the working of Binary search with an example.

Ans: Binary search is performed only on sorted array. Search method starts with calculating mid position from an array and compare the mid position element with the search element. If a match is found then the search process ends otherwise divide the i/p list into 2 parts. First part contains all the numbers less than mid position element and second part contains all the numbers greater than mid position element. Then select one of the part depending on search element is less or greater than mid position element and calculate mid position for selected part. Again compare mid position element with search element. The binary search performs comparison and division task the element is found or division of list gives one element for comparison. To calculate mid element perform $(\text{lower} + \text{upper}) / 2$. lower-lower index position of an array (initially 0) upper-upper index position of an array (initially size-1)

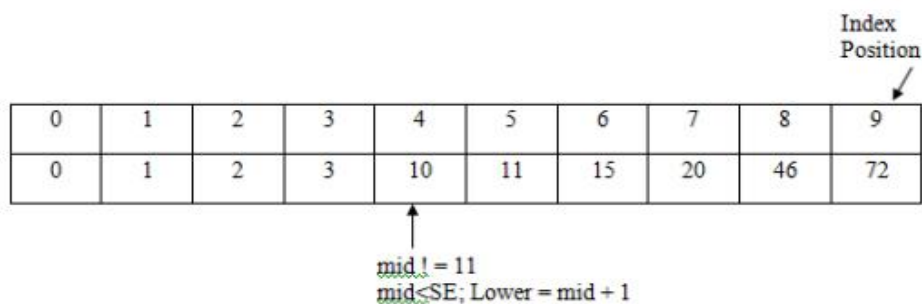
Example:

Consider Input list 0, 1, 2, 9, 10, 11, 15, 20, 46, 72

Search element: 11

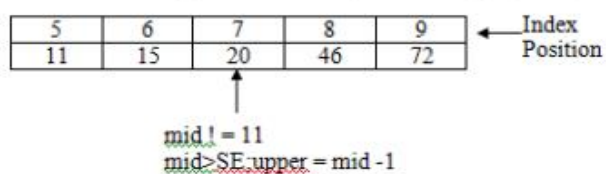
→ Iteration 1

Lower = 0 Upper = 9 $mid = (lower + upper) / 2 = (0 + 9/2) = 4.5$



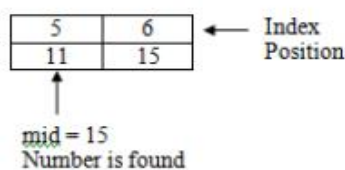
→ Iteration 2

Lower = 5 Upper = 9 $mid = (Lower + Upper) / 2 = (5 + 9) / 2 = 7$



→ Iteration 3

Lower = 5 upper = 6 $mid = (Lower + Upper) / 2 = (5 + 6) / 2 = 5.5$



2. Write a program to traverse a linked list.

Ans:

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

```
void create_list(int);
void addatbeg(int);
void display();
struct node
```

```
{
int info;
struct node *next;
}*start=NULL;

void main()
{
int m;
clrscr();
printf("enter data value\n");
scanf("%d",&m);
create_list(m);
printf("enter data value\n");
scanf("%d",&m);
addatbeg(m);
display();
getch();
}

void create_list(int data)
{
struct node *tmp,*q;
tmp=malloc(sizeof(struct node));
tmp->info=data;
tmp->next=NULL;
start=tmp;
}
void addatbeg(int data)
{
struct node *tmp;
tmp=malloc(sizeof(struct node));
tmp->info=data;
tmp->next=start;
start=tmp;
}

void display()
{
```

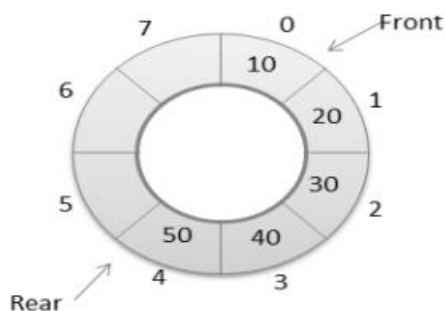


```

struct node *q;
if(start==NULL)
{
printf("list is empty\n");
}
q=start;
printf("list is:\n");
while(q!=NULL)
{
printf("%d\t",q->info);
q=q->next;
}
}
    
```

3. Draw and explain construction of circular queue.

Ans: A queue, in which the last node is connected back to the first node to form a cycle, is called as circular queue.



The above diagram represents a circular queue using array. It has rear pointer to insert an element and front pointer to delete an element. It works in FIFO manner where first inserted element is deleted first. Initially front and rear both are initialized to -1 to represent queue empty. First element inserted in circular queue is stored at 0th index position pointed by rear pointer. For the very first element, front pointer is also set to 0th position. Whenever a new element is inserted in a queue rear pointer is incremented by one. If rear is pointing to max-1 and no element is present at 0th position then rear is set to 0th position to continue cycle. Before inserting an element, queue full condition is checked. If rear is set to max-1 position and front is set to 0 then queue is full. Otherwise if rear = front+1 then also queue is full. If queue is full then new element cannot be added in a queue. For deletion, front pointer position is checked and queue empty condition is checked. If front pointer is pointing to -1 then queue is empty and deletion operation cannot be performed. If queue contains any element then front pointer is incremented by one to remove an element. If front pointer is pointing to max-1 and element is present at 0th position then front pointer is initialize to 0th position to continue cycle. Circular queue has advantage of utilization of space. Circular

queue is full only when there is no empty position in a queue. Before inserting an element in circular queue front and rear both the pointers are checked. So if it indicates any empty space anywhere in a queue then insertion takes place.

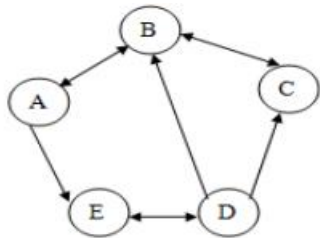
4. Explain indegree and outdegree of a graph with example.

Ans: Indegree of node: It is number of edges coming towards a specified node i.e. number of edges that have that specified node as the head is known as indegree of a node.

Outdegree of node: It is number of edges going out from a specified node i.e. number of edges that have that specified node as the tail is known as outdegree of a node

In undirected graph each edge is bidirectional so each edge coming towards node is also going out of that node. Due to this indegree and outdegree of a node is same number. In directed graph, each edge is having direction associated with it, so indegree and outdegree depends on the direction.

Example:-



Indegree of node A= 1 Outdegree of node A=2

Indegree of node B= 3 Outdegree of node B=2

Indegree of node C= 2 Outdegree of node C=1

Indegree of node D= 1 Outdegree of node D=3

Indegree of node E= 2 Outdegree of node E=1

5. Write C program for performing following operations on array: insertion, display.

Ans:

```
#include<stdio.h>
#include<conio.h>
void main()
{
inta[10],x,i,n,pos;
clrscr();
printf("Enter the number of array element\n");
scanf("%d",&n);
printf("Enter the array with %d element\n", n);
for(i=0;i<n;i++)
scanf("%d",&a[i]);
printf("Enter the key value and its position\n");
scanf("%d%d" ,&x,&pos);
for(i=n; i >= pos; i--)
{
a[i]=a[i-1];
}
a[pos-1]=x;
printf("Array element\n ");
for(i=0;i<n+1;i++)
printf("%d\t",a[i]);
getch();
}
```

6. Evaluate the following postfix expression: 5, 6, 2, +, *, 12, 4, /, - Show diagrammatically each step of evolution using stack.

Ans:

Scanned Symbol	Operand 1	Operand 2	Value	Stack Content
5				5
6				5,6
2				5,6,2
+	6	2	8	5,8
*	5	8	40	40
12				40,12
4				40,12,4
/	12	4	3	40,3
-	40	3	37	37

Result of above postfix expression evaluation- 37

7. Sort the following numbers in ascending order using quick sort.

Given numbers 50, 2, 6, 22, 3, 39, 49, 25, 18, 5.

