Array Operations

```
int i,a[20],n,e,top=-1,MAX=20,p;
```

1 . for Insert by choice

```
cout<<"Enter no of Elements/Items do you want to enter :";
cin>>n;
cout<<"Enter elements:\n";
for(i=0;i<n;i++)
{  cout<<"A["<<i<<"]: ";cin>>a[i];
   top++;  }
```

2. for insert in End

```
cout<<"Enter Elemant : ";
cin>>e;
if(top>=MAX)
cout<<"Overflow";
else
{  top++;
   a[top]=e; }
```

3 for Insert in beg

```
cout<<"Enter Elemant : "; cin>>e;
if(top>=MAX) cout<<"Overflow";
else
{ for(i=top+1;i>=0;i--)
  {
  a[i]=a[i-1];
  }
 a[0]=e;
 }
 top+=1;
```

4 for display Array

```
for(i=0;i<=top;i++) cout<<"[ "<<a[i]<<" ] ";
```

5 for Enter element on selected position

```
cout<<"Enter Position :";cin>>p;
if(p<top||top==-1)
{
cout<<"Enter Elemant : "; cin>>e;
if(top>=MAX) cout<<"Overflow";
else
{
for(i=top+1;i>=p;i--)
{  a[i]=a[i-1];  }
 a[p]=e;
 }
 top+=1;
 }
 else cout<<"Position is not valid";
 }
```

6 delete from beg

```
if(top<0)
cout<<"\nUnderflow/Array is empty\n";
else
{
cout<<"\n The Array before Delete \n";
for(i=0;i<=top;i++)  cout<<"[ "<<a[i]<<" ] ";
for(i=0;i<top;i++)
{
a[i]=a[i+1];
}
top-=1;
}
cout<<"\n The Array After Delete \n";
for(i=0;i<=top;i++)   cout<<"[ "<<a[i]<<" ] ";
```

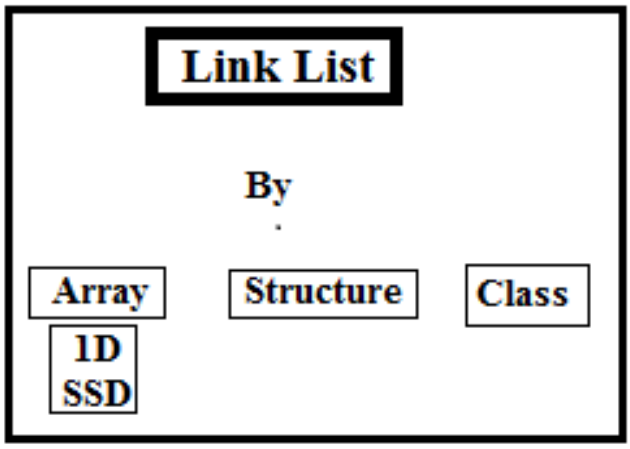
7 delete from End

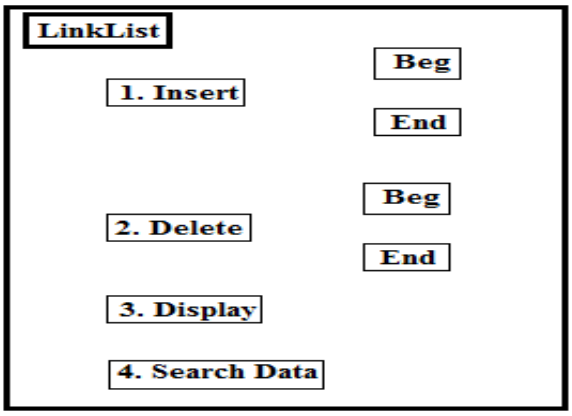
```
if(top<0)
cout<<"\nUnderflow/Array is empty\n";
else
{
cout<<"\n The Array before Delete \n";
for(i=0;i<=top;i++) cout<<"[ "<<a[i]<<" ] ";
top-=1;
cout<<"\n The Array After Delete \n";
for(i=0;i<=top;i++)
{
cout<<"[ "<<a[i]<<" ] ";
}
```

8 delete from selected position

```
if(p<top)
{  if(top<0) cout<<"\nUnderflow  \n";
   else
   {  cout<<"\n The Array before Delete \n";
      for(i=0;i<=top;i++) cout<<"[ "<<a[i]<<" ] ";
   for(i=p;i<top;i++)
   {  a[i]=a[i+1];  }
   top-=1;
   }
cout<<"\n The Array After Delete \n";
for(i=0;i<=top;i++)  cout<<"[ "<<a[i]<<" ] ";
}
else
cout<<"Deletion not possible on that Position is not valid\n";
```

Link List Using [Structure and Class], Array, Stack, Queue





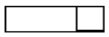
Step 1/2: Create Structure

```
struct NODE
{
int info;  //char ch,W[25];
NODE *Next;  // NODE *Link;
}*p,*s,*temp;
```

