1.Program to Implement Stack ADT using array.

Program:

```
#include<stdio.h>
#define MAX 50
char stack[MAX];
int TOP=-1;

int pop(int[]);
int display(int[]);
int push(int[],int);

int main(){
        int choice,num;
        while(1){
        printf("Enter your choice:\n1)Display\n2)Push\n3)Pop\n4)Exit\n");
        scanf("%d",&choice);
                switch(choice){
                        case 1:
                                display(stack);
                                break;
                        case 2:
                                printf("Enter number to enter\n");
                                scanf("%d",&num);
                                push(stack,num);
                                break;
                        case 3:
                                pop(stack);
                                break;
                        case 4:
                                exit(0);
                                break;
                        default:
                                printf("Invalid choice");
                }
        }
        return 0;
}

int display(int stack[]){
        int i;
        if(TOP==-1)
                printf("Stack is empty\n");
        else{
        printf("%d <--TOP\n",stack[TOP]);
        for(i=(TOP-1);i>=0;i--){
                printf("%d",stack[i]);
                printf("\n");
                }
```

```
	}
}

int push(int stack[],int num){
	if((TOP-1)==MAX)
		printf("Stack is full\n");
	else{
		TOP++;
		stack[TOP]=num;
	}
}

int pop(int stack[]){
	int del;
	if(TOP==-1)
		printf("Stack is empty\n");
	else{
		del=stack[TOP];
		TOP--;
		printf("Deleted number is %d\n",del);
	}
}
```

Output:

```
Enter your choice:
1)Display
2)Push
3)Pop
4)Exit
2
Enter number to enter
45
Enter your choice:
1)Display
2)Push
3)Pop
4)Exit
2
Enter number to enter
```

```
56
Enter your choice:
1)Display
2)Push
3)Pop
4)Exit
2
Enter number to enter
77
Enter your choice:
1)Display
2)Push
3)Pop
4)Exit
1
77 <--TOP
56
45
Enter your choice:
1)Display
2)Push
3)Pop
4)Exit

3
Deleted number is 77
Enter your choice:
1)Display
2)Push
3)Pop
```

```
4)Exit

1

56 <--TOP

45

Enter your choice:

1)Display

2)Push

3)Pop

4)Exit

4


--------------------------------

Process exited after 37.1 seconds with return value 0

Press any key to continue . . .
```

2.Program to convert an Infix expression to postfix expression using stack ADT.

Program:

```
#include<stdio.h>


char stack[20];
int top=-1;
void push (char x)
{
        stack[++top]=x;
}
char pop()
{
        if(top==-1)
          return -1;
        else
          return stack[top--];
}
int priority (char x)
{
        if(x=='(')
          return 0;
   if(x=='+' || x=='-')
     return 1;
   if(x=='*' || x=='/'|| x=='%')
     return 2;
   if(x=='^')
     return 3;
}
```

```
int main()
{
        char exp[20];
        char *e,x;
        printf("Enter the expression :");
        scanf("%s",exp);
        e = exp;
        while(*e!='\0')
         {
                if(isalnum(*e))
                  printf("%c",*e);
                else if(*e=='(')
                   push(*e);
                else if(*e==')')
                {
                        while((x = pop()) != '(')
                          printf("%c",x);
                }

           else
           {
                while(priority(stack[top]) >= priority(*e))
                    printf("%c",pop());
                push(*e);
                }
                e++;
   }
   while(top !=-1)
     {
        printf("%c",pop());
        }
}
```

Output:

```
Enter the expression :(A-B)*(C+D)

AB-CD+*

--------------------------------

Process exited after 77.69 seconds with return value 4294967295

Press any key to continue . . .
```

3.Program to Evaluate Postfix Expression using Stack ADT.

Program:

```
#include<stdio.h>

char stack[20];
int top=-1;

void push (int x)
{
        stack[++top]=x;
}
int pop()
{
        if(top==-1)
          return -1;
        else
          return stack[top--];
}
int main()
{
        char exp[20];
        char *e;
        int n1,n2,n3,num;
        printf("Enter the expression :");
        scanf("%s",exp);
        e = exp;
        while(*e!='\0')
        {
             if(isdigit(*e))
 {
          num = *e - 48;
   push(num);
 }
else
{
   n1 = pop();
   n2 = pop();
         switch(*e)
           {
             case '+':
                        {
            n3 = n1+n2;
               break;
           }
             case '-':
                        {
                             n3 = n2-n1;
                              break;
                                  }
                                     case '*':
                                                      {
                                                       n3 = n1*n2;
                                                       break;
                                                    }
                                                     case '/':
                                                               {
                                                          n3 = n2/n1;
                                                          break;
```

```
                                       }
                     }
                   push(n3);
            }
        e++;
      }
      printf("\nThe result of expression %s = %d\n\n",exp,pop());
  return 0;
}
```

Output:

```
Enter the expression :456*+

The result of expression 456*+ = 34



--------------------------------
Process exited after 21.28 seconds with return value 0
Press any key to continue . . .
```