Ans:

Given array

Array elements	50	2	6	22	3	39	49	25	18	5
indexes	0	1	2	3	4	5	6	7	8	9

Set $l=0$, $h=9$, $\text{pivot} = a[h]=5$

Initialize index of smaller element, $i = l-1 = -1$

Traverse elements from $j=l$ to $j=h-1$

1. $j=0$ $i=-1$ since $a[j] > \text{pivot}$ do nothing array will remain same

Array elements	50	2	6	22	3	39	49	25	18	5
indexes	0	1	2	3	4	5	6	7	8	9

2. $j=1$ since $a[j] \leq \text{pivot}$, do $i++$ and $\text{swap}(a[i], a[j])$
 $i=0$

Array elements	2	50	6	22	3	39	49	25	18	5
indexes	0	1	2	3	4	5	6	7	8	9

3. $j=2, i=0$ since $a[j] > \text{pivot}$ do nothing array will remain same

Array elements	2	50	6	22	3	39	49	25	18	5
indexes	0	1	2	3	4	5	6	7	8	9

4. $j=3, i=0$ since $a[j] > \text{pivot}$ do nothing array will remain same

Array elements	2	50	6	22	3	39	49	25	18	5
indexes	0	1	2	3	4	5	6	7	8	9

5. $j=4$, since $a[j] \leq \text{pivot}$ do, $i++$ and $\text{swap}(a[i], a[j])$
 $i=1$

Array elements	2	3	6	22	50	39	49	25	18	5
indexes	0	1	2	3	4	5	6	7	8	9

6. $j=5, i=1$ since $a[j] > \text{pivot}$ do nothing array will remain same

Array elements	2	3	6	22	50	39	49	25	18	5
indexes	0	1	2	3	4	5	6	7	8	9

7. $j=6, i=1$ since $a[j] > \text{pivot}$ do nothing array will remain same

Array elements	2	3	6	22	50	39	49	25	18	5
indexes	0	1	2	3	4	5	6	7	8	9

8. $j=7, i=1$ since $a[j] > \text{pivot}$ do nothing array will remain same

Array elements	2	3	6	22	50	39	49	25	18	5
indexes	0	1	2	3	4	5	6	7	8	9

9. $j=8, i-1$ since $a[j] > \text{pivot}$ do nothing array will remain same

Array elements	2	3	6	22	50	39	49	25	18	5
indexes	0	1	2	3	4	5	6	7	8	9

We come out of loop because j is now equal to $\text{high}-1$.

Finally we place pivot at correct position by swapping $a[i+1]$ and $a[h]$ (or pivot)

$a[] = \{2,3,5,22,50,39,49,25,18,6\}$ // 6 and 5 Swapped

Now, 5 is at its correct place. All elements smaller than 5 are before it and all elements greater than 5 are after it.

Similarly rest of the passes will be executed and will provide the following output

Output of pass 1

Array elements	2	3	5	22	50	39	49	25	18	6
indexes	0	1	2	3	4	5	6	7	8	9

Pass 2

$A[] = \{2,3\}$ pivot=3

Array elements	2	3	5
indexes	0	1	2

$a[] = \{22,50,39,49,25,18,6\}$ pivot=6

Array elements	6	50	39	49	25	18	22
indexes	3	4	5	6	7	8	9

$a[] = \{50,39,49,25,18,22\}$ pivot=22

Array elements	18	22	49	25	50	39
indexes	4	5	6	7	8	9

$a[] = \{18\}$, pivot=18

Array elements	18	22
indexes	4	5

$a[] = \{49, 25, 50, 39\}$, pivot=39

Array elements	25	39	50	49
indexes	6	7	8	9

$a[] = \{25\}$, pivot=25

Array elements	25	39
indexes	6	7

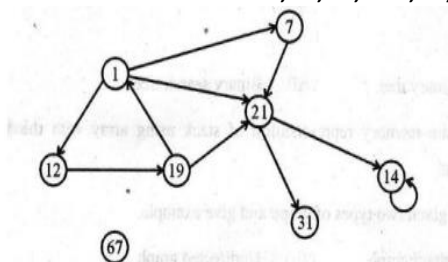
$a[] = \{50, 49\}$, pivot=49

Array elements	49	50
indexes	8	9

Final sorted array using quick sort will be

Array elements	2	3	5	6	18	22	25	39	49	50
indexes	0	1	2	3	4	5	6	7	8	9

8. Sort the following numbers in ascending order using quick sort.
Given numbers 50, 2, 6, 22, 3, 39, 49, 25, 18, 5.



- (i) Indegree of node 21
- (ii) Adjacent node of 19
- (iii) Path of 31
- (iv) Successor of node 67

Ans: (i) Indegree of node 21: node 1, 7, 19

(i1) Adjacent node of 19: node 1, 21

(iii) Path of 3l: Path1: 1-21-31 Path2: 1-7-21-31 Path3: 1-7-21-31

(iv) Successor of node 67: No Successor of node 67 since it is isolated node or not connected node in node.

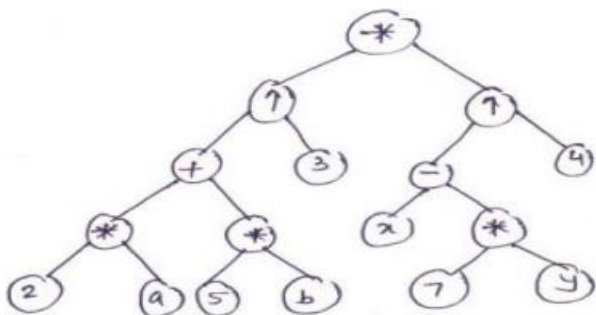
9. Differentiate between binary search and sequential search (linear search).

Ans:

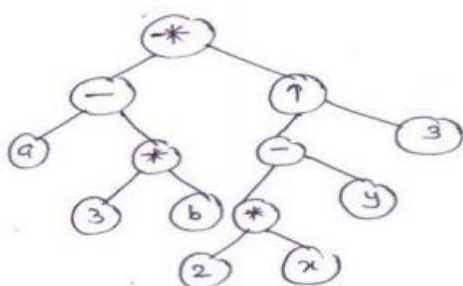
Sr. No.	Binary Search	Sequential search (linear search)
1	Input data needs to be sorted in Binary Search	Input data need not to be sorted in Linear Search.
2	In contrast, binary search compares key value with the middle element of an array and if comparison is unsuccessful then cuts down search to half.	A linear search scans one item at a time, without jumping to any item.
3	Binary search implements divide and conquer approach.	Linear search uses sequential approach.
4	In binary search the worst case complexity is $O(\log n)$ comparisons.	In linear search, the worst case complexity is $O(n)$, comparisons.
5	Binary search is efficient for the larger array.	Linear search is efficient for the smaller array.

10. Draw the tree structure of the following expressions:

(i) $(2a+5b)^3 * (x-7y)^4$ (ii) $(a-3b) * (2x-y)^3$
 (i) $(2a+5b)^3 * (x-7y)^4$



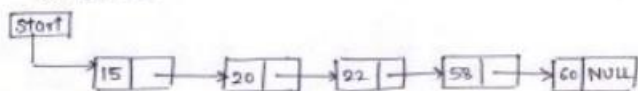
(ii) $(a-3b) * (2x-y)^3$



11. Create a singly linked list using data fields 15, 20, 22, 58, 60. Search a node 22 from the SLL and show procedure step-by-step with the help of diagram from start to end.

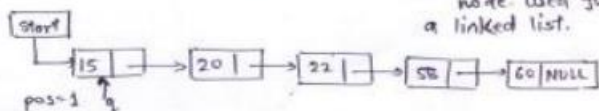
Ans:

① With given data fields, singly linked list is created as follows



② Operation - Search a node 22 from the above SLL

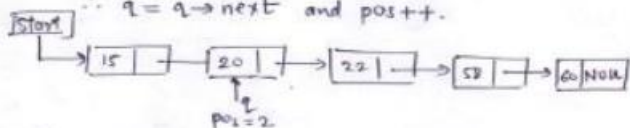
a) Initially $q = \text{start}$ where q is a pointer of type struct node used for traversing a linked list.



b) $q \neq \text{NULL}$ and $\text{pos} = 1$

$q \rightarrow \text{data} \neq \text{key value}$
i.e. $15 \neq 22$

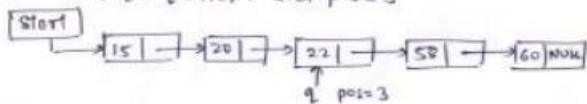
$\therefore q = q \rightarrow \text{next}$ and $\text{pos}++$.



c) $q \neq \text{NULL}$ and $\text{pos} = 2$

$q \rightarrow \text{data} \neq \text{key value}$
i.e. $20 \neq 22$

$\therefore q = q \rightarrow \text{next}$ and $\text{pos} = 3$



$q \neq \text{NULL}$ and $\text{pos} = 3$

$q \rightarrow \text{data} == \text{key value}$

i.e. $22 == 22$

\therefore node 22 is located at position 3
search is successful.