Create NODE and it's Object

Step 3:Create Blank Node

```
*newNode = new NODE;
```

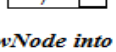


Step 4: ***Input Data/Info***

```
int element;  //char s[30];
cin>>element;  //gets(s);
element=7;// s="Apple";
```

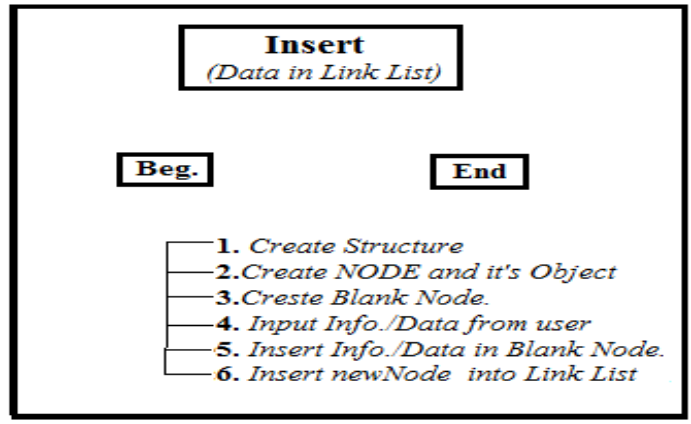
Step 5: ***Insert Data/Info in Blank Node***

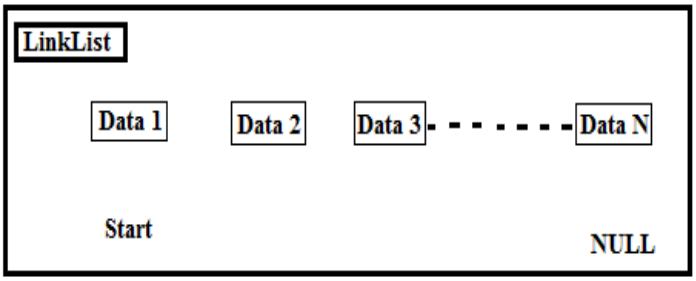
```
newNode->Info=element;//Info/Data, (string) s
newNode->Next=NULL;//Next/Link
```

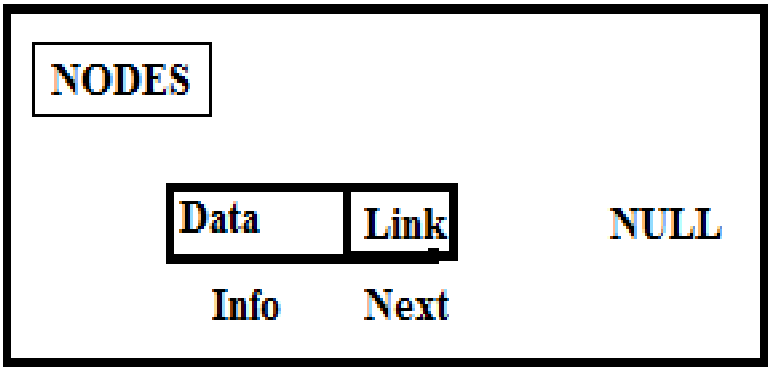


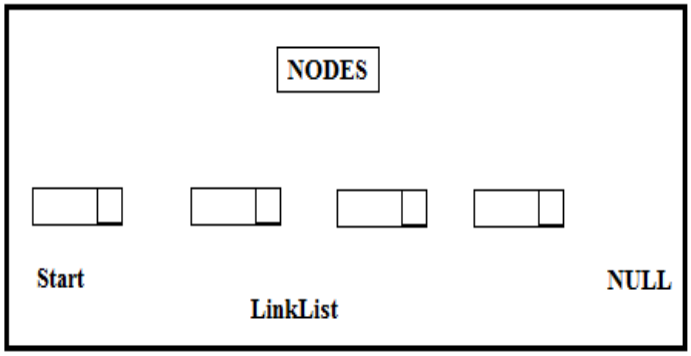
Step 6 :Insert newNode into List

Begning/End



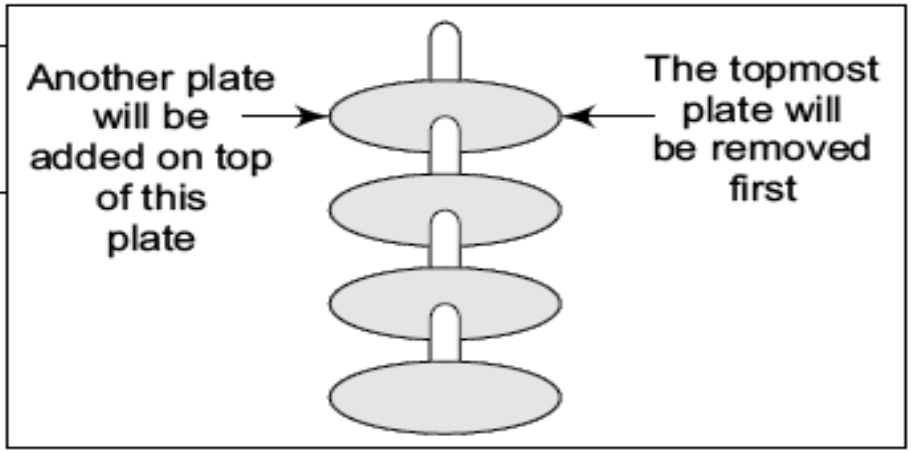






Stacks

A stack is called a LIFO (Last-In-First-Out)