4. Implement (Menu Driven Program) Linear Queue ADT using array.

Program:

```
#include<stdio.h>
#include<stdlib.h>
#define MAX 50

void insert();
void deleteq();
void display();
int queue_array[MAX];
int rear = -1;
int front = -1;
int main()
{
        int choice;
        while(1)
        {
                printf("1.Insert element to queue\n");
                printf("2.Delete element from queue\n");
     printf("3.Display all elements of queue\n");
     printf("4.Quit\n");
     printf("Enter your choice : ");
     scanf("%d", &choice);
   switch(choice)
    {
        case 1:
                insert();
```

```
                       break;
                       case 2:
                           deleteq();
                            break;
                            case 3:
                                display();
                                break;
                                case 4:
                                   exit(1);
                                   default:
                                               printf("Wrong choice\n");
              }
            }
}
void insert()
{
 int item;
 if(rear == MAX-1)
    printf("queue overflow\n");
 else
 {
            if(front == -1)
            front = 0;
            printf("insert the element in queue :");
            scanf("%d",&item);
            rear = rear+1;
            queue_array[rear] = item;
 }

}
```

```
void deleteq()
{
            if(front == -1 || front > rear)
            {
              printf("queue overflow\n");
              return;
    }
            else
            {
              printf("Element deleted from the queue is :%d\n",queue_array[front]);
              front = front + 1;
    }
}
void display()
{
            int i;
    if(front == - 1)
      printf("Queue is empty\n");
    else
    {
      printf("queue is :");
      for(i=front;i<=rear;i++)
              {
        printf("%d ",queue_array[i]);


       }
            }
            printf("\n");
}
```

Output:

```
1.Insert element to queue
2.Delete element from queue
3.Display all elements of queue
4.Quit
Enter your choice : 1
insert the element in queue :19
1.Insert element to queue
2.Delete element from queue
3.Display all elements of queue
4.Quit
Enter your choice : 1
insert the element in queue :27
1.Insert element to queue
2.Delete element from queue
3.Display all elements of queue
4.Quit
Enter your choice : 1
insert the element in queue :66
1.Insert element to queue
2.Delete element from queue
3.Display all elements of queue
4.Quit
Enter your choice : 1
insert the element in queue :86
1.Insert element to queue
2.Delete element from queue
3.Display all elements of queue
4.Quit
Enter your choice : 3
queue is :19 27 66 86
```

```
1.Insert element to queue
2.Delete element from queue
3.Display all elements of queue
4.Quit
Enter your choice : 2
Element deleted from the queue is :19
1.Insert element to queue
2.Delete element from queue
3.Display all elements of queue
4.Quit
Enter your choice : 3
queue is :27 66 86
1.Insert element to queue
2.Delete element from queue
3.Display all elements of queue
4.Quit
Enter your choice : 4

--------------------------------
Process exited after 56.77 seconds with return value 1
Press any key to continue . . .
```

5.Program to Implement Circular Queue ADT using array.

Program:

```
#include<stdio.h>
#define MAX 5

int cqueue_arr[MAX];
int front = -1;
int rear = -1;

int main()
{
  int choice,item;
  do
        {

        printf("1.Insert\n");
        printf("2.Delete\n");
        printf("3.Display\n");
        printf("4.Quit\n");
        printf("Enter your choice :");
        scanf("%d",&choice);
        switch(choice)
          {
              case 1:
                      printf("Input the element for element for insertion in queue:");
                      scanf("%d",&item);
                      insert(item);
                      break;
                      case 2:
                              deletion();
```

```
                break;
                case 3:
                    display();
                    break;
                    case 4:
                        exit(0);
                        break;
                        default:
                            printf("Wrong choice\n");
     }
 }
 while(choice!=4);
 return 0;
}


void insert(int item)
{
    if((front == 0 && rear ==MAX-1)||(front == rear+1))
     {
        printf("Queue overflow\n");
        return;
     }
    if(front == -1)
     {
        front = 0;
        rear = 0;
     }
    else
     {
```

```
                        if(rear == MAX-1)
                          rear = 0;
                        else
                          rear = rear + 1;
             }
            cqueue_arr[rear] = item;
}

void deletion()
{
            if(front == -1)
             {
                        printf("Queue underflow\n");
                        return;
             }
            printf("Element deleted from queue is:%d\n",cqueue_arr[front]);
            if(front==rear)
              {
                        front = -1;
                        rear = -1;
              }
            else
            {


                        if(front == MAX-1)
                          front = 0;


                else
                        front = front + 1;
```

```
}
}

void display()
{
          int front_pos = front, rear_pos = rear;
          if(front == -1)
           {
                    printf("Queue is empty\n");
                    return;
           }
           printf("Queue elements:\n");
           if(front_pos <= rear_pos)
           {

              while(front_pos <= rear_pos)
                {
                  printf("%d ",cqueue_arr[front_pos]);
                  front_pos++;
         }
     }
    else
      {
       while(front_pos <= MAX-1)
        {

            printf("%d ",cqueue_arr[front_pos]);
            front_pos++;
                       }
                       front_pos = 0;
```

```
                   while(front_pos <= rear_pos)
                   {
                          printf("%d ",cqueue_arr[front_pos]);
                          front_pos++;
                pit}
        }
                 printf("\n");
   }
```

