Aim:- To write a C program that uses functions to perform the following operations

(a) To create a singly linked list of integers.

(b) To delete a given integer from the linked list

(c) To display the contents of the linked list.

Source Code:-

```
#include <stdio.h>
#include<stdlib.h>
struct node{
    int data;
    struct node *next;
};
struct node *head=NULL;
struct node *create(struct node *head){
    struct node *temp=head;
    while(1){
        struct node *newnode=(struct node *)malloc(sizeof(
                                    struct node));
        printf("Enter element : ");
        scanf("%d",&newnode->data);
        newnode->next = NULL;
        if(head==NULL)
            temp=head=newnode;
        else{
            temp->next=newnode;
            temp=temp->next;
        }
```

```
        int opt;
        printf ("Enter '0' to stop : ");
        scanf ("%d", &opt);
        if (opt==0)
            break;

    }
    return head;
}
void display (* struct node *head){
    if (head == NULL)
        printf ("List is empty\n");
    else{
        printf ("\n List elements : ");
        while (head != NULL){
            printf ("%d"->", head->data);
            head = head->next;
        }
    }
}
struct node * delete(struct node *head, int x){
    struct node *temp, *pre;
    temp = head;
    pre = NULL;
    if (head == NULL){
        printf("List is Empty\n");
        return NULL;
    }
    while (temp != NULL && temp->data != x){
        pre = temp;
        temp = temp->next;
    }
```

```
    if(temp==head){
        head=head->next;
        free(temp);
    }
    else if(temp==NULL)
        printf("Element is not found to delete\n");
    else{
        pre->next=temp->next;
        free(temp);
    }
    return head;
}
int main(){
    int option,x;
    do{
        printf("\n ***** MAIN MENU *****");
        printf("\n 1: Create a list");
        printf("\n 2: Display the list");
        printf("\n 3: Delete a node before a given node");
        printf("\n 4: Exit");
        printf("\n Enter your Option : ");
        scanf("%d",&option);
        switch(option){
            case 1:
                head=create(head);
                break;
            case 2:
                display(head);
                break;
```

```
        case 3:
                printf("Enter element to delete :");
                scanf("%d", &x);
                head = delete(head, x);
                break;
        }
    }while(option != 1);
    return 0;
}
```

Output: Sample Output-1:

```
**** MAIN MENU ****
1: Create a list
2: Display the list
3: Delete a node before a given node
4: Exit
Enter your option : 1
Enter element : 3
Enter '0' to stop : 1
Enter element : 5
Enter '0' to stop : 1
Enter element : 7
Enter '0' to stop : 1
Enter element : 9
Enter '0' to stop : 1
Enter element : 1
Enter '0' to stop : 0
```

```
*****MAIN MENU *****
1 : Create a list
2 : Display the list
3 : Delete a node before a given node
4 : Exit
Enter your Option : 2

List elements : 3 -> 5 -> 7 -> 9 -> 1 ->
***** MAIN MENU *****
1 : Create a list
2 : Display the list
3 : Delete a node before a given node
4 : Exit
Enter your option : 3
Enter element to delete : 3

***** MAIN MENU *****
1 : Create a list
2 : Display the list
3 : Delete a node before a given node
4 : Exit
Enter your option : 2

List elements : 5 -> 7 -> 9 -> 1 ->
***** MAIN MENU *****
1 : Create a list
2 : Display the list
3 : Delete a node before a given node
4 : Exit
```

Enter your Option: 3

Enter element to delete : 3

Element is not found to delete

***** MAIN MENU *****

1: Create a list

2: Display the list.