ARRAY REPRESENTATION OF STACKS

A	AB	ABC	ABCD	ABCDE					
0	1	2	3	TOP = 4	5	6	7	8	9

TOP MAX–1

TOP = NULL stack is empty	TOP = MAX–1 stack is full.

Push Operation

```
Step 1: IF TOP = MAX-1
            PRINT "OVERFLOW"
            Goto Step 4
        [END OF IF]
Step 2: SET TOP = TOP + 1
Step 3: SET STACK[TOP] = VALUE
Step 4: END
```

```
if(top == MAX-1)
{ printf("\n STACK OVERFLOW");
  }
else
{ top++;
  st[top] = val; }
```

Pop Operation

```
Step 1: IF TOP = NULL
            PRINT "UNDERFLOW"
            Goto Step 4
        [END OF IF]
Step 2: SET VAL = STACK[TOP]
Step 3: SET TOP = TOP - 1
Step 4: END
```

```
if(top == -1)
{
printf("\n STACK UNDERFLOW");
}
else { val = st[top];
       top--;  }
```

Display Operation

```
if(top == -1)
printf("\n STACK IS EMPTY");
else
{
for(i=top;i>=0;i--)
printf("\n %d",st[i]);
printf("\n"); // Added for formatting purposes
}
```

Queues

ARRAY REPRESENTATION OF QUEUES

FRONT = -1
REAR = -1

A queue is a FIFO (First-In, First-Out)

12	9	7	18	14	36				
0	1	2	3	4	5	6	7	8	9

REAR = MAX – 1,
overflow

Push Operation

```
Step 1: IF REAR = MAX-1
            Write OVERFLOW
            Goto step 4
        [END OF IF]
Step 2: IF FRONT = -1 and REAR = -1
            SET FRONT = REAR = 0
        ELSE
            SET REAR = REAR + 1
        [END OF IF]
Step 3: SET QUEUE[REAR] = NUM
Step 4: EXIT
```

```
if(rear == MAX-1)
printf("\n OVERFLOW");
else if(front == -1 && rear == -1)
front = rear = 0;
else
rear++;
queue[rear] = num;
}
```

Pop Operation

```
Step 1: IF FRONT = -1 OR FRONT > REAR
            Write UNDERFLOW
        ELSE
            SET VAL = QUEUE[FRONT]
            SET FRONT = FRONT + 1
        [END OF IF]
Step 2: EXIT
```

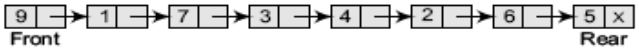
```
if(front == -1 || front>rear)
  printf("\n UNDERFLOW");
else
{ val = queue[front];
  front++;
  if(front > rear)
  front = rear = -1;
  return val; }
```

Display Operation

```
if(front == -1 || front > rear)
printf("\n QUEUE IS EMPTY");
else
{
for(i = front;i <= rear;i++)
printf("\t %d", queue[i]);
}
```

```
printf("\n Enter the number :");
scanf("%d", &num);
```

LINKED REPRESENTATION OF QUEUES




```
printf("\n Enter the number the queue:");
scanf("%d", &val);
    ptr = (struct node*)malloc(sizeof(struct node));
```

```
q -> rear = NULL;
q -> front = NULL;
```

```
Step 1: Allocate memory for the new node and name
        it as PTR
Step 2: SET PTR -> DATA = VAL
Step 3: IF FRONT = NULL
            SET FRONT = REAR = PTR
            SET FRONT -> NEXT = REAR -> NEXT = NULL
        ELSE
            SET REAR -> NEXT = PTR
            SET REAR = PTR
            SET REAR -> NEXT = NULL
        [END OF IF]
Step 4: END
```