Output:

```
1.Insert
2.Delete
3.Display
4.Quit
Enter your choice :1
Input the element for element for insertion in queue:14
1.Insert
2.Delete
3.Display
4.Quit
Enter your choice :1
Input the element for element for insertion in queue:34
1.Insert
2.Delete
3.Display
4.Quit
Enter your choice :1
Input the element for element for insertion in queue:45
1.Insert
2.Delete
3.Display
```

```
4.Quit
Enter your choice :1
Input the element for element for insertion in queue:78
1.Insert
2.Delete
3.Display
4.Quit
Enter your choice :3
Queue elements:
14 34 45 78
1.Insert
2.Delete
3.Display
4.Quit
Enter your choice :2
Element deleted from queue is:14
1.Insert
2.Delete
3.Display
4.Quit
Enter your choice :4

--------------------------------
Process exited after 40.33 seconds with return value 0
Press any key to continue . . .
```

6A. Implement Singly Linked List ADT.

Program:

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#include<malloc.h>
struct node
{
int data ;
struct node *next ;
}*head ;
void createList (int n) ;
void insert_after_pos (int data) ;
void insert_before_pos (int pos) ;
void displayList ();
void insertBeginning (int data);

void insertEnd (int data) ;
void delete_beginning () ;
void delete_end () ;
void delete_from_pos ();
void reverse ();
int count=0 ;

int main ()
{
int n ,pos,data,i ;
printf ("\nEnter the number of the nodes:\t");
scanf ("%d",&n);
for (i=0;i<=n;i++)
```

```
{
count++ ;
}
createList(n);
printf ("\nList is :");
displayList();
insertBeginning (data) ;
printf ("\nList after inserting the begginning :");
displayList();
insertEnd (data);
printf ("\nList after entering at end:");
displayList();
insert_after_pos (data);
printf ("\nThe new list after entering after given position");
displayList();
insert_before_pos (pos) ;
printf ("\nThe new list after entering before given position");
displayList();
delete_beginning ();
printf ("\nList after deleting at beginning :");
displayList();
delete_end ();
printf ("\nList after deleting at end :");
displayList();
delete_from_pos ();

printf ("\nList afetr deleting after the given position :");
displayList();
reverse();
printf ("\nReversed list is :");
```

```
displayList();

}
void createList (int n)
{
struct node *newnode , *temp ;
int i ;
head = 0 ;
for (i = 0; i<n ; i++)
{
newnode = (struct node*)malloc(sizeof(struct node));
printf ("\nEnter the new node");
scanf ("%d",&newnode->data);
newnode->next = 0 ;
if (head == 0)
{
head = temp = newnode ;
temp = newnode ;
}
else
{
temp->next = newnode ;
temp = newnode ;
}
}
}
void displayList()
{
struct node *temp ;
int count = 0 ;
```

```
if (head == NULL)
{
printf("\nList is empty");
}
else
{
temp = head ;

while (temp != NULL)
{
printf("\n%d",temp->data);
temp = temp->next ;
count++ ;
}
printf("\n");
printf("\nCOUNT IS %d",count);

}
}

void insert_after_pos (int pos)
{
int i=1;
struct node *temp,*newnode;
newnode = (struct node*)malloc(sizeof(struct node));
printf ("\nEnter the position after which the new node to be inserted");
scanf("\n%d",&pos) ;
if (pos>count)
{
printf("\nInvalid position");
}
else
{
temp = head ;

while (i < pos)
{
temp = temp->next ;
i++ ;
}
printf ("\nEnter data to insert after position %d of the list :",pos);
scanf("%d",&newnode->data);
newnode->next = temp->next ;
temp->next = newnode ;

}
}
void insert_before_pos (int pos)
{

int i=1;
struct node *temp,*newnode;
newnode = (struct node*)malloc(sizeof(struct node));
printf ("\nEnter the position before which the new node to be inserted");
scanf("\n%d",&pos) ;
if (pos>count)
{
printf("\nInvalid position");
}
else
```

```
{
temp = head ;

while (i < pos-1)
{
temp = temp->next ;
i++ ;
}
printf ("\nEnter data to insert before position %d of the list :",pos);
scanf("%d",&newnode->data);
newnode->next = temp->next ;
temp->next = newnode ;

}
}
void insertBeginning (int data)
{
printf ("\nEnter the data at the beginning of the list :");
scanf ("\n%d",&data);
struct node *temp ;
temp = (struct node*)malloc(sizeof(struct node));
temp->data = data ;
temp->next = head ;
head = temp ;
}
void insertEnd (int data)
{
printf ("\nEnter the data at the end of the list :");
scanf ("\n%d",&data);
struct node *temp,*newnode ;
```

```
newnode = (struct node*)malloc(sizeof(struct node));
if (newnode== NULL)
{
printf("\nUnable to allocate the memory");


}
else
{
newnode->data = data ;
newnode->next = NULL ;
temp = head ;
//Now travesring through the list
while (temp->next != NULL )
{
temp = temp->next ;
}
temp->next = newnode ;
}
}
void delete_beginning()
{
struct node *temp ;
temp = head ; //bringing the temp at starting
head = head->next ; // linking the head to second node and discard the first node
free(temp); //clearing the memory at first node whicj is temp
}
void delete_end()
{
struct node *temp,*prevnode ;
temp = head ; //bringing the temp at starting
```