3: Delete a node before a given node

4: Exit

Enter your option: 3

Enter element to delete: 1

***** MAIN MENU *****

1: Create a list

2: Display the list

3: Delete a node before a given node

4: Exit

Enter your option: 2

List elements : 5->7->9->

***** MAIN MENU *****

1: Create a list

2: Display the list

3: Delete a node before a given node

4: Exit

Enter your option: 4

Sample Output-2:

```
***** MAIN MENU *****
1: Create a list
2: Display the list
3: Delete a node
4: Exit
Enter your option: 1
Enter element: 4
Enter '0' to stop: 1
Enter element: 3
Enter '0' to stop: 1
Enter element: 2
Enter '0' to stop: 1
Enter element: 1
Enter '0' to stop: 0

***** MAIN MENU *****
1: Create a list
2: Display the list
3: Delete a node
4: Exit
Enter your option: 2

List elements: 4 -> 3 -> 2 -> 1 ->
***** MAIN MENU *****
1: Create a list
2: Display the list
3: Delete a node
```

4: Exit
Enter your option: 3
Enter element to delete: 3

***** MAIN MENU *****
1: Create a list
2: Display the list
3: Delete a node
4: Exit
Enter your option: 2
Enter
list elements: 4 -> 2 -> 1 ->
***** MAIN MENU *****
1: Create a list
2: Display the list
3: Delete a node
4: Exit
Enter your option: 4

Aim: To write a C program that uses functions to perform the following:

(a) Creates a doubly linked list of integers

(b) Display the contents of the list

(c) Deletes a given integers from the doubly linked list

Source Code:

```
#include <stdio.h>
#include <stdlib.h>
struct node {
        int data;
        struct node *prev, * next;
};
struct node  *head = NULL;
struct node * create (struct node *head){
            struct node * temp=head;
            while(1){
                struct node *newnode= (struct node *)malloc (sizeof(
                                                    struct node));

                printf (" Enter element : ");
                scanf (" %d", & newnode -> data);
                newnode -> prev = NULL;
                newnode -> next = NULL;
                if(head == NULL)
                        temp= head= newnode;
                else {
                        newnode-> prev= temp;
                        temp-> next = newnode;
                        temp= temp->next;
                }
                int opt;
                printf (" Enter '0' to stop");
                scanf("%d", & opt);
```

```
        if(opt==0)
            break;
    }
    return head;
}
void display(struct node *head){
    if(head==NULL)
        printf("List is empty\n");
    else{
        printf("\n List elements : ");
        while(head!=NULL){
            printf("%d->", head->data);
            head=hea->next;
        }
    }
}
struct node *delete(struct node *head, int x){
    struct node *temp;
    temp=head;
    if(head==NULL){
        printf("List is empty\n");
        return NULL;
    }
    while(temp!=NULL && temp->data!=x){
        temp=temp->next;
    }
    if(temp==head){
        head=head->next;
        if(head!=NULL)
            head->prev=NULL;
    }
    free(temp);
```

break;

```
        else if (temp==NULL)
                printf("Element is not found to delete\n");
        else{
                temp->prev->next=temp->next
                if(temp->next!=NULL)
                        temp->next->prev=temp->prev;
                free(temp);
        }
        return head;
}
int main(){
        int option,x;
        do{
                printf("\n ***** MAIN MENU *****");
                printf("\n 1: Create a list");
                printf("\n 2: Display the list");
                printf("\n 3: Delete a node");
                printf("\n 4: Exit");
                printf("Enter your option :");
                scanf("%d",&option);
                switch(option){
                        case 1:
                                head=create(head);
                                break;
                        case 2:
                                display(head);
                                break;
                        case 3:
                                printf("Enter element to delete");
                                scanf("%d",&x);
                                head=delete(head,x);
                                break;
```

```
    }
  }while(option!=4);
  return 0;
}
```

Output :-

Sample output-1:

```
**** MAIN MENU ****
1: Create a list
2: Display the list
3: Delete a node
4: Exit
Enter your option : 1
Enter element: 5
Enter '0' to stop: 1
Enter element: 2
Enter '0' to stop: 1
Enter element: 9
Enter '0' to stop: 1
Enter element: 4
Enter '0' to stop: 0
****MAIN MENU ****
1: Create a list
2: Display a list
3: Delete a node
4: Exit
Enter your option: 2

List elements: 5-> 2-> 9-> 4->
```

***** MAIN MENU *****

1: Create a list
2: Display the list
3: Delete a node
4: Exit
Enter your option: 3
Enter element to delete : 5

***** MAIN MENU *****

1: Create a list
2: Display the list
3: Delete a node
4: Exit
Enter your option: 3
Enter element to delete: 9