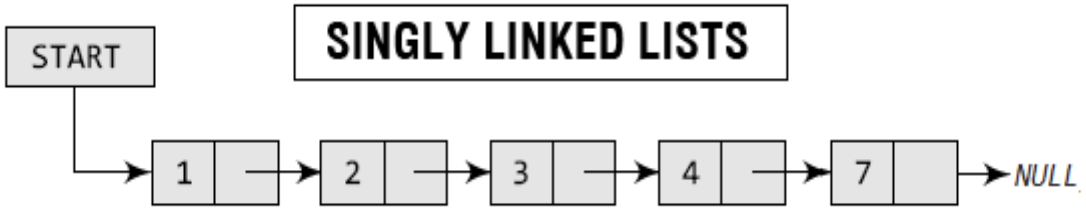
```
ptr -> data = val;
if(q -> front == NULL)
{ q -> front = ptr;
  q -> rear = ptr;
  q -> front -> next = q -> rear -> next = NULL;
}
else
{ q -> rear -> next = ptr;
  q -> rear = ptr;
  q -> rear -> next = NULL; }
```

```
ptr = q -> front;
if(ptr == NULL)
printf("\n QUEUE IS EMPTY");
else
{  printf("\n");
  while(ptr!=q -> rear)
  {  printf("%d\t", ptr -> data);
     ptr = ptr -> next; }
printf("%d\t", ptr -> data); }
```

```
Step 1: IF FRONT = NULL
            Write "Underflow"
            Go to Step 5
        [END OF IF]
Step 2: SET PTR = FRONT
Step 3: SET FRONT = FRONT -> NEXT
Step 4: FREE PTR
Step 5: END
```

```
ptr = q -> front;
if(q -> front == NULL)
printf("\n UNDERFLOW");
else
{ q -> front = q -> front -> next;
  free(ptr); }
```

SINGLY LINKED LISTS




```
printf("\n Enter the data : ");
scanf("%d", &num);
new_node = (struct node*)malloc(sizeof(struct node));
new_node -> data=num;
```

Insert List

```
if(start==NULL)
{
new_node -> next = NULL;
start = new_node;
}
else
{
ptr=start;
while(ptr->next!=NULL)
ptr=ptr->next;
ptr->next = new_node;
new_node->next=NULL;
}
```

Insert Begning

```
new_node -> data = num;
new_node -> next = start;
start = new_node;
```

Insert End

```
ptr = start;
while(ptr -> next != NULL)
ptr = ptr -> next;
ptr -> next = new_node;
```

Insert After

```
ptr = start;
preptr = ptr;
while(preptr -> data != val)
{
preptr = ptr;
ptr = ptr -> next;
}
preptr -> next=new_node;
new_node -> next = ptr;
```

Insert Before

```
ptr = start;
while(ptr -> data != val)
{
preptr = ptr;
ptr = ptr -> next;
}
preptr -> next = new_node;
new_node -> next = ptr;
```

Delete Begning

```
ptr = start;
start = start -> next;
free(ptr);
```

Display List

```
ptr = start;
while(ptr != NULL)
{
printf("\t %d", ptr -> data);
ptr = ptr -> next;
}
```

Delete End

```
ptr = start;
while(ptr -> next != NULL)
{
preptr = ptr;
ptr = ptr -> next;
}
preptr -> next = NULL;
free(ptr);
```

Delete After Data/Val

```
ptr = start;
preptr = ptr;
while(preptr -> data != val)
{
preptr = ptr;
ptr = ptr -> next;
}
preptr -> next=ptr -> next;
free(ptr);
```

Delete Node Data/Val

```
ptr = start;
if(ptr -> data == val)
{
start = delete_beg(start);
return start;
}
else
{ while(ptr -> data != val)
  {
  preptr = ptr;
  ptr = ptr -> next;
  }
preptr -> next = ptr -> next;
free(ptr);
```

LINKED REPRESENTATION OF STACKS


```
ptr = (struct stack*)malloc(sizeof(struct stack));
ptr -> data = val;
```

Push Operation

```
Step 1: Allocate memory for the new
        node and name it as NEW_NODE
Step 2: SET NEW_NODE -> DATA = VAL
Step 3: IF TOP = NULL
            SET NEW_NODE -> NEXT = NULL
            SET TOP = NEW_NODE
        ELSE
            SET NEW_NODE -> NEXT = TOP
            SET TOP = NEW_NODE
        [END OF IF]
Step 4: END
```

```
if(top == NULL)
{ ptr -> next = NULL;
  top = ptr; }
else
{ ptr -> next = top;
  top = ptr; }
```

Pop Operation

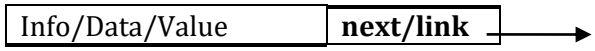
```
Step 1: IF TOP = NULL
            PRINT "UNDERFLOW"
            Goto Step 5
        [END OF IF]
Step 2: SET PTR = TOP
Step 3: SET TOP = TOP -> NEXT
Step 4: FREE PTR
Step 5: END
```

```
ptr = top;
if(top == NULL)
printf("\n STACK IS EMPTY");
else
{while(ptr != NULL)
 { printf("\n %d", ptr -> data);
   ptr = ptr -> next; } }
```

Display Operation

```
ptr = top;
if(top == NULL)
printf("\n STACK UNDERFLOW");
else
{ top = top -> next;
  printf("\n The value being deleted is: %d", ptr -> data);
  free(ptr); }
```

Node



1. Syntax Creating Node:

struct **<NODE Name >**
{
(Data member Declaration **)[Info/Data/Value**]
[Node inside object as pointer](**next/link)**
}<Object Name>;

Example:

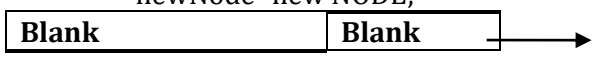
struct **NODE**
{
int Info;
NODE *next;
}*start,*save,*newNode,*ptr,*real,*front,*np;

2. Input element/value for node:

cin>>e; //e=7

3. Create blank node:

newNode=new NODE;



4. Insert info into node

newNode->info=e;
newNode->next=NULL;

LINK LIST

Insert New Node into link list :-in Beginning or in End

a. In Beginning:

If(start==NULL)
start =newNode;
else
{
save=start;
start=newNode;
[start->next=save;]=[newNode->next=save;]
}

b. In End:

If(start==NULL)
start =real=newNode;
else
{
real->next=newNode;
real=newNode;
}

6.Delete Node

If(start==NULL)
cout<< "Underflow";//No Element present
else
start=start->next;

7. Traversal/Display Link List:

np=start;
while(np!==NULL)
{
cout<<np->info;
np=np->next;
}

Stack using LINK list

Push operation

If(top==NULL)
top =newNode;
else
{
save=top;
top=newNode;
[top->next=save;]=[newNode->next=save;]
}

POP operation

If(top ==NULL)
cout<< "Underflow";//No Element present
else
top = top ->next;

Traversal/Display Stack:

np= top;
while(np!==NULL)
{
cout<<np->info;
np=np->next;
}

Stack using Array

Push operation

If(top>=(N-1))
{
cout<< "OverFlow";
}
else
{
top ++;
STACK[top]=e;
}

POP operation

If(top<=-1))
{
cout<< "UnderFlow";
}
else
{
top --;
}

Display

```
for(int i=0;i<=top;i++)
```

PUSH:

```
If(front==NULL)
 front =real=newNode;
else
{
real->next=newNode;
real=newNode;
 }
```

POP:

```
If(front ==NULL)
```

Push operation

```
If(real>=(size-1))
 {
cout<< "OverFlow";
}
else if(real==-1)
{
front=real=0;
QUEUE[real]=e;
}
```

POP operation

```
If(front<=-1)
 {
cout<< "UnderFlow";
cout<< STACK[i];
```

Queue using LINK list

```
cout<< "Underflow";//No Element present
else
front = front ->next;
```

Traversal/Display:

```
np= front;
while(np!==NULL) p
{
cout<<np->info;
np=np->next;
}
```

Queue using Array

```
}
else
{
if(front==real) front=real=-1;
else front++;
}
```

Display

```
If(front<=-1))cout<<"No Item ";
else
{
for(int i=front;i<=real;i++)
cout<< QUEUE[i];

}
```

- **Structure and Class Operations are same**
- **Stack, Queue and Link List using Structure are same**
- **Stack and Queue using Array are same [Stack using Top/Queue Using Front and Real]**

1. Link list Insert[beg/end], delete, display menu program using structure

```
#include<stdio.h>
#include<iostream.h>
#include<conio.h>
struct NODE
{
int Info;
NODE *next;
}*start,*save,*newNode,*ptr,*real,*front,*np,*newptr;
NODE *CNN(int);
void inBeg(NODE*);
void inEnd(NODE*);
void Del();
void Dis(NODE*);
 void main()
{
start=real=NULL;
int n,e;
char ch= 'y';
while(ch== 'y'||ch== 'Y')
{
cout<< "Data Link List Operation's";
cout<< "1 for insert node in beginning of Link List";
cout<< "2 for insert node in End of Link List";
cout<< "3 for Delete node from beg of Link List";
cout<< "4 for Display all node from beg to end ";
cout<< "5  for exit ";
cin>>n;
switch(n)
{
case 1: cout<< "Enter ITEM for list : ";
        cin<<e;
        newptr=CNN(e);
        inBeg(newptr);
        break;
case 2: cout<< "Enter ITEM for list : ";
        cin<<e;
        newptr=CNN(e);
        inEnd(newptr);
        break;
case 3: Del();
        break;
case 4: Dis(start);
        break;
default cout<< "Thanks";
}
cout<< "Enter Y/N :";
```

```
cin<<ch;
}
}
getch();
}
NODE *CNN(int n)
{
ptr=new NODE;
ptr->info=n;
ptr->next=NULL;
return ptr;
}
void inBeg(NODE* newNode)
{
If(start==NULL)
 start =newNode;
else
{
save=start;
start=newNode;
[start->next=save;]/[ newNode->next=save;]
}
}
void inEnd(NODE* newNode)
{
If(start==NULL)
 start =real=newNode;
else
{
real->next=newNode;
real=newNode;
 }
}
void Del()
{
If(start==NULL)
cout<< "Underflow";//No Element present
else
start=start->next;
}
void Dis(NODE* np)
{
while(np!==NULL)
{
cout<<np->info;
np=np->next;
}
}
```