```
while (temp->next != 0 )
{
prevnode = temp ; //prevnode is pointing to the node befpre the temp node
temp = temp->next ;
}
if (temp == head )
{
head = 0 ;
}
else
{
prevnode->next = 0 ; //set the next of prevnode to 0 so it will be the last element of the list
}
free (temp);
}
void delete_from_pos ()
{
struct node *temp, *nextnode ;

int pos, i=1;
temp = head ; //bringing temp at starting
printf("\nEnter position");
scanf("\n%d",&pos);
if (pos > count)
{
printf ("\nInvalid position");
}
else
{
```

```
while (i<pos-1)
{
temp = temp->next ;
i++ ;
}
nextnode = temp->next ;
temp->next = nextnode->next ;
free (nextnode) ;
}
}
void reverse ()
{
struct node *prevnode , *currentnode, *nextnode ;
prevnode = 0 ;
currentnode = nextnode = head ;
while (nextnode!=0)
{
nextnode = nextnode->next ;
currentnode->next = prevnode ;
prevnode = currentnode ;
currentnode = nextnode ;
}
head = prevnode ;
}
```

Output:

```
Enter the number of the nodes:  5

Enter the new node99
Enter the new node88

Enter the new node77

Enter the new node66

Enter the new node555

List is :
99
88
77
66
555

COUNT IS 5
Enter the data at the beginning of the list :5

List after inserting the begginning :
5
99
88
77
66
555

COUNT IS 6
Enter the data at the end of the list :9
```

```
List after entering at end:
5
99
88
77
66
555
9

COUNT IS 7
Enter the position after which the new node to be inserted3

Enter data to insert after position 3 of the list :58

The new list after entering after given position
5
99
88
58
77
66
555
9

COUNT IS 8
Enter the position before which the new node to be inserted1

Enter data to insert before position 1 of the list :653

The new list after entering before given position
```

```
5
653
99
88
58
77
66
555
9

COUNT IS 9
List after deleting at beginning :
653
99
88
58
77
66
555
9

COUNT IS 8
List after deleting at end :
653
99
88
58
77
66
555
```

```
COUNT IS 7
Enter position5

List afetr deleting after the given position :
653
99
88
58
66
555

COUNT IS 6
Reversed list is :
555
66
58
88
99
653

COUNT IS 6
--------------------------------
Process exited after 60.99 seconds with return value 0
Press any key to continue . . .
```

6B. Implement Circular Linked List ADT.

Program:

```
#include<stdio.h>
#include<stdlib.h>
struct node

{
int data;
struct node *next;
};
struct node *head,*tail;
void createList(int n);
void insertBig();
void insertEnd();
void insert_after_given_position();
void del_from_beg();
void del_from_end();
void del_from_pos();
void displayList();
int main()
{
int n;
printf("\nEnter total no of nodes");
scanf("%d",&n);
createList(n);
printf("\nThe nodes are\n");
displayList();
insertBig();
printf("\nThe nodes are\n");
displayList();
insertEnd();
printf("\nThe nodes are\n");
```

```
displayList();
insert_after_given_position();
printf("\nThe nodes are\n");
displayList();
del_from_beg();
printf("\nThe nodes after deleting first node\n");
displayList();
del_from_end();
printf("\nThe nodes after deleting last node\n");
displayList();
del_from_pos();
displayList();
}
void createList(int n)
{

struct node *newnode;
int i;
head=0;
for(i=0;i<n;i++)

{
newnode=(struct node*)malloc(sizeof(struct node));
printf("\n enter new node");
scanf("%d",&newnode->data);
newnode->next=0;
if(head==0)
{
head=tail=newnode;
tail->next=head;
}
else
{
tail->next=newnode;
tail=newnode;

tail->next=head;
}

}
}
void insertBig()
{
//Create new node
struct node *newNode, *temp;
newNode = (struct node*)malloc(sizeof(struct node));
printf("\n Enter data to insert at begin\n");
scanf("%d",&newNode->data);
//Checks if the list is empty.
if(head == NULL){
//If list is empty, both head and tail would point to new node.
head = newNode;
tail = newNode;
newNode->next = head;
}
else {
temp = head;
newNode->next = temp;
head = newNode;
```

```
//Since, it is circular linked list tail will point to head.
tail->next = head;
}
}

void insertEnd()
{
//Create new node
struct node *newNode, *temp;
newNode = (struct node*)malloc(sizeof(struct node));
printf("\n Enter data to insert at end\n");
scanf("%d",&newNode->data);
tail->next=newNode;
tail=newNode;
tail->next=head;
}
void insert_after_given_position()
{
int i=1,pos;
struct node *newNode, *temp;
newNode = (struct node*)malloc(sizeof(struct node)); // Allocate memory for the node
printf("\nEnter position\n");
scanf("%d", &pos);
/*if(position>count)
{
printf("\n Invalid Position");
}
else
{*/
temp = head;
```