12. Evaluate the following prefix expression: $- * + 4 3 2 5$ show diagrammatically each step of evaluation using stack.

Ans:

Scanned Symbol	Operand 1	Operand 2	Value	Stack Content
5				5
2				5,2
3				5,2,3
4				5,2,3,4
+	4	3	12	5,2,12
*	12	2	24	5,24
-	24	5	19	19

Result of above prefix expression evaluation - 19

13. Write an algorithm to delete a node from the beginning of a circular linked list.

Ans: Algorithm to delete a node from the beginning of a circular linked list

Consider the function delatbeg()

1. Start

2. Declare struct node *tmp,*q;

3. Set q=last->link;

4. While (q!= last)

Do

tmp = q; // Identifies beginning node of Circular Linked List

last->link=q->link; // Set the address field before deleting identified node free(tmp);

// Delete the beginning node End of While

5. last=NULL;

// Set last= NULL if only one node is present in the Circular Linked List

6. End of function

(WINTER-19)

1. Sort the given number in ascending order using Radix sort: 348, 14, 641, 3851, 74.

Ans:

Pass 1:

	0	1	2	3	4	5	6	7	8	9
0348									0348	
0014					0014					
0641		0641								
3851		3851								
0074					0074					

0641,3851,0014,0074,0348

Pass 2:

	0	1	2	3	4	5	6	7	8	9
0641					0641					
3851						3851				
0014		0014								
0074								0074		
0348					0348					

0014,0641,0348,3851,0074

Pass 3:

	0	1	2	3	4	5	6	7	8	9
0014	0014									
0641							0641			
0348				0348						
3851									3851	
0074	0074									

0014,0074,0348,0641,3851

Pass 4:

	0	1	2	3	4	5	6	7	8	9
0014	0014									
0074	0074									
0348	0348									
0641	0641									
3851					3851					

Sorted Elements are: 14, 74, 348, 641, 3851

2. Write an algorithm to insert a new node at the beginning and end of the singly linked list.
 Ans 1. Algorithm for inserting a node at the beginning

Insert first(start, item)

1. [check the overflow]

if Ptr=NULL then print 'Overflow'

exit

else

Ptr=(node *) malloc (size of (node))

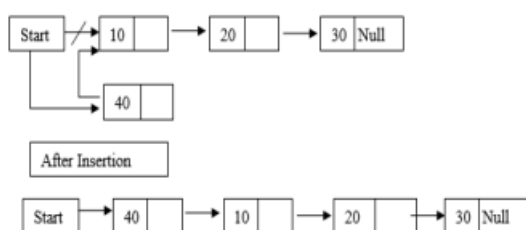
//create new node from memory and assign its address to ptr

End if

2. set Ptr->num = item

3. set Ptr->next=start

4. set start=Ptr



2. Algorithm for Inserting A Node at the End

insert last (start, item)

1. [check for overflow]

If Ptr=NULL, then print 'Overflow'

exit

else

Ptr=(node *) malloc (sizeof (node));

end if

2. set Ptr->info=item

3. set Ptr->next=NULL

4. if start=NULL and

if then set start=P

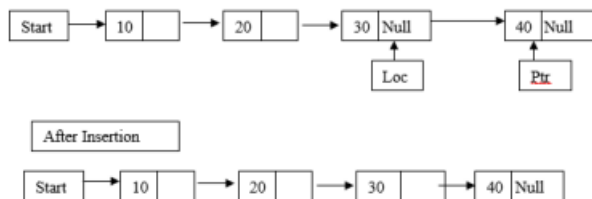
5. set loc=start

6. repeat step 7

until loc->next != NULL

7. set loc=loc->next

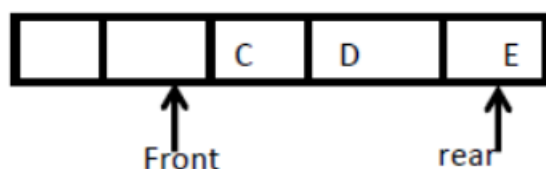
8. set loc->next=P



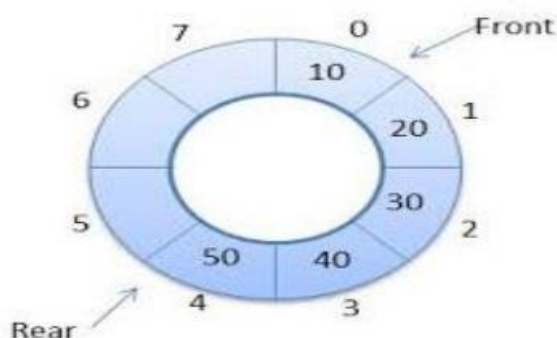
3. Explain the concept of circular Queue along with its need.

Ans: • Circular queue are the queues implemented in circular form rather than in a straight line.

- Circular queues overcome the problem of unutilized space in linear queue implemented as an array.
- The main disadvantage of linear queue using array is that when elements are deleted from the queue, new elements cannot be added in their place in the queue, i.e. the position cannot be reused. After rear reaches the last position, i.e. MAX-1 in order to reuse the vacant positions, we can bring rear back to the 0th position, if it is empty, and continue incrementing rear in same manner as earlier.
- Thus rear will have to be incremented circularly. For deletion, front will also have to be incremented circularly.
- Rear can be incremented circularly by the following code. If $((\text{rear} == \text{MAX}-1) \text{ and } (\text{front} \neq 0))$ $\text{Rear} = 0$; Else $\text{Rear} = \text{rear} + 1$; Example: Assuming that the queue contains three elements.



Now we insert an element F at the beginning by bringing rear to the first position in the queue. this can be represented circularly as shown.



Need of Circular Queue:

- Circular queues overcome the problem of unutilized space in linear queue implemented as an array.
- The element can be stored efficiently in an array so as to wrap around so that the end of queue is followed by front of the queue.

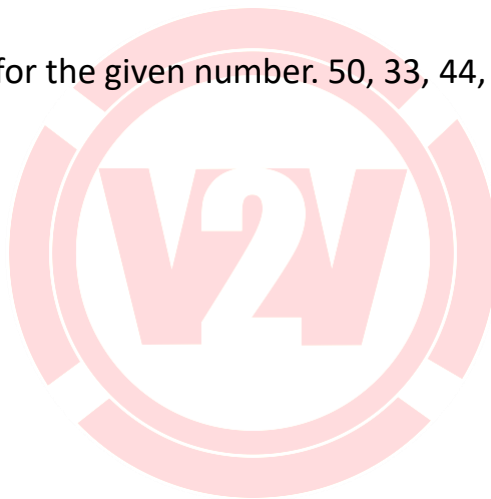
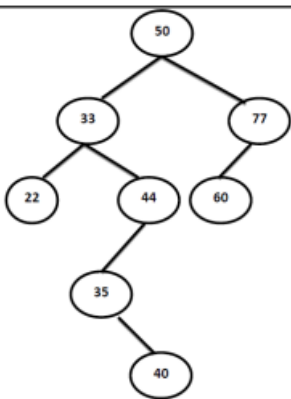
Now we insert an element F at the beginning by bringing rear to the first position in the queue. this can be represented circularly as shown.

Need of Circular Queue:

- Circular queues overcome the problem of unutilized space in linear queue implemented as an array.
- The element can be stored efficiently in an array so as to wrap around so that the end of queue is followed by front of the queue.

4. Draw a binary search tree for the given number. 50, 33, 44, 22, 77, 35, 60, 40.

Ans:



5. Explain time and space complexity with an example.

Ans Time Complexity: Time complexity of program or algorithm is amount of computer time that it needs to run to completion. To measure time complexity of an algorithm we concentrate on developing only frequency count for key statements.

Example: #include

void main ()

{

int i, n, sum, x;

sum=0;

printf("\n Enter no of data to be added");

scanf("% d", &n);

for(i=0 ; i<n; i++)

Statement	Frequenc y	Computational Time
sum=0	1	t ₁
printf("\n Enter no of data to be added")	1	t ₂
scanf("%d", &n)	1	t ₃
for(i=0 ; i<n; i++)	n+1	(n+1)t ₄
scanf("%d", &x)	n	nt ₅
sum=sum+x	n	nt ₆
printf("\n Sum = %d", sum)	1	t ₇

Total computational time= t₁+t₂+t₃+(n+1)t₄ +nt₆+nt₅+t₇

T= n(t₄+t₅+t₆) + (t₁+t₂+t₃+t₄+t₇)

For large n , T can be approximated to

T= n(t₄+t₅+t₆)= kn where k= t₄+t₅+t₆

Thus T = kn or

Space Complexity: Total amount of computer memory required by an algorithm to complete its execution is called as space complexity of that algorithm. When a program is under execution it uses the computer memory for THREE reasons.

They are as follows...

- Instruction Space: It is the amount of memory used to store compiled version of instructions.
- Environmental Stack: It is the amount of memory used to store information of partially executed functions at the time of function call.
- Data Space: It is the amount of memory used to store all the variables and constants. If the amount of space required by an algorithm is increased with the increase of input value, then that space complexity is said to be Linear Space Complexity.