2. STACK Operation using Structure

```
#include<stdio.h>
#include<iostream.h>
#include<conio.h>
struct STACK
{
int Info;
STACK *next;
}*top,*save,*newNode,*ptr,*np,*newptr;
NODE *CNN(int);
void inBeg(STACK *);
void inEnd(STACK *);
void Del();
void Dis(STACK *);
 void main()
{
top=real=NULL;
int n,e;
char ch= 'y';
while(ch== 'y'||ch== 'Y')
{
cout<< "STACK Operation's";
cout<< "1 for insert(PUSH) in STACK ";
cout<< "2 for Delete (POP) from STACK";
cout<< "3 for Display all STACK items ";
cout<< "4  for exit ";
cin>>n;
switch(n)
{
case 1: cout<< "Enter ITEM for STACK : ";
          cin<<e;
          newptr=CNN(e);
          inBeg(newptr);
          break;
case 2: Del();
          break;
case 3: Dis(top);
          break;
default cout<< "Thanks";
}
cout<< "Enter Y/N :";
cin<<ch;
}
}
getch();
}
NODE *CNN(int n)
{
ptr=new NODE;
ptr->info=n;
ptr->next=NULL;
return ptr;
}
void inBeg(STACK * newNode)
{
If(top==NULL)
 top =newNode;
else
{
save=top;
top=newNode;
[top->next=save;]/[ newNode->next=save;]
}
```

```
}
void Del()
{
If(top==NULL)
cout<< "Underflow";//No Element present
else
top=top->next;
}
```

```
void Dis(STACK * np)
{
while(np!==NULL)
{
cout<<np->info;
np=np->next;
}
}
```

3. STACK Operation using Array

```
#include<stdio.h>
#include<iostream.h>
#include<conio.h>
void PUSH(int [],int &,int );
void POP(int [],int &);
void DIS(int [],int &);
const int size=50
void main()
{
int  STACK[size],e, top=-1,n;
char ch= 'y';
while(ch== 'y'||ch== 'Y')
{
cout<< "STACK Operation's";
cout<< "1 for insert(PUSH) in STACK ";
cout<< "2 for Delete (POP) from STACK";
cout<< "3 for Display all STACK items ";
cout<< "4  for exit ";
cin>>n;
switch(n)
{
case 1: cout<< "Enter ITEM for STACK : ";
        cin<<e;
        PUSH(STACK ,top, e);
        break;
case 2: POP(STACK ,top);
        break;
case 3: Dis(top);
        break;
default cout<< "Thanks";
}
```

```
cout<< "Enter Y/N :";
cin<<ch;}}
getch();
}
void PUSH(int STACK [],int & top, int e )
{
If(top>=(size-1))
 {
cout<< "OverFlow";
}
else
{
top ++;
STACK[top]=e;
}
}
void POP(int STACK [],int & top)
{
if(top<=-1))
 {
cout<< "UnderFlow";
}
else
{
top --;}}
void DIS(int STACK [],int & top);
{
for(int i=0;i<=top;i++)
cout<< STACK[i];
}
```

4. QUEUE Operation using Structure

```
#include<stdio.h>
#include<iostream.h>
#include<conio.h>
struct QUEUE
{
int Info;
QUEUE *next;
}*start,*save,*newNode,*ptr,*real,*front,*np;
NODE *CNN(int);
void inEnd(QUEUE *);
void Del();void Dis(QUEUE *); void main()
{
top=real=NULL;
int n,e;
```

```
char ch= 'y';
while(ch== 'y'||ch== 'Y')
{
cout<< "QUEUE Operation's";
cout<< "1 for insert(PUSH) in QUEUE ";
cout<< "2 for Delete (POP) from QUEUE";
cout<< "3 for Display all QUEUE items ";
cout<< "4  for exit ";
cin>>n;
switch(n)
{
case 1: cout<< "Enter ITEM for list : ";
        cin<<e;
        newptr=CNN(e);
```

```
        inEnd(newptr);
        break;
case 2: Del();
        break;
case 3: Dis(front);
        break;
default cout<< "Thanks";
}
cout<< "Enter Y/N :";
cin<<ch;}}
getch();
}
NODE *CNN(int n)
{
ptr=new NODE;
ptr->info=n;
ptr->next=NULL;
return ptr;
}
void inBeg(QUEUE * newNode)
{
If(front==NULL)
 front =real=newNode;
else
{
real->next=newNode;
real=newNode; }}
void Del()
{
If(front ==NULL)
cout<< "Underflow";//No Element present
else
front = front ->next;
}
void Dis(QUEUE * np)
{
while(np!==NULL)
{
cout<<np->info;
np=np->next;}}
```