```
// Traverse to the given position in the list
while(i<pos)
{
temp=temp->next;
i++;
}
printf("\nEnter data to insert after position %d of the list: ",pos);
scanf("%d", &newNode->data);
newNode->next = temp->next; //Link the inserted node with the next node
temp->next = newNode; // Link the previous node and the inserted node
tail->next=head;
}
void del_from_beg()
{
struct node* temp;
temp = head; // bringing temp at starting

head=head->next; //link between head and second node
free(temp); //relese memory
tail->next=head;
}
void del_from_end()
{
struct node* temp1,*prev;
temp1=head;
while(temp1 -> next != head)
{
prev = temp1;
temp1 = temp1 -> next;
}
```

```
prev -> next = head;

free(temp1);

}

void del_from_pos()

{

struct node* temp,*nextnode;

int pos,i=1;

temp = head; // bringing temp at starting

printf("\n Enter position");

scanf("%d",&pos);

/* if(pos>count)

{

printf("\nInvalid Position");

}

*/

while(i<pos-1)

{

temp=temp->next;

i++;

}

nextnode=temp->next;

temp->next=nextnode->next;

free(nextnode);

printf("\n\nThe List after deleting position %d node is \n",pos);

}


void displayList()

{


struct node *temp;
```

```
if(head == NULL)
{
printf("List is empty.");
}
else
{
temp = head;
while(temp -> next != head)
{
printf("%d\t", temp->data);
temp = temp->next;
}
printf("%d\t", temp->data);
}
printf("Circular Print %d",tail->next->data);
}
```

Output:

```
Enter the number of the nodes:  4

Enter the new node66

Enter the new node55

Enter the new node44

Enter the new node33

List is :
```

```
66
55
44
33

COUNT IS 4
Enter the data at the beginning of the list :11

List after inserting the begginning :
11
66
55
44
33
COUNT IS 5
Enter the data at the end of the list :99

List after entering at end:
11
66
55
44
33
99
COUNT IS 6
Enter the position after which the new node to be inserted2

Enter data to insert after position 2 of the list :77

The new list after entering after given position
```

```
11
66
77
55
44
33
99
COUNT IS 7
Enter the position before which the new node to be inserted4
Enter data to insert before position 4 of the list :88
The new list after entering before given position
11
66
77
88
55
44
33
99
COUNT IS 8
List after deleting at beginning :
66
77
88
55
44
33
99
COUNT IS 7
List after deleting at end :
```

```
66

77

88

55

44

33

COUNT IS 6

Enter position6

Invalid position

List afetr deleting after the given position :

66

77

88

55

44

33

COUNT IS 6

Reversed list is :

33

44

55

88

77

66

COUNT IS 6

--------------------------------

Process exited after 31.62 seconds with return value 0

Press any key to continue . . .
```

7. Implement Stack / Linear Queue ADT using Linked List.

Program (Stack with Linked List):

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

struct Node
 {
  int data;
  struct Node *next;
 }*top = NULL;

 void push(int);
 void pop();
 void display();
 int main()
 {
  int choice, value;
  printf("\nIMPLEMENTING STACKS USING LINKED LISTS\n");
  while(1)
   {
    printf("1. Push\n2. Pop\n3. Display\n4. Exit\n");
    printf("\nEnter your choice : ");
    scanf("%d",&choice);
    switch(choice)
      {
        case 1: printf("\nEnter the value to insert: ");
        scanf("%d", &value);
        push(value);
        break;
         case 2: pop();
         break;
          case 3: display();
           break;
            case 4: exit(0);
             break;
              default: printf("\nInvalid Choice\n");
      }
   }
 }
 void push(int value)
 {
  struct Node *newNode;
  newNode = (struct Node*)malloc(sizeof(struct Node));
  newNode->data = value;
  if(top == NULL)
   newNode->next = NULL;
  else
   newNode->next = top;
   top = newNode;
   printf("Node is Inserted\n\n");
 }
 void pop()
 {
  if(top == NULL)
  printf("\nEMPTY STACK\n");
  else
  {
   struct Node *temp = top;
   printf("\nPopped Element : %d", temp->data);
   printf("\n");
   top = temp->next;
   free(temp);
  }
 }
 void display()
 {
  if(top == NULL)
  printf("\nEMPTY STACK\n");
  else
   {
    printf("The stack is \n");
    struct Node *temp = top;
    while(temp->next != NULL)
            {
      printf("%d--->",temp->data);
      temp = temp -> next;
    }
    printf("%d--->NULL\n\n",temp->data);
  }
 }
```

Output:

```
IMPLEMENTING STACKS USING LINKED LISTS
1. Push
2. Pop
3. Display
4. Exit

Enter your choice : 1
Enter the value to insert: 38
Node is Inserted

1. Push
2. Pop
3. Display
4. Exit

Enter your choice : 1

Enter the value to insert: 45
Node is Inserted

1. Push
2. Pop
3. Display
4. Exit

Enter your choice : 1

Enter the value to insert: 66
Node is Inserted

1. Push
2. Pop
3. Display
4. Exit

Enter your choice : 1
Enter the value to insert: 78
Node is Inserted

1. Push
2. Pop
3. Display
4. Exit

Enter your choice : 3
The stack is
78--->66--->45--->38--->NULL

1. Push
2. Pop
3. Display
4. Exit

Enter your choice : 2

Popped Element : 78
1. Push
2. Pop
3. Display
4. Exit

Enter your choice : 3
The stack is
66--->45--->38--->NULL

1. Push
```

```
2. Pop

3. Display

4. Exit


Enter your choice : 4


--------------------------------

Process exited after 45.85 seconds with return value 0

Press any key to continue . . .
```