Example:

```
int sum(int A[ ], int n)
{
    int sum = 0, i;
    for(i = 0; i < n; i++)
        sum = sum + A[i];
    return sum;
}
```

In the above piece of code it requires

'n*2' bytes of memory to store array variable

'a[]' 2 bytes of memory for integer parameter

'n' 4 bytes of memory for local integer variables 'sum' and 'i' (2 bytes each)

2 bytes of memory for return value.

That means, totally it requires ' $2n+8$ ' bytes of memory to complete its execution. Here, the total amount of memory required depends on the value of ' n '. As ' n ' value increases the space required also increases proportionately. This type of space complexity is said to be Linear Space Complexity.

6. Convert the following infix expression to postfix expression using stack and show the details of stack in each step.

$((A+B)*D)^{(E-F)}$

Ans: infix expression: $((A+B)*D)^{(E-F)}$



Current Symbol	Operator Stack	Postfix array
((Empty
(((Empty
(((Empty
A	((A
+	(((+	A
B	(((+	AB
)	((AB+
*	((*	AB+
D	((*	AB+D
)	(AB+D*
^	(^	AB+D*
((^(AB+D*
E	(^(AB+D*E
-	(^(-	AB+D*E
F	(^(-	AB+D*EF
)	(^	AB+D*EF-
)	EMPTY STACK	AB+D*EF-^

Postfix expression: AB+D*EF-^

7. Implement a 'C' program to search a particular data from the given array using Linear Search.

Ans: Program:

```
# include<stdio.h>
#include <conio.h>
void main ()
{
int a[10], n, key,i,c=0;
clrscr( );
printf (“Enter number of array elements\n”);
scanf (“%d”, &n);
printf (“Enter array elements\n”);
for (i=0; i< n; i++)
scanf (“%d”, &a[i]);
printf (“Enter key value\n”);
scanf (“%d”, &key);

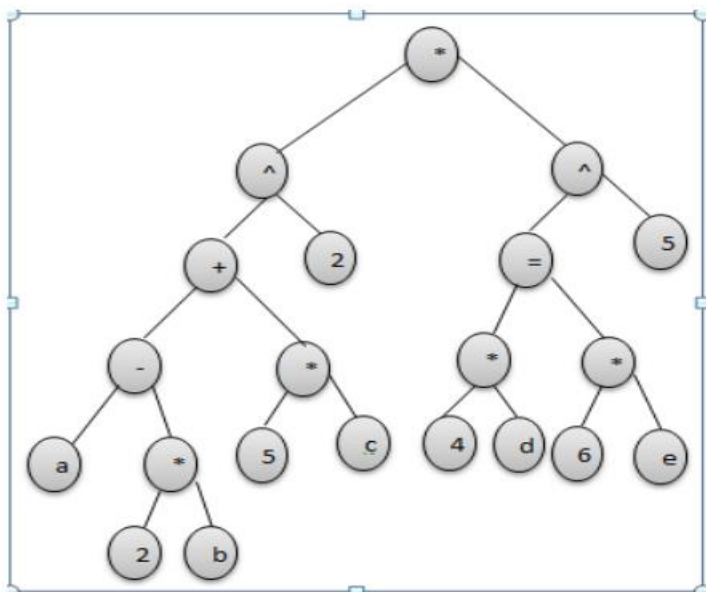
for(i=0;i<n-1;i++)
{

if (key == a[i])
{
c=1;
printf (“%d is found at location %d\n”, key, i+1);
break;
}

}
if (c==0)
printf (“%d not present in the list\n”,key);
getch();
}
```



8. Draw an expression tree for the following expression: $(a-2b+5e)^2 * (4d=6e)^5$.
Ans:



9. Find the position of element 21 using binary search method in array 'A' given below:

A={11,5,21,3,29,17,2,45}

Ans:

Given Array

11	5	21	3	29	17	2	45
----	---	----	---	----	----	---	----

Sorted Array for input:

2	3	5	11	17	21	29	45
---	---	---	----	----	----	----	----

Key element to be searched=21

Step1

0	1	2	3	4	5	6	7
2	3	5	11	17	21	29	45

$$l=0 \text{ and } u=n-1 =7$$

$$\text{mid}=(l+u)/2 = 7/2 = 3$$

a[mid]=11 not equal to 21

and

$21 > 11$ $l = \text{mid} + 1 = 4$ and $u = 7$

Step 2:

4	5	6	7
17	21	29	45

$l = 4$ and $u = 7$

$\text{mid} = 11/2 = 5$

$a[\text{mid}] = 21$ equal to key element 21

therefore key element 21 is found un array at position 6

10. Difference between tree and graph(Any 4 points)

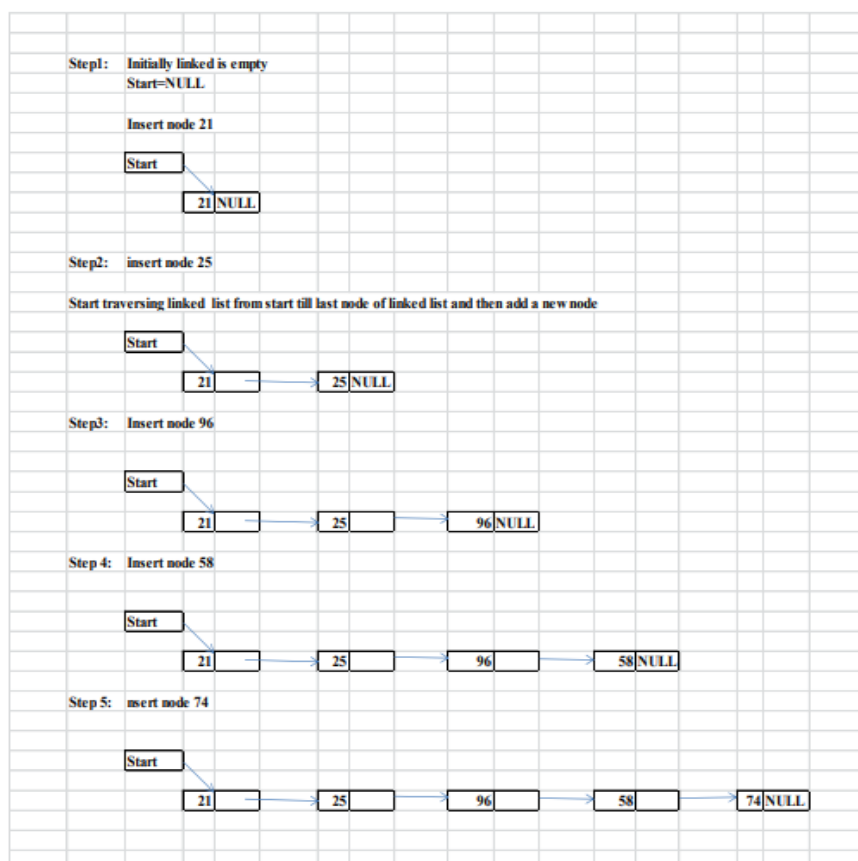
Ans:

Tree	Graph
Tree is special form of graph i.e. minimally connected graph and having only one path between any two vertices.	In graph there can be more than one path i.e. graph can have uni-directional or bi-directional paths (edges) between nodes
Tree is a special case of graph having no loops, no circuits and no self-loops.	Graph can have loops, circuits as well as can have self-loops.
Tree traversal is a kind of special case of traversal of graph. Tree is traversed in Pre-Order, In-Order and Post-Order	Graph is traversed by DFS: Depth First Search and in BFS : Breadth First Search algorithm
Different types of trees are: Binary Tree, Binary Search Tree, AVL tree, Heaps.	There are mainly two types of Graphs: Directed and Undirected graphs.

Tree applications: sorting and searching like Tree Traversal & Binary Search.	Graph applications : Coloring of maps, in OR (PERT & CPM), algorithms, Graph coloring, job scheduling, etc.
Tree always has n-1 edges.	In Graph, no. of edges depends on the graph.
Tree is a hierarchical model.	Graph is a network model.