5. QUEUE Operation using Array

```
#include<stdio.h>
#include<iostream.h>
#include<conio.h>
void PUSH(int [],int );
void POP(int [],int );
void DIS(int [],int ,int);
const int size=50
int  QUEUE [size],front=-1,real=-1;
void main()
{
int e,n;
char ch= 'y';
while(ch== 'y'||ch== 'Y')
{
cout<< "QUEUE Operation's";
cout<< "1 for insert(PUSH) in QUEUE ";
cout<< "2 for Delete (POP) from QUEUE";
cout<< "3 for Display all QUEUE items ";
cout<< "4  for exit ";
cin>>n;
switch(n)
{
case 1: cout<< "Enter ITEM for QUEUE : ";
     cin<<e;
PUSH(QUEUE , e);
break;
case 2: POP(QUEUE );
break;
case 3: Dis(QUEUE,front,real);
break;
default cout<< "Thanks";
}
cout<< "Enter Y/N :";
cin<<ch;
}
}
getch();
}
void PUSH(int QUEUE [],int e )
{
 If(real>=(size-1))
 {
 cout<< "OverFlow";
 }
 else if(real==-1)
 {
 front=real=0;
 QUEUE[real]=e;
 }
}
void POP(int QUEUE)
{
 If(front<=-1)
 {
 cout<< "UnderFlow";}
 else
 {
 if(front==real) front=real=-1;
 else front++;}}
void DIS(int QUEUE [],int front, int real);
{
 If(front<=-1))cout<<"No Item ";
 else
 {
 for(int i=front;i<=real;i++)
 cout<< QUEUE[i];}}
```

Linked List, Stack Queues

Q1.
```
Class stack
{       int data[10];
        int top;
        Public:
        Stack( ) { top=-1;}
        Void push ( ); // to push an element into the stack
        Void pop ( ) ; // to pop an element into the stack
        Void display( );// to display all the elements from the stack
};
```
complete the class with all function definition.

Q2. Write a function in C++ to perform insert operation in dynamically allocated Queue containing names of students.

Q3. Write a function in C++ to perform push operation in a dynamically allocated stack containing admission number of students. Also declare the relevant class/ structure and pointers.

Q4. Write a function in C++ to perform a DELETE operation in a dynamically allocated queue considering the following description:
```
Struct Node
{       float U,V;
        Node *Link;
};
class QUEUE
{       Node *Raer, *Front;
        Public:
        QUEUE( ) { Rear =NULL; Front= NULL;}
        Void INSERT ( );
        Void DELETE ( );
        ~QUEUE ( );             }
```
Q5. Write a function in C++ to perform a PUSH operation in a dynamically allocated stack considering the following :
```
Struct Node
{       int X,Y;
        Node *Link;
};
class STACK
{       Node * Top;
        Public:
        STACK( ) { TOP=NULL;}
        Void PUSH( );
        Void Pop( );
        ~STACK( );
};
```
Q6. Define function stackpush( ) to insert nodes and stackpop( ) to delete nodes, for a linklist implemented stack having the following structure for each node:
```
Struct Node
{       char name[20];
        int age;
        Node *Link;
};
class STACK
{       Node * Top;
        Public:
        STACK( ) { TOP=NULL;}
```

Convert the following infix expressions to postfix expressions

1. (((x*y –a*b)-d/f)+a)*b
2. (a+(((b-(c*d-e)+f))/g))*(h-j)
3. A+B*(P+Q)^C/D
4. A+B-C*D+F-G
5. A^B-A/(B*(A-B-A*D)/B)
6. A+(((B-C)*(D-E)+F/G)^(H-J)
7. (A+B) *(C^(D-E)+F-G
8. A*(B+(C+D)*(E+F)/G)*H
9. A*(B+C)/D-E-(F+G/H)
10. A + B * C ^ D – (E / F – G)
11. A – B + C * D ^ E * G / H
12. ((A+B)-((C_D)*E/F)*G
13. (TRUE && FALSE) || ! (FALSE || TRUE)
14. (A + B * C) / D + E / (F * G + H / I)
15. NOT A OR NOT B AND NOT C

Evaluate the following postfix expression E given below, show the contents of the <u>stack during the evaluation</u>

16. E= 10,*,15,-,25,5,/,2+
17. E= 7,6,2,^,*,18,+
18. E= 5,9,+2,*,4,1,1,3,_,*,-
19. TRUE, FALSE, TRUE ,FALSE, NOT, OR, TRUE, OR, AND
20. E= 30,5,2,^,12,6,/,+,-
21. 15, 3, 2, +, /, 7, + 2, *
22. 25, 8, 3, - , / 6, *, 10 +
23. AB – CD + E * + WHERE A = 5, B = 3, C = 5, D = 4 AND E = 2
24. 7,6, +, 8, * , 2, -, 3, * , 2, 4, * , -
25. 8, 7, +, 9, 8, +, -, 2, / , 3,4, * 2, / +
26. E= 5,20,15,-,*,25,2,*,+

Evaluate the following <u>postfix</u> expression using a stack and show the Contents of stack after execution of each operation:

(i) 50,40,+, 18,14,-,4,*,+
(ii) 100,40,8,+,20,10,-,+,*
(iii) 5,6,9,+,80,5,*,-,/
(iv) 120,45,20,+,25,15,-,+,*
(v) 20,45,+,20,10,-,15,+,*
(vi) TRUE,FALSE, TRUE FALSE, NOT, OR, TRUE , OR,OR,AND

Give postfix form of the following expression:

(i) A*(B+(C+D)*(E+F)/G)*H
(ii) A+[(B+C)*(D+E)*F]/G
(iii) A*(B+D)/E-F-(G+H/K)
(iv) ((A-B)*(D/E))/(F*G*H)

Mark Questions : Linked List, Stack, Queue Convert the following infix expressions to postfix expressions

1. A + (B * C) ^ D – (E / F – G)
2. A * B / C * D ^ E * G / H
3. ((A*B)-((C_D)*E/F)*G
4. A+B/(P+Q)^C/D-E/F
5. A+B/C*D+F*G
6. A+B-A/(B*(A-B-A*D)^B)
7. (B+(C+D)*(E+F)/G)/H
8. A*(B/C)/D-E-(F+G/H)
9. (TRUE || FALSE) && ! (FALSE || TRUE)
10. (A / B + C) / D + E / (F + G * H / I)
11. A OR NOT B AND C
12. (((ax/by –a/b)-dx/fx)+a)+b
13. (((b-(c*d-e)+f))/g)+(h*j+x)
14. A+(((B*C)*(D+E)+F*G)^(H-J)
15. (A-B) *(C/(D-E)+F-G

Evaluate the following postfix expression E given below, show the contents of the stack during the evaluation

1. E= 5,9,+2,/,4,1,1,3,_,*,+
2. E= 80,35,20,-,25,5,+,-,*
3. E= 30,5,2,^,12,6,/,+,-
4. E=15, 3, 2, +, /, 7, + 2, *
5. E=25, 8, 3, - , / 6, *, 10 +
6. E=8, 7, -, 9, 8, *, -, 2, / , 3,4, * 2, / -
7. E= 5,20,15,-,*,25,2,*,-
8. IF A=2,C=3,D=2,E=5,F=4,G=6 then EFG^D+AC/- +
9. E= 10,+,15,*,25,5,/,2+
10. E= 7,6,2,/,+,18,-
11. E=AB – CD + E * + WHERE A = 5, B =
A*(B + (C + D) * (E * F)/ G) * H
3, 9, 4, +, *, 10, 2, /, -
50,40,+,18,14,-, 4,*,+
45, 7, +, 8, 10, -, *
True, False, AND, True, True, NOT, OR, AND

2 Mark Questions : Linked List, Stack, Queue

Convert the following infix expressions to postfix expressions

1. A + (B * C) ^ D – (E / F – G)
2. A * B / C * D ^ E * G / H
3. ((A*B)-((C_D)*E/F)*G
4. A+B/(P+Q)^C/D-E/F
5. A+B/C*D+F*G
6. A+B-A/(B*(A-B-A*D)^B)

7. (B+(C+D)*(E+F)/G)/H
8. A*(B/C)/D-E-(F+G/H)
9. (TRUE || FALSE) && ! (FALSE || TRUE)
10. (A / B + C) / D + E / (F + G * H / I)
11. A OR NOT B AND C
12. (((ax/by –a/b)-dx/fx)+a)+b
13. (((b-(c*d-e)+f))/g)+(h*j+x)
14. A+(((B*C)*(D+E)+F*G)^(H-J)
15. (A-B) *(C/(D-E)+F-G

2. Evaluate the following postfix expression E given below, show the contents of the stack during the evaluation

1. E= 5,9,+2,/,4,1,1,3,_,*,+
2. E= 80,35,20,-,25,5,+,-,*
3. E= 30,5,2,^,12,6,/,+,-
4. E=15, 3, 2, +, /, 7, + 2, *
5. E=25, 8, 3, - , / 6, *, 10 +
6. E=8, 7, -, 9, 8, *, -, 2, / , 3,4, * 2, / -
7. E= 5,20,15,-,*,25,2,*,-
8. IF A=2,C=3,D=2,E=5,F=4,G=6 then EFG^D+AC/- +
9. E= 10,+,15,*,25,5,/,2+
10. E= 7,6,2,/,+,18,-
11. E=AB – CD + E * + WHERE A = 5, B =

Problem 3:	Give the Postfix form of the following expression showing stack status: A*(B + (C + D) * (E * F)/ G) * H
Problem 4:	Evaluate the following postfix notation of expression (Show stack status after execution of each operation): 3, 9, 4, +, *, 10, 2, /, -
Problem 5:	Evaluate the following postfix notation of expression (Show status of stack after execution of each operation) : 50,40,+,18,14,-, 4,*,+
Problem 6:	Evaluate the following postfix notation of expression (Show status of stack after execution of each operation) : 45, 7, +, 8, 10, -, *
Problem 7:	Evaluate the following postfix notation of expression (Show status of stack after execution of each operation) : True,False,AND,True,True,NOT,OR,AND