Program (Queue with Linked List):

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

struct node
 {
  int data;
  struct node*next;
 };
 struct node *front=0,*rear=0;
 void enqueue(int n);
 void dequeue();
 void display();
 int main()
  {
   int i=1,select,item;
   while(i)
    {
      printf("\nMainMenu");
      printf("\n1:ENQUEUE");
      printf("\n2:DEQUEUE");
      printf("\n3:DISPLAY");
      printf("\n4:EXIT");
      printf("\nEnteryourchoice :");
      scanf("\n%d",&select);
      switch(select)
       {
```

```
       case 1:
          printf("\nEnter the data to insert in the Queue from rear:");
          scanf("\n%d",&item);
          enqueue(item);
          break;
          case 2:
             printf("\nDeletingfromthefront:");
             dequeue();
             break;
             case 3:
                 printf("\nThe list is:");
                 display();
                 break;
                 case 4:
                   exit(0);
                   break;
                   default:
                     printf("\nInvalid Choice");
                     break;
       }
     printf("\n Do u want to continue, please enter 1 or 0\n");
     scanf("%d", &i);
   }
 return 0;
}

void enqueue(int n)
{
 struct node *newnode;
 newnode=(struct node*)malloc(sizeof(struct node));
 newnode->data=n;
 newnode->next=0;
 if(front==0&&rear==0)
  {
   front=rear=newnode;
  }
 else
  {
   rear->next=newnode;
   rear=newnode;
  }
}

void dequeue()
{
 struct node *temp;
 if(front==0&&rear==0)
  {
   printf("\nUnderflow");
  }
 else
  {
   temp=front;
   printf("\nDeleted item is %d",front->data);
   front=front->next;
   free(temp);
  }
}

void display()
```

```
 {
  struct node *temp;
  temp=front;
  if(front==NULL)
   {
    printf("\nUnderflow");
   }
  else
   {
    while(temp!=NULL)
     {
      printf("\t%d",temp->data);
      temp=temp->next;
     }
   }
 }
```

Output:

```

MainMenu
1:ENQUEUE
2:DEQUEUE
3:DISPLAY
4:EXIT
Enteryourchoice :1

Enter the data to insert in the Queue from rear:22

 Do u want to continue, please enter 1 or 0
1
```

```
MainMenu
1:ENQUEUE
2:DEQUEUE
3:DISPLAY
4:EXIT
Enteryourchoice :1

Enter the data to insert in the Queue from rear:33

 Do u want to continue, please enter 1 or 0

1

MainMenu
1:ENQUEUE
2:DEQUEUE
3:DISPLAY
4:EXIT
Enteryourchoice :1

Enter the data to insert in the Queue from rear:44

 Do u want to continue, please enter 1 or 0
1

MainMenu
1:ENQUEUE
2:DEQUEUE
3:DISPLAY
4:EXIT
```

```
Enteryourchoice :1

Enter the data to insert in the Queue from rear:55

 Do u want to continue, please enter 1 or 0
1

MainMenu
1:ENQUEUE
2:DEQUEUE
3:DISPLAY
4:EXIT
Enteryourchoice :3

The list is:    22      33      44      55
 Do u want to continue, please enter 1 or 0
1

MainMenu
1:ENQUEUE
2:DEQUEUE
3:DISPLAY
4:EXIT
Enteryourchoice :2

Deletingfromthefront:
Deleted item is 22
 Do u want to continue, please enter 1 or 0
1
```