***** MAIN MENU *****

1. Create a list
2: Display the list
3: Delete a node
4: Exit
Enter your option: 2

List elements : 2 -> 4 ->

***** MAIN MENU *****

1: Create a list
2: Display the list
3: Delete a node
4: Exit
Enter your option: 4

Sample Output 2 :-


```
***** MAIN MENU *****
1: Create a list
2: Display the list
3: Delete a node
4: Exit
Enter your option : 1
Enter element: 1
Enter '0' to stop : 1
Enter element : 2
Enter '0' to stop: 1
Enter element: 3
Enter '0' to stop: 1
Enter element: 4
Enter '0' to stop: 1
Enter element: 5
Enter '0' to stop: 0

***** MAIN MENU*****
1: Create a list
2: Display the list:
3: Delete a node
4: Exit
Enter your option: 2

List elements: 1-> 2-> 3-> 4->5->
***** MAIN MENU *****
1: Create a list
2: Display the list
3: Delete a node
```

printf (" stack elements are : ");

4: Exit
Enter your option: 3
Enter element to delete: 1

***** MAIN MENU *****
1: Create a list
2: Display the list
3: Delete a node
4: Exit
Enter your option: 3
Enter element to delete: 6
Element is not found to delete.

***** MAIN MENU *****
1: Create a list
2: Display the list
3: Delete a node
4: Exit
Enter your option: 2

List elements: 2-> 3-> 4-> 5->

***** MAIN MENU *****
1: Create a list
2: Display the list
3: Delete a node
4: Exit
Enter your option: 4

Aim:- To write a C program to implement the stack ADT using arrays.

Source Code:-

```
#include <stdio.h>
#define N 20
int stack[N];
int top=-1;
void push(int stack[], int x){
        if(top==N-1)
                printf("Overflow\n");
        else
                stack[++top]=x;
}
int pop(int stack[]){
        if(top==-1){
                printf("Underflow\n");
                return -1;
        }
        return stack[top--];
}
int peek(int stack[]){
        if(top==-1)
                return -1;
        return stack[top];
}
void display(){
        if(top==-1)
                printf("stack is empty\n");
        else{
                printf("stack elements are: ");
```

```
        for(int i= top; i>=0; i--)
            printf("%od", stack[i]);
    }
}
int main(){
    int val, option;
    do{
        printf("\n ***** MAIN MENU *****");
        printf("\n 1. PUSH");
        printf("\n 2. POP");
        printf("\n 3. PEEK");
        printf("\n 4. DISPLAY");
        printf("\n 5. Exit");
        printf("\n Enter your option: ");
        scanf("%od", &Option);
        switch (option){
            case 1:
                printf("\n Enter the number to push:");
                scanf("%d", &val);
                push(stack, val);
                break;
            case 2:
                val = pop(stack);
                if (val != -1)
                    printf("\n The value
                    deleted from stack is %d",
                    val);
                break;
            case 3:
                val = peek(stack);
```

```
            if (val!=-1)
                printf("\n The value stored at the top is %d",
                        val);
                break;
            case 4:
                display (stack);
                break;
        }
    }while (option != 5);
    return 0;
}
```

Output:

Sample Output - 1:

```
***** MAIN MENU *****
1. PUSH
2. POP
3. PEEK
4. DISPLAY
5. Exit
Enter your option : 1

Enter the number to be pushed on stack : 5
***** MAIN MENU *****
1. PUSH
2. POP
3. PEEK
4. DISPLAY
5. EXIT
Enter your option : 1
```