11. Construct a singly linked list using data fields 21 25 96 58 74 and show procedure step-by-step with the help of diagram start to end

Ans:



12. Show the effect of PUSH and POP operation on the stack of size 10.

PUSH(10)

PUSH(20)

POP

PUSH(30)

Ans: Initial Stack empty

	stack[9]	
	stack[8]	
	stack[7]	
	stack[6]	
	stack[5]	
	stack[4]	
	stack[3]	
	stack[2]	
	stack[1]	
	stack[0]	top= -1

Step 1:

PUSH(0)
top=top+1 stack[0]=10

	stack[9]	
	stack[8]	
	stack[7]	
	stack[6]	
	stack[5]	
	stack[4]	
	stack[3]	
	stack[2]	
	stack[1]	
10	stack[0]	top=0



Step 2:

PUSH(0)
top=top+1 stack[1]=20

	stack[9]	
	stack[8]	
	stack[7]	
	stack[6]	
	stack[5]	
	stack[4]	
	stack[3]	
	stack[2]	
20	stack[1]	top=1
10	stack[0]	

Step 3:

POP

top=top-1 20 is deleted

	stack[9]	
	stack[8]	
	stack[7]	
	stack[6]	
	stack[5]	
	stack[4]	
	stack[3]	
	stack[2]	
	stack[1]	
10	stack[0]	top=0

Step 4:

PUSH(0)

top=top+1

stack[1]=30

	stack[9]	
	stack[8]	
	stack[7]	
	stack[6]	
	stack[5]	
	stack[4]	
	stack[3]	
	stack[2]	
30	stack[1]	top=1
10	stack[0]	



13. Compare Linked List and Array (any 4 points)

Ans:

Linked List	Array
Array is a collection of elements of similar data type.	Linked List is an ordered collection of elements of same type, which are connected to each other using pointers.
Array supports Random Access, which means elements can be accessed directly using their index, like arr[0] for 1st element, arr[6] for 7th element etc.	Linked List supports Sequential Access, which means to access any element/node in a linked list; we have to sequentially traverse the complete linked list, up to that element.

<p>Hence, accessing elements in an array is fast with a constant time complexity of $O(1)$. In array, Insertion and Deletion operation takes more time, as the memory locations are consecutive and fixed.</p>	<p>To access nth element of a linked list, time complexity is $O(n)$. In case of linked list, a new element is stored at the first free and available memory location, with only a single overhead step of storing the address of memory location in the previous node of linked list. Insertion and Deletion operations are fast in linked list.</p>
<p>Memory is allocated as soon as the array is declared, at compile time. It's also known as Static Memory Allocation.</p>	<p>Memory is allocated at runtime, as and when a new node is added. It's also known as Dynamic Memory Allocation.</p>
<p>In array, each element is independent and can be accessed using its index value</p>	<p>In case of a linked list, each node/element points to the next, previous, or maybe both nodes.</p>
<p>Array can single dimensional, two dimensional or multidimensional</p>	<p>Linked list can be Linear (Singly), Doubly or Circular linked list.</p>
<p>Size of the array must be specified at time of array declaration.</p>	<p>Size of a Linked list is variable. It grows at runtime, as more nodes are added to it.</p>

6 MARKS QUESTION
(WINTER-18)

1. Write algorithm for performing push and pop operations on stack.

Ans Push algorithm: -

Max is maximum size of stack.

Step 1: [Check for stack full/ overflow]

If stack_top is equal to max-1 then

Display output as "Stack Overflow" and
return to calling function

Otherwise

Go to step 2

Step 2: [Increment stack_top]

Increment stack top pointer by one.

stack_top=stack_top +1;

Step 3: [Insert element]

stack [stack_top] = item;

Step 4: return to calling function

Pop algorithm: -

Max is maximum size of stack.

Step 1: [Check for stack empty/underflow]

If stack_top is equal to -1 then

Display output as "Stack Underflow" and
return to calling function

Otherwise

Go to step 2

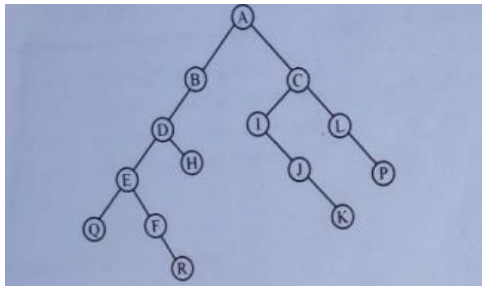
Step 2: [delete element]

stack [stack_top] = item;

Step 3: [Decrement stack_top]
 Decrement stack top pointer by one.
 stack_top=stack_top -1;
 Step 4: return to calling function

2. For given binary tree write in-order, pre-order and post-order traversal.

Ans:



Inorder Traversal: Q,E,F,R,D,H,B,A,I,J,K,C,L,P
 Preorder Traversal: A,B,D,E,Q,F,R,H,C,I,J,K,L,P
 Postorder Traversal: Q,R,F,E,H,D,B,K,J,I,P,L,C,A

3. Write an algorithm to insert an element at the beginning and end of linked list.

Ans Algorithm to insert an element at the beginning of linked list:

1. Start
2. Create the node pointer *temp Struct node * temp
3. Allocate address to temp using malloc
temp = malloc(sizeof(struct node));
4. Check whether temp is null, if null then
Display "Overflow"
Else
temp-> info=data
temp-> next=start
5. Start=temp
6. stop

Algorithm to insert an element at the end of linked list:

1. Start
2. Create two node pointers *temp, *q struct node * temp, *q;
3. q= start
4. Allocate address to temp using malloc


```
temp = malloc(sizeof(struct node));  
5. Check whether temp is null, if null  
then
```

```
Display "Overflow"
```

```
else
```

```
temp->info=data
```

```
temp->next=null
```

```
6. While(q->next!=null)
```

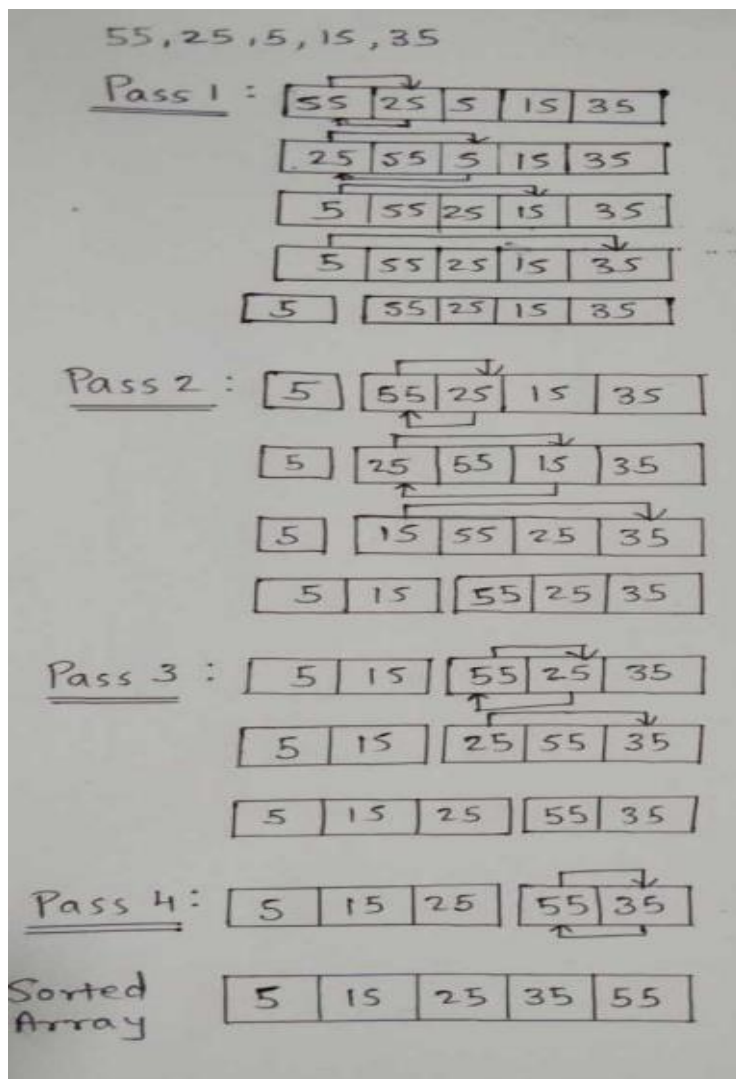
```
q= q-> next
```

```
7. q->next= temp
```

```
8. stop
```

4. Describe working of selection sort method. Also sort given input list in ascending order using selection sort input list:- 55, 25, 5, 15, 35.

Ans Working of Selection sort: Selection Sort algorithm is used to arrange a list of elements in a particular order (Ascending or Descending). In selection sort, the first element in the list is selected and it is compared repeatedly with remaining all the elements in the list. If any element is smaller than the selected element (for ascending order), then both are swapped. Then we select the element at second position in the list and it is compared with remaining all elements in the list. If any element is smaller than the selected element, then both are swapped. This procedure is repeated till the entire list is sorted.



5. Define the term recursion. Write a program in C to display factorial of an entered number using recursion.

Ans Definition: Recursion is the process of calling function by itself. A recursive function body contains function call statement that calls itself repeatedly.