```
MainMenu
1:ENQUEUE
2:DEQUEUE
3:DISPLAY
4:EXIT
Enteryourchoice :3

The list is:    33      44      55
 Do u want to continue, please enter 1 or 0
1

MainMenu
1:ENQUEUE
2:DEQUEUE
3:DISPLAY
4:EXIT
Enteryourchoice :4

--------------------------------
Process exited after 79.67 seconds with return value 0
Press any key to continue . . .
```

8. Implement Binary Search Tree ADT using Linked List.

Program:

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

struct node
 {
  int data;
  struct node *right_child;
  struct node *left_child;
 };

 void search(int i, struct node *n)
 {
 if (n == NULL)
  printf("\nValue does not exist in tree!");
 else if(n->data == i)
  printf("\nValue found!");
 else if(i > n->data)
  search(i, n->right_child);
 else
  search(i, n->left_child);
 }

 struct node* smallest(struct node *root)
 {
  while(root != NULL && root->left_child != NULL)
  {
   root = root->left_child;
  }
  return root;
 }

 struct node* largest(struct node *root)
 {
  while (root != NULL && root->right_child != NULL)
   {
    root = root->right_child;
   }
  return root;
 }


 struct node* new_node(int x)
 {
  struct node *p;
  p = malloc(sizeof(struct node));
  p->data = x;
  p->left_child = NULL;
  p->right_child = NULL;
  return p;
 }

 struct node* insert(struct node *root, int x)
 {
  if(root==NULL)
  return new_node(x);
  else if(x>root->data)
   root->right_child = insert(root->right_child, x);
  else
   root->left_child = insert(root->left_child,x);
  return root;
 }

 struct node* delete(struct node *root, int x)
 {
  if(root==NULL)
  return NULL;
  if (x>root->data)
   root->right_child = delete(root->right_child, x);
  else if(x<root->data)
   root->left_child = delete(root->left_child, x);
  else
  {
   if(root->left_child==NULL && root->right_child==NULL)
    {
     free(root);
      return NULL;
    }
  else if(root->left_child==NULL || root->right_child==NULL)
   {
    struct node *temp;
    if(root->left_child==NULL)
    temp = root->right_child;
    else
    temp = root->left_child;
    free(root);
    return temp;
  }
```

```
  else
  {
   struct node *temp = smallest(root->right_child);
   root->data = temp->data;
   root->right_child = delete(root->right_child, temp->data);
  }
 }
  return root;
 }

 void inorder(struct node *root)
 {
  if(root!=NULL)
  {
   inorder(root->left_child);
   printf(" %d ", root->data);
   inorder(root->right_child);
  }
 }

int main()
{
 struct node *root,*min,*max;
 int x;
 root = new_node(20);
 insert(root,5);
 insert(root,1);
 insert(root,15);
 insert(root,9);
 insert(root,7);
insert(root,12);
insert(root,30);
insert(root,25);
insert(root,40);
insert(root, 45);
insert(root, 42);

inorder(root);
printf("\n");

root = delete(root, 1);
root = delete(root, 40);


root = delete(root, 45);

root = delete(root, 9);
inorder(root);

printf("\n");
min=smallest(root);

printf("\nSmallest value is %d\n", min->data);
max=largest(root);
printf("\nlargest value is %d\n", max->data);
printf("\n enter element to search\n");
scanf("%d",&x);
search(x,root);
return 0;
}
```

Output:

```
1  5  7  9  12  15  20  25  30  40  42  45
 5  7  12  15  20  25  30  42

Smallest value is 5

largest value is 42

 enter element to search
25

Value found!
--------------------------------
Process exited after 15.03 seconds with return value 0
Press any key to continue . . .

1  5  7  9  12  15  20  25  30  40  42  45
 5  7  12  15  20  25  30  42

Smallest value is 5

largest value is 42

 enter element to search
4

Value does not exist in tree!
--------------------------------
```

```
Process exited after 5.276 seconds with return value 0

Press any key to continue . . .
```

9. Implement Graph Traversal techniques a) Depth First Search b) Breadth First Search

Program:

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

 int q[20],top=-1,front=-1,rear=-1,a[20][20],vis[20],stack[20];
 int delete();
 void add(int item);
 void bfs(int s,int n);
 void dfs(int s,int n);
 void push(int item);
 int pop();

 void main()
 {
  int n,i,s,ch,j;
  char c,dummy;
  printf("ENTER THE NUMBER VERTICES:" );
  scanf("%d",&n);
  for(i=1;i<=n;i++)
  {
   for(j=1;j<=n;j++)
   {
    printf("ENTER 1 IF %d HAS A NODE WITH %d ELSE 0 ",i,j);
    scanf("%d",&a[i][j]);
   }
  }
 printf("THE ADJACENCY MATRIX IS\n");
 for(i=1;i<=n;i++)
 {
```

```
 for(j=1;j<=n;j++)
 {
  printf("%d",a[i][j]);
 }
 printf("\n");
}
do
{
 for(i=1;i<=n;i++)
 vis[i]=0;
 printf("\nMENU");
 printf("\n1.B.F.S");
 printf("\n2.D.F.S");
 printf("\nENTER YOUR CHOICE");
 scanf("%d",&ch);
 printf("ENTER THE SOURCE VERTEX :");
 scanf("%d",&s);
 switch(ch)
  {
   case 1:
            bfs(s,n);
     break;
      case 2:
       dfs(s,n);
       break;
  }
  printf(" DO U WANT TO CONTINUE(Y/N) ? ");
  scanf("%c",&dummy);
  scanf("%c",&c);
}
```

```
  while((c=='y')||(c=='Y'));
}

void bfs(int s,int n)
 {
  int p,i;
  add(s);
  vis[s]=1;
  p=delete();
  if(p!=0)
   printf("%d",p);
  while(p!=0)
  {
   for(i=1;i<=n;i++)
   if((a[p][i]!=0)&&(vis[i]==0))
   {
    add(i);
    vis[i]=1;
   }
   p=delete();
   if(p!=0)
    printf("%d",p);
  }
  for(i=1;i<=n;i++)
   if(vis[i]==0)
   bfs(i,n);
 }

 void add(int item)
 {
```

```
 if(rear==19)
  printf("QUEUE FULL");
 else
 {
  if(rear==-1)
  {
   q[++rear]=item;
   front++;
  }
 else
 q[++rear]=item;
}
}

int delete()
{
 int k;
 if((front>rear)||(front==-1))
  return(0);
 else
 {
  k=q[front++];
   return(k);
 }
}

void dfs(int s,int n)
{
 int i,k;
 push(s);
 vis[s]=1;
 k=pop();
 if(k!=0)
  printf(" %d ",k);
 while(k!=0)
 {
  for(i=1;i<=n;i++)
    if((a[k][i]!=0)&&(vis[i]==0))
     {
      push(i);
      vis[i]=1;
     }
  k=pop();
  if(k!=0)
    printf("%d",k);
 }
 for(i=1;i<=n;i++)
   if(vis[i]==0)
     dfs(i,n);
 }

void push(int item)
{
 if(top==19)
  printf("Stack overflow ");
 else
 stack[++top]=item;
}
int pop()
{
  int k;
  if(top==-1)
  return(0);
 else
 {
  k=stack[top--];
  return(k);
 }
}
```

Output:

```
ENTER THE NUMBER VERTICES:4
ENTER 1 IF 1 HAS A NODE WITH 1 ELSE 0 1
ENTER 1 IF 1 HAS A NODE WITH 2 ELSE 0 0
ENTER 1 IF 1 HAS A NODE WITH 3 ELSE 0 1
ENTER 1 IF 1 HAS A NODE WITH 4 ELSE 0 0
ENTER 1 IF 2 HAS A NODE WITH 1 ELSE 0 0
ENTER 1 IF 2 HAS A NODE WITH 2 ELSE 0 1
ENTER 1 IF 2 HAS A NODE WITH 3 ELSE 0 1
ENTER 1 IF 2 HAS A NODE WITH 4 ELSE 0 0
ENTER 1 IF 3 HAS A NODE WITH 1 ELSE 0 0
ENTER 1 IF 3 HAS A NODE WITH 2 ELSE 0 0
ENTER 1 IF 3 HAS A NODE WITH 3 ELSE 0 1
ENTER 1 IF 3 HAS A NODE WITH 4 ELSE 0 1
ENTER 1 IF 4 HAS A NODE WITH 1 ELSE 0 1
ENTER 1 IF 4 HAS A NODE WITH 2 ELSE 0 1
ENTER 1 IF 4 HAS A NODE WITH 3 ELSE 0 0
ENTER 1 IF 4 HAS A NODE WITH 4 ELSE 0 1
THE ADJACENCY MATRIX IS
1010
0110
```

```
0011
1101

MENU
1.B.F.S
2.D.F.S
ENTER YOUR CHOICE1
ENTER THE SOURCE VERTEX :1
1342 DO U WANT TO CONTINUE(Y/N) ? Y

MENU
1.B.F.S
2.D.F.S
ENTER YOUR CHOICE2
ENTER THE SOURCE VERTEX :1
 1 342 DO U WANT TO CONTINUE(Y/N) ? N

--------------------------------
Process exited after 123.9 seconds with return value 78
Press any key to continue . . .
```

10. Implementations of Binary Search algorithm on given list.

Program:

Iterative Method

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

int binary(int low,int high,int key,int a[100])
 {
  int mid,flag=0;
  while(low<=high)
  {
   mid=(low+high)/2;
   if(a[mid]==key)
    {
     flag=1;
     return mid;
    }
   else if(key<a[mid])
   {
    flag=0; high=mid-1;
    binary(low,high,key,a);
   }
   else
   {
    flag=0; low=mid+1;
    binary(low,high,key,a);
   }
  }
  if(flag==0) return-1;
 }
int main()
 {
  int a[100],high,low,i,n,key,result;
  printf("\n How many array elements?=");
  scanf("%d",&n);
  printf("\n Enter array element in ascending order=");
  for(i=0;i<n;i++)
  {
   scanf("%d",&a[i]);
  }
  printf("\n Enter the number that you have to search=");
  scanf("%d",&key);
  low=0;high=n-1;
  result=binary(low,high,key,a);
  if(result==-1)
   printf("Element %d is not present",key);
  else
   printf("Element %d is at index=%d",key,result); getch();
}
```

Output:

```

 How many array elements?=5


 Enter array element in ascending order=11 22 33 44 55


 Enter the number that you have to search=33
Element 33 is at index=2
--------------------------------
Process exited after 25.32 seconds with return value 0
```

```
Press any key to continue . . .
```

Recursive Method

```
#include<stdio.h>
#include<stdlib.h>
#define size 10

int binsearch(int[], int, int, int);
int main()
 {
  int num, i, key, position;
  int low, high, list[size];
  printf("\nEnter the total number of elements");
  scanf("%d", &num);
  printf("\nEnter the elements of list :");
  for (i = 0; i < num; i++)
   {
    scanf("%d", &list[i]);
   }
  low = 0;
  high = num - 1;
  printf("\nEnter element to be searched : ");
  scanf("%d", &key);
  position = binsearch(list, key, low, high);
  if (position != -1)
  {
   printf("\nNumber present at %d", (position + 1));
  }
  else
   printf("\n The number is not present in the list");
  return (0);
}

 int binsearch(int a[], int x, int low, int high)
 {
  int mid;
  if (low > high)
  return -1;
  mid = (low + high) / 2;
  if (x == a[mid])
  {
   return (mid);
  }
  else if (x < a[mid])
  {
   binsearch(a, x, low, mid - 1);
  }
  else
  {
   binsearch(a, x, mid + 1, high);
  }
 }
```

Output:

```
Enter the total number of elements5

Enter the elements of list :10 20 30 40 50

Enter element to be searched : 11
```

```
 The number is not present in the list

--------------------------------

Process exited after 26.76 seconds with return value 0

Press any key to continue . . .
```