Enter the number to push: 3

*** ** MAIN MENU *****

1. PUSH
2. POP
3. PEEK
4. DISPLAY
5 EXIT

Enter your option: 1

Enter the number to push: 6

***** MAIN MENU *****

1. PUSH
2. POP
3. PEEK
4. DISPLAY
5. EXIT

Enter your option: 1

Enter the number to push: 8

***** MAIN MENU *****

1. PUSH
2. POP
3. PEEK
4. DISPLAY
5. EXIT

Enter your option: 4

Stack elements are: 8 6 3 5

***** MAIN MENU *****

1. PUSH
2. POP
3. PEEK

4. DISPLAY
5. EXIT
Enter your option: 1
Enter the number to push: 1

**** MAIN MENU ****
1. PUSH
2. POP
3. PEEK
4. DISPLAY
5. EXIT
Enter your option: 3

The value stored at the top of stack is 1

**** MAIN MENU ****
1. PUSH
2. POP
3. PEEK
4. DISPLAY
5. EXIT
Enter your option : 2

The value deleted from stack is 1

**** MAIN MENU ****
1. PUSH
2. POP
3. PEEK
4. DISPLAY
5. EXIT
Enter your option : 5

Output:-

Sample Output-2:

```
***** MAIN MENU *****
1. PUSH
2. POP
3. PEEK
4. DISPLAY
5 EXIT
Enter your option : 1

Enter the number to push: 5

***** MAIN MENU *****
1. PUSH
2. POP
3. PEEK
4. DISPLAY
5. EXIT
Enter your option: 1

Enter the number to push: 3

***** MAIN MENU*****
1. PUSH
2. POP
3. PEEK
4. DISPLAY
5. EXIT
Enter your option: 2

The deleted value from stack is 3
***** MAIN MENU *****
1. PUSH
2. POP
```

3. PEEK
4. DISPLAY
5. EXIT
Enter your option : 3

The value stored at top of stack is: 5
***** MAIN MENU *****
1. PUSH
2. POP
3. PEEK
4. DISPLAY
5. EXIT
Enter your option: 1

Enter the number to push : 2
***** MAIN MENU *****
1. PUSH
2. POP
3. PEEK
4. DISPLAY
5. EXIT
Enter your option: 4
Stack elements are: 2 5
***** MAIN MENU *****
1. PUSH
2. POP
3. PEEK
4. DISPLAY
5. EXIT
Enter your option : 5

Aim:- To write a C program to implement Stack ADT using Linked list.

Source Code:-

```
#include<stdio.h>
#include<stdlib.h>
struct node{
        int data;
        struct node *next;
};
struct node *top=NULL;
struct node * push(struct node *top, int x){
        struct node *newnode=(struct node *)malloc(sizeof(struct node
                                                        ));
        newnode->next=top;
        newnode->data=x;
        top=newnode;
        return top;
}
struct node *pop(struct node *top){
        if (top==NULL){
                printf("Underflow\n");
                return NULL;
        }
        printf("popped element is %d", top->data);
        struct node *temp=top;
        top=top->next;
        free(temp);
        return top;
}
```

```
int peek(struct node *top){
        if(top==NULL)
            return -1;
        return top->data;
}
void display(struct node *top){
        if(top==NULL)
            printf("Stack is empty\n");
        else{
            printf("Stack elements are : ");
            struct node *temp=top;
            while(temp!=NULL){
                printf("%d ",temp->data);
                temp=temp->next;
            }
        }
}
int main(){
    int val,option;
    do{
        printf("\n ***** MAIN MENU *****");
        printf("\n 1. Push");
        printf("\n 2. Pop");
        printf("\n 3. Peek");
        printf("\n 4. Display");
        printf("\n 5. Exit");
        printf("\n Enter your option : ");
        scanf("%d",&option);
        switch(option){
            case 1:
                printf("Enter the number to be pushed:");
```

```
            scanf("%d",&val);
            top = push(top,val);
            break;
        case 2:
            top= pop(top);
            break;
        case 3:
            val= peek(top);
            if(val != 1)
                printf("\n The value stored at top of stack
                        is : %d", val);
            break;
        case 4:
            display(top);
            break;
        }
    }while(option!=5);
    return 0;
}
```

Output:-

Sample output-1:

```
**** * MAIN MENU **** *
1. PUSH
2. POP
3. PEEK
4. DISPLAY
5. EXIT
Enter your option : 1
Enter the number to be pushed : 1
```