Program:

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

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

```
int fact(int n);
```

```
void main()
```

```

{
int n;

clrscr();

printf("\nThe factorial of % is = %d",n,fact(n));

getch();

}

int fact(int n)
{
if(n==1)
return 1;

else
return(n*fact(n-1));

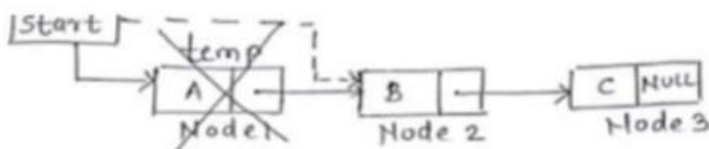
}

```

6. Describe procedure to delete an element from singly linked list using diagram.

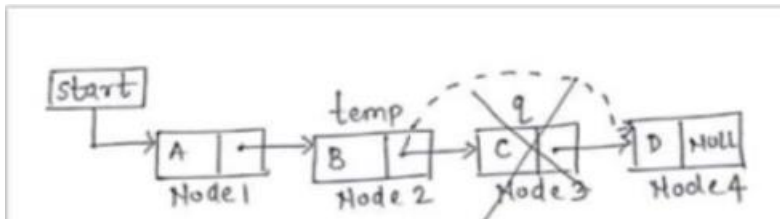
Ans In a linear linked list, a node can be deleted from the beginning of list, from in between positions and from end of the list.

Delete a node from the beginning:-



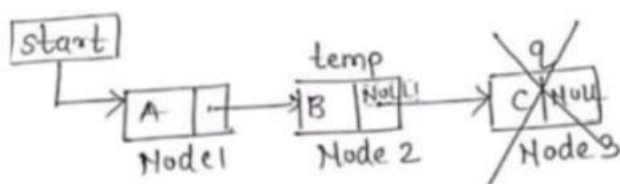
Node to be deleted is node1. Create a temporary node as 'temp'. Set 'temp' node with the address of first node. Store address of node 2 in header pointer 'start' and then delete 'temp' pointer with free function. Deleting temp pointer deletes the first node from the list.

Delete a node from in between position:-



Node to be deleted is node 3. Create a temporary node as 'temp' and 'q'. Set 'temp' node with the address of first node. Traverse the list up to the previous node of node 3 and mark the next node (node 3) as 'q'. Store address from node 'q' into address field of 'temp' node. Then delete 'q' pointer with free function. Deleting 'q' pointer deletes the node 3 from the list.

Delete a node from the end:-



Node to be deleted is node 3. Create a temporary node as 'temp' and 'q'. Set 'temp' node with the address of first node. Traverse the list up to the second last node and mark the last node as 'q'. Store NULL value in address field of 'temp' node and then delete 'q' pointer with free function. Deleting q pointer deletes the last node from the list.

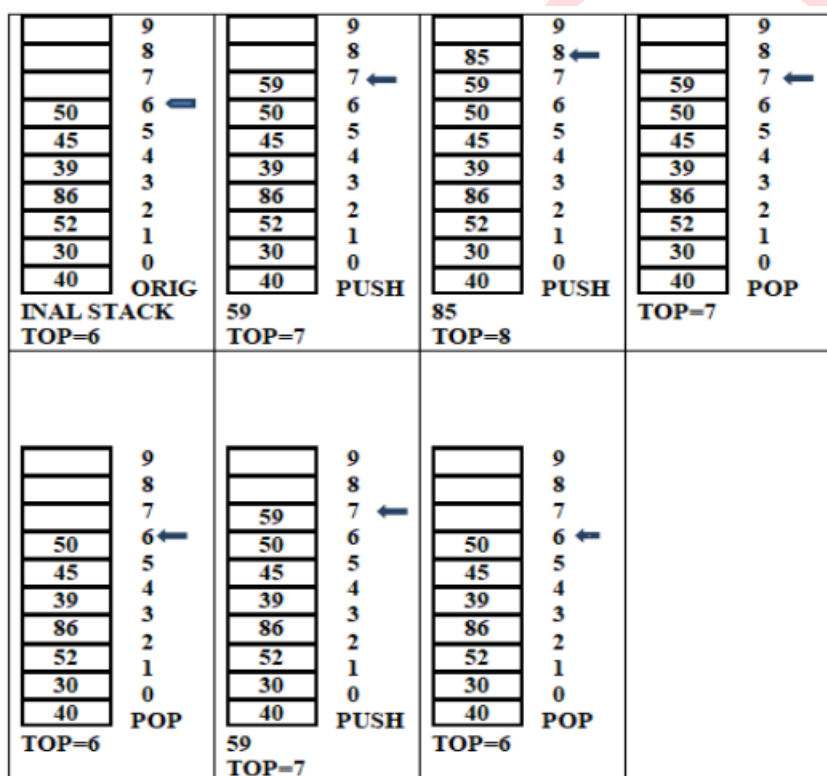
(SUMMER-19)

1. Show the effect of PUSH and POP operation on to the stack of size 10. The stack contains 40, 30, 52, 86, 39, 45, 50 with 50 being at top of the stack. Show diagrammatically the effect of:

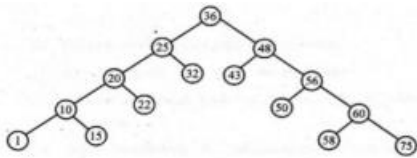
- (i) PUSH 59
- (ii) PUSH 85
- (iii) POP
- (iv) POP
- (v) PUSH 59
- (vi) POP

Sketch the final structure of stack after performing the above said operations.

Ans:



2. Traverse the following tree by the in-order, pre-order and postorder methods:



Ans:

INORDER (LVR) 1,10,15,20,22,25,32,36,43,48,50,56,58,60,75

PREORDER (VLR) 36,25,20,10,1,15,22,32,48,43,56,50,60,58,75

POST ORDER (LRV) 1,15,10,22,20,32,25,43,50,58,75,60,56,48,36

3. Write an algorithm to count number of nodes in singly linked list.

Ans: Let start is pointer variable which always stores address of first node in single linked list.

If single linked list is empty then start will point to NULL.

q is pointer variable used to store address of nodes in single linked list.

Step 1: Start

Step 2: [Assign starting address of single linked list to pointer q]

q=start

Step 3: [Initially set count of nodes in Linked list as zero]

count=0

Step 4: [Check if Linked list empty or not]

if start==NULL

Display "Empty Linked List"

go to step 6.

Step 5: [Count number of nodes in single linked list]

while q!=NULL count++ and

q=q->next;

Step 6: Display count (total number of nodes in single linked list)

Step 7: stop

4. Sort the following numbers in ascending order using Bubble sort.

Ans: Given numbers: 29, 35, 3, 8, 11, 15, 56, 12, 1, 4, 85, 5 & write the output after each interaction.

Pass 1 :

Enter no of elements :12

Enter array elements :29 35 3 8 11 15 56 12 1 4 85 5

Unsorted Data: 29 35 3 8 11 15 56 12 1 4 85 5

After pass 1 : 29 35 3 8 11 15 56 12 1 4 85 5
 After pass 1 : 29 3 35 8 11 15 56 12 1 4 85 5
 After pass 1 : 29 3 8 35 11 15 56 12 1 4 85 5
 After pass 1 : 29 3 8 11 35 15 56 12 1 4 85 5
 After pass 1 : 29 3 8 11 15 35 56 12 1 4 85 5
 After pass 1 : 29 3 8 11 15 35 56 12 1 4 85 5
 After pass 1 : 29 3 8 11 15 35 12 56 1 4 85 5
 After pass 1 : 29 3 8 11 15 35 12 1 56 4 85 5
 After pass 1 : 29 3 8 11 15 35 12 1 4 56 85 5
 After pass 1 : 29 3 8 11 15 35 12 1 4 56 85 5
 After pass 1 : 29 3 8 11 15 35 12 1 4 56 5 85

Pass 2

After pass 2 : 3 29 8 11 15 35 12 1 4 56 5 85
 After pass 2 : 3 8 29 11 15 35 12 1 4 56 5 85
 After pass 2 : 3 8 11 29 15 35 12 1 4 56 5 85
 After pass 2 : 3 8 11 15 29 35 12 1 4 56 5 85
 After pass 2 : 3 8 11 15 29 35 12 1 4 56 5 85
 After pass 2 : 3 8 11 15 29 12 35 1 4 56 5 85
 After pass 2 : 3 8 11 15 29 12 1 35 4 56 5 85
 After pass 2 : 3 8 11 15 29 12 1 4 35 56 5 85
 After pass 2 : 3 8 11 15 29 12 1 4 35 56 5 85
 After pass 2 : 3 8 11 15 29 12 1 4 35 5 56 85

Pass 3

After pass 3 : 3 8 11 15 29 12 1 4 35 5 56 85
 After pass 3 : 3 8 11 15 29 12 1 4 35 5 56 85
 After pass 3 : 3 8 11 15 29 12 1 4 35 5 56 85
 After pass 3 : 3 8 11 15 29 12 1 4 35 5 56 85
 After pass 3 : 3 8 11 15 12 29 1 4 35 5 56 85
 After pass 3 : 3 8 11 15 12 1 29 4 35 5 56 85
 After pass 3 : 3 8 11 15 12 1 4 29 35 5 56 85
 After pass 3 : 3 8 11 15 12 1 4 29 35 5 56 85
 After pass 3 : 3 8 11 15 12 1 4 29 5 35 56 85

Pass 4

After pass 4 : 3 8 11 15 12 1 4 29 5 35 56 85
 After pass 4 : 3 8 11 15 12 1 4 29 5 35 56 85
 After pass 4 : 3 8 11 15 12 1 4 29 5 35 56 85
 After pass 4 : 3 8 11 12 15 1 4 29 5 35 56 85

After pass 4 : 3 8 11 12 1 15 4 29 5 35 56 85
 After pass 4 : 3 8 11 12 1 4 15 29 5 35 56 85
 After pass 4 : 3 8 11 12 1 4 15 29 5 35 56 85
 After pass 4 : 3 8 11 12 1 4 15 5 29 35 56 85

Pass 5

After pass 5 : 3 8 11 12 1 4 15 5 29 35 56 85
 After pass 5 : 3 8 11 12 1 4 15 5 29 35 56 85
 After pass 5 : 3 8 11 12 1 4 15 5 29 35 56 85
 After pass 5 : 3 8 11 1 12 4 15 5 29 35 56 85
 After pass 5 : 3 8 11 1 4 12 15 5 29 35 56 85
 After pass 5 : 3 8 11 1 4 12 15 5 29 35 56 85
 After pass 5 : 3 8 11 1 4 12 5 15 29 35 56 85

Pass 6

After pass 6 : 3 8 11 1 4 12 5 15 29 35 56 85
 After pass 6 : 3 8 11 1 4 12 5 15 29 35 56 85
 After pass 6 : 3 8 1 11 4 12 5 15 29 35 56 85
 After pass 6 : 3 8 1 4 11 12 5 15 29 35 56 85
 After pass 6 : 3 8 1 4 11 12 5 15 29 35 56 85
 After pass 6 : 3 8 1 4 11 5 12 15 29 35 56 85

Pass 7

After pass 7 : 3 8 1 4 11 5 12 15 29 35 56 85
 After pass 7 : 3 1 8 4 11 5 12 15 29 35 56 85
 After pass 7 : 3 1 4 8 11 5 12 15 29 35 56 85
 After pass 7 : 3 1 4 8 11 5 12 15 29 35 56 85
 After pass 7 : 3 1 4 8 5 11 12 15 29 35 56 85

Pass 8

After pass 12 : 1 3 4 8 5 11 12 15 29 35 56 85

Sorted elements are 1 3 4 8 5 11 12 15 29 35 56 85

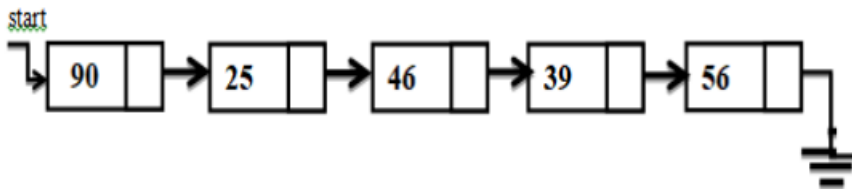
5. Evaluate the following postfix expression: $5\ 7\ +\ 6\ 2\ -\ *$

Ans:

Symbols to be scanned	STACK					Expression Evaluation and Result
	4	3	2	1	0	
5					5	----
7				7	5	----
+					12	7+5=12
6				6	12	----
2			2	6	12	6-2=4
-				4	12	----
*					48	12*4

6. Create a singly linked list using data fields 90, 25, 46, 39, 56. Search a node 40 from the SLL and show procedure step-by-step with the help of diagram from start to end.

To Search a data field in singly linked list, need to start searching the data field from first node of singly linked list. ORIGINAL LIST:



SEARCHING A NODE STEP 1:

Compare 40 with 90

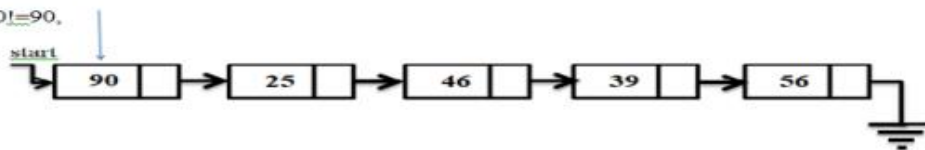
40!=90

SEARCHING A NODE

STEP 1:

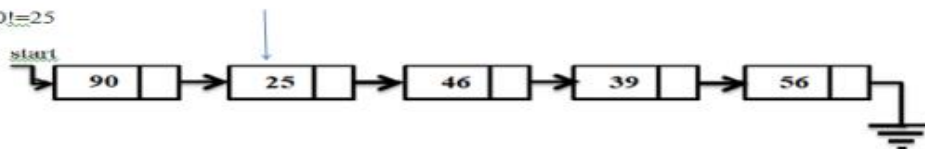
Compare 40 with 90

$40 \neq 90$,



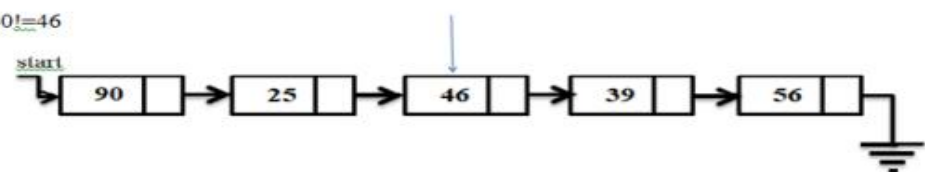
STEP 2:

$40 \neq 25$



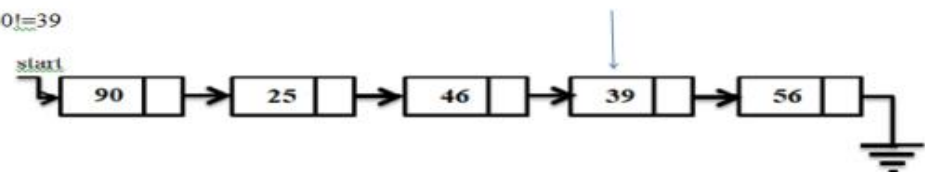
STEP 3:

$40 \neq 46$



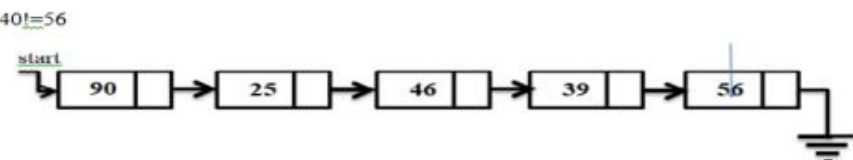
STEP 4:

$40 \neq 39$



Step 5:

$40 \neq 56$



Node not found. Search unsuccessful

(WINTER-19)

1. Implement a 'C' program to insert element into the queue and delete the element from the queue.

Ans:

```
#include<stdio.h>
#include<conio.h>
#define max 5
void main()
{
int a[max],front,rear,no,ch,i;
clrscr();
front=rear=-1;
do
{
printf("\n 1.INSERT");
printf("\t 2.DELETE");
printf("\t 3.EXIT");
printf("\n\n ENTER YOUR CHOICE:- ");
scanf("%d",&ch);
switch(ch)
{
case 1:
printf("\n ENTER ITEM TO BE INSERTED :- ");
scanf("%d",&no);
if(rear==max-1)
{
printf ("\n QUEUE IS FULL.");
```

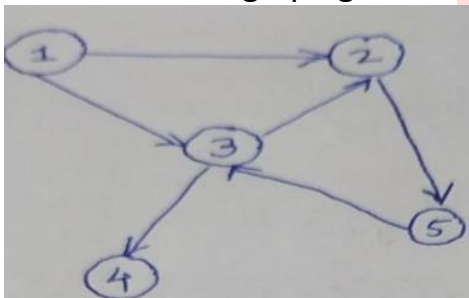


```

break;
}
rear=rear+1;
a[rear]=no;
if(front==-1)
front=0;
break;
case 2:
if(front==-1)
{
printf("\n QUEUE IS EMPTY.");
break;
}
no=a[front];
printf("\n DELETED ELEMENT IS:- %d",no);
if(front==rear)
front=rear=-1;
else
front=front+1;
break;
case 3:
exit(0);
}
printf("\n\n DO YOU WANT TO CONTINUE:(1 FOR YES/2 FOR NO):-");
scanf("%d",&ch);
}while(ch==1);
getch();
}