***** MAIN MENU *****

1. PUSH
2. POP
3. PEEK
4. DISPLAY
5. EXIT

Enter your option: 1

Enter the number to be pushed : 3

***** MAIN MENU *****

1. PUSH
2. POP
3. PEEK
4. DISPLAY
5. EXIT

Enter your option: 1

Enter the number to be pushed: 4

***** MAIN MENU *****

1. PUSH
2. POP
3. PEEK
4. DISPLAY
5. EXIT

Enter your option : 4

Stack elements are: 4 3 1

***** MAIN MENU *****

1. PUSH
2. POP
3. PEEK
4. DISPLAY
5. EXIT

Enter your option: 2

Popped element is 4

***** MAIN MENU *****

1. PUSH

2. POP

3. PEEK

4. DISPLAY

5. EXIT

Enter your option : 3

The value stored at top of stack is : 3

***** MAIN MENU *****

1. PUSH

2. POP

3. PEEK

4. DISPLAY

5. EXIT

Enter your option : 5

SAMPLE OUTPUT - 2 :

***** MAIN MENU *****

1. PUSH

2. POP

3. PEEK

4. DISPLAY

5. EXIT

Enter your option : 1

Enter the value to be pushed : 2

***** MAIN MENU *****

1. PUSH

2. POP

3. PEEK
4. DISPLAY
5. EXIT
Enter your option: 2

Popped element is 2
***** MAIN MENU *****
1. PUSH
2. POP
3. PEEK
4. DISPLAY
5. EXIT
Enter your option: 4

Stack is empty
***** MAIN MENU *****
1. PUSH
2. POP
3. PEEK
4. DISPLAY
5. EXIT
Enter your option: 1

Enter the value to be pushed: 9

***** MAIN MENU *****
1. PUSH
2. POP
3. PEEK
4. DISPLAY
5. EXIT
Enter your option: 1

Enter the value to be pushed: 4

***** MAIN MENU *****

1. PUSH
2. POP
3. PEEK
4. DISPLAY
5. EXIT

Enter your option: 1

Enter the value to be pushed: 1

***** MAIN MENU *****

1. PUSH
2. POP
3. PEEK
4. DISPLAY
5. EXIT

Enter your option: 4

Stack elements are: 1 4 9

***** MAIN MENU *****

1. PUSH
2. POP
3. PEEK
4. DISPLAY
5. EXIT

Enter your option: 5

Aim:- To write a C program to convert a infix expression into its postfix expression using stack operations.

Source Code:-

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>
char stack[30];
int top=-1;
void push(char x){
        stack[++top]=x;
}
char pop(){
        return stack[top--];
}
void display(){
int pre(char x){
        if(x=='*' || x=='/' || x=='%')
                return 1;
        else if(x=='+' || x=='-')
                return 0;
}
int main(){
        char infix[40],post[40];
        printf("Enter expression : ");
        fgets(infix, 40, stdin);
        strcpy(post,"");
        int i=0,k=0;
```

```
while (infix[i]!='\0'){
        if (infix[i] == '('){
            push(infix[i]);
        }
        else  if (infix[i]==')'){
            while(top!=-1  && stack[top]!='(')
                post[k++] = pop();
            pop();
        }
        else if (isdigit(infix[i]) || isalpha(infix[i]))
            post[k++] = infix[i];
        else{
            while (top!= 1 && stack[top]!='(' &&
                pre(infix[i]<= pre(stack[top]))
            post[k++]= pop();
            push(infix[i]);
        }
        i++;
    }
while( (top!=-1)  && stack[top]!='(')
        post[k++] = pop(stack);
post[k]='\0';
printf (" Postfix expression is : ");
puts(post);
}
```

Output:-

Sample output-1:

Enter expression: a+b*(c*e-e)/(f+g*h)-i

Postfix expression is: abce*e-fgh*+/+i-

Sample output-2:

Enter expression: A+(B/C-(D*E/F)+G)*B

Postfix expression is: ABC/DE*F/-G+B*+

Aim: To write a C program to evaluate a postfix expression.

Source Code:

```
#include<stdio.h>
#include<string.h>
#include<ctype.h>
int top=-1;
float stack[20];
void push(float x){
    stack[++top]=x;
}
float pop(){
    return stack[top--];
}
int main(){
    char post[20];
    printf("Enter expression : ");
    fgets(post,20,stdin);
    int flag=0;
    for(int i=0;post[i]!='\0';i++){
        if(isdigit(post[i]))
            push(post[i]-'0');
        else{
            if(top>=1){
                float val1=pop(),val2=pop();
                switch(post[i]){
                    case'+':
                        push(val1+val2);
                        break;
```