```

2. Consider the graph given in following figure and answer given questions.



- 1) All simple path from 1 to 5
- 2) In-degree of and out-degree of 4
- 3) Give Adjacency matrix for the given graph.
- 4) Give Adjacency list representation of the given graph

Ans:

- i) Nodes: 1-2-5
- ii) Nodes: 1-3-2-5 2)

In degree of node 4- 1, Out degree of node 4 - 0

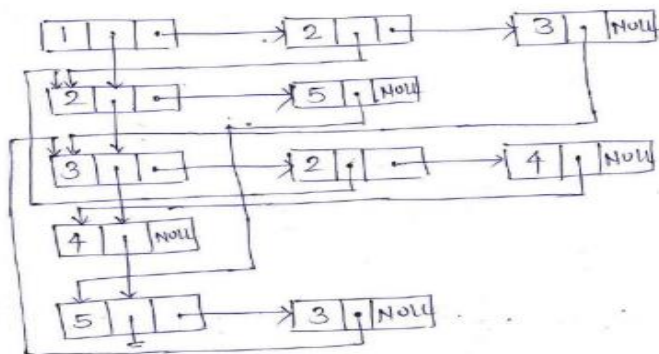
3) Correct adjacency matrix:

$$A = \begin{matrix} & \begin{matrix} 1 & 2 & 3 & 4 & 5 \end{matrix} \\ \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{matrix} & \begin{bmatrix} 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix} \end{matrix}$$

b) Adjacency list representation

Node	Adjacent nodes
1	2,3
2	5
3	2,4
4	NIL
5	3

Representation:



3. Write an algorithm to search a particular node in the give linked list.

Ans Assumption: Node contains two fields: info and next pointer start pointer : Header node that stores address of first node

step 1: start

step 2: Declare variable no, flag and pointer temp

step 3: Input search element

step 4: Initialize pointer temp with the address from start pointer.

(temp=start), flag with 0

step 5: Repeat step 6

till

temp != NULL

step 6: compare: temp->info = no then set flag=1 and

go to step 7
 otherwise increment pointer temp and
 go to step5
 step 7: compare: flag=1 then display "Node found"
 otherwise
 display "node not found"
 step 8: stop

4. Elaborate the steps for performing selection sort for given elements of array.

A={37,12,4,90,49,23,-19}

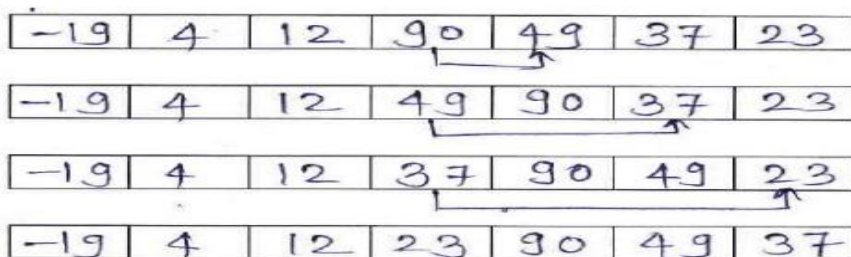
Ans:



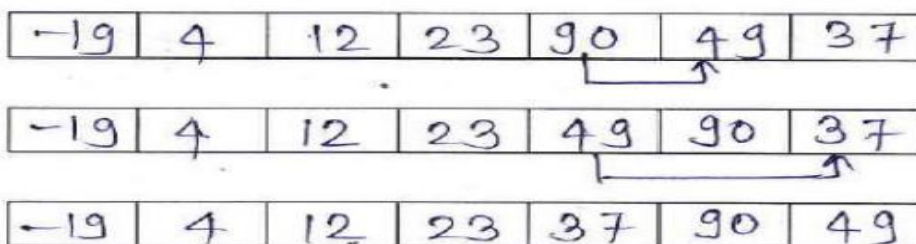
Pass 3



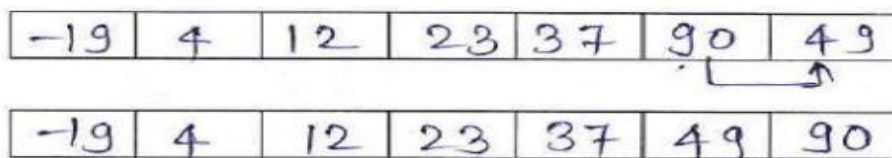
Pass 4



Pass 5



Pass 6



5. Explain the concept of recursion using stack.

Ans: Recursion is a process of calling a function by itself. a recursive function body contains a function call statement that calls itself repetitively. Recursion is an application of stack. When a recursive function calls itself from body, stack is used to store temporary data handled by the function in every iteration.

Example: function call from main() :

fact(n);

// consider n=5 Function definition:

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

In the above recursive function a function call fact (n-1) makes a recursive call to fact function. Each time when a function makes a call to itself, it save its current status in stack and then executes next function call. When fact () function is called from main function, it initializes n with 5. Return statement inside function body executes a recursive function call. In this call, first value of n is stored using push () operation in stack (n=5) and a function is called again with value 4(n-1). In each call, value of n is push into the stack and then it is reduce by 1 to send it as argument to recursive call. When a function is called with n=1, recursive process stops. At the end all values from stack are retrieved one by one using pop () operation to perform multiplication to calculate factorial of number.

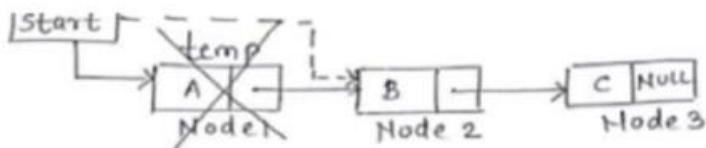
f(1) true return 1;	POP					
f(2) false return 2*f(1)	f(2) false return 2*1	POP				
f(3) false return 3*f(2)	f(3) false return 3*f(2)	f(3) false return 3*2	POP			
f(4) false return 4*f(3)	f(4) false return 4*f(3)	f(4) false return 4*f(3)	f(4) false return 4*6	POP		
f(5) // line 1 false return 5*f(4)	f(5) // line 1 false return 5*f(4)	f(5) // line 1 false return 5*f(4)	f(5) // line 1 false return 5*f(4)	f(5) // line 1 false return 5*24	POP	
main() y = f(5)	main() y = f(5)	main() y = f(5)	main() y = f(5)	main() y = f(5)	main() y = 120	POP

In the above diagram, first column shows result of push operation after each recursive call execution. Next columns shows result of pop operation for calculating factorial.

6. Show with suitable diagrams how to delete a node from singly linked list at the beginning, in between and at the end of the list.

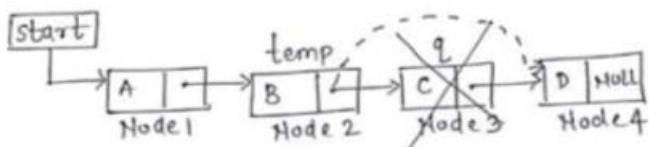
Ans: In a linear linked list, a node can be deleted from the beginning of list, from in between positions and from end of the list.

Delete a node from the beginning:-



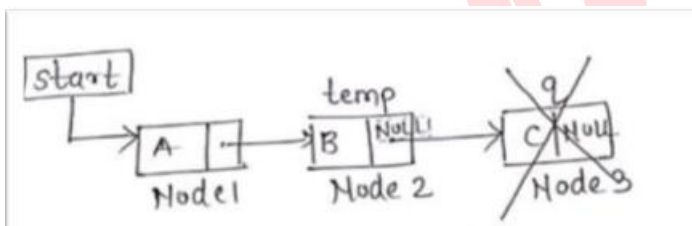
Node to be deleted is node1. Create a temporary node as 'temp'. Set 'temp' node with the address of first node. Store address of node 2 in header pointer 'start' and then delete 'temp' pointer with free function. Deleting temp pointer deletes the first node from the list.

Delete a node from in between position:-



Node to be deleted is node3. Create a temporary node as 'temp' and 'q'. Set 'temp' node with the address of first node. Traverse the list up to the previous node of node 3 and mark the next node (node3) as 'q'. Store address from node 'q' into address field of 'temp' node. Then delete 'q' pointer with free function. Deleting 'q' pointer deletes the node 3 from the list.

Delete a node from the end:-



Node to be deleted is node 3. Create a temporary node as 'temp' and 'q'. Set 'temp' node with the address of first node. Traverse the list up to the second last node and mark the last node as 'q'. Store NULL value in address field of 'temp' node and then delete 'q' pointer with free function. Deleting q pointer deletes the last node from the list.