```
                case '-':
                        push(val2 - val1);
                        break;
                case '*':
                        push(val2 * val1);
                        break;
                case '/':
                        push(val2 / val1);
                        break;
                case '%':
                        push(val2 % val1);
                        break;
            }
        }
        else {flag=1;
              printf(" Invalid Expression\n");
              break;
        }
    }
}
if (flag==0)
        printf("\nExpression Value = %.2f\n", pop());
}
```

Output:

Sample output-1:

Enter expression: 4572+-*

Expression value = -16

Sample output-2:

Enter expression: 42+351-*+

Expression value = 18

Aim: To write a C program to implement queue ADT

(a) Using array

Source Code:

```
#include<stdio.h>
#define N 20
int rear=-1, front=-1, queue[N];
void enque(int x){
    if(rear==N-1)
        printf("Overflow\n");
    else if(rear==-1 && front==-1){
        front=rear=0;
        queue[rear]=x;
    }
    else
        queue[++rear]=x;
}
int deque(){
    int data;
    if(rear==-1 && front==-1){
        printf("Underflow\n");
        return -1;
    }
    else if(front==rear){
        data=queue[front];
        front=rear=-1;
    }
    else
        data=queue[front++];
    return data;
}
```

```
void display(){
    if(rear==-1 && front==-1)
        printf("Queue is empty\n");
    else{
        printf("Queue elements are : ");
        for(int i=front; i<=rear; i++){
            printf("%d", queue[i]);
        }
    }
}
int peek(){
    if (front==-1)
        return -1;
    return queue[front];
}
int main(){
    int option, val, x;
    do{
        printf("\n ***** MAIN MENU *****");
        printf("\n 1. Insert an element");
        printf("\n 2. Delete an element");
        printf("\n 3. Peek");
        printf("\n 4. Display the queue");
        printf("\n 5. Exit");
        printf("\n Enter your option");
        scanf("%d", &option);
        switch(option){
            case 1:
                printf("Enter element to be added:");
                scanf("%d", &x);
```

```
            enque(x);
            break;
        case 2:
            val = deque();
            if(val != -1)
                printf("\n The number deleted is : %od", val);
            break;
        case 3:
            val = peek();
            if (val != -1)
                printf("\n The front value in queue is
                    : %d", val);
            break;
        case 4:
            display();
            break;
    }
} while(option != 5);
}
```

Output:-

Sample Output-1:-

```
**** MAIN MENU *****
1. Insert an element
2. Delete an element
3. Peek
4. Display the queue
5. Exit
Enter your option : 1
Enter element to be added: 5
```

**** MAIN MENU ****
1. Insert an element
2. Delete an element
3. Peek
4. Display the queue
5. Enter element
5. Exit

Enter your option: 1
Enter elemented to be added: 4

**** MAIN MENU ****
1. Insert an element
2. Delete an element
3. Peek
4. Display the queue
5. Exit

Enter your option: 2
The number deleted is: 5

**** MAIN MENU ****
1. Insert an element
2. Delete an element
3. Peek
4. Display the queue
5. Exit

Enter your option: 3
The front value in queue is: 4

**** MAIN MENU ****
1. Insert an element
2. Delete an element
3. Peek
4. Display the queue

5. Exit

Enter your option: 1

Enter element to be added: 6

***** MAIN MENU *****

1. Insert an element

2. Delete an element

3. Peek

4. Display the queue

5. Exit

Enter your option: 4

Queue elements are : 4, 6

***** MAIN MENU *****

1. Insert an element

2. Delete an element

3. Peek

4. Display the queue

5. Exit

Enter your option: 5

Sample Output - 2 :

***** MAIN MENU *****

1. Insert an element

2. Delete an element

3. Peek

4. Display the queue

5. Exit

Enter your option: 1

Enter element to be added : 4

***** MAIN MENU *****

2. Delete an element
3. Peek
4. Display the queue
5. Exit

Enter your option: 1

Enter element to be added: 6

***** MAIN MENU *****

1. Insert an element
2. Delete an element
3. Peek
4. Display the queue
5. Exit

Enter your option: 1

Enter element to be added: 9

***** MAIN MENU *****

1. Insert an element
2. Delete an element
3. Peek
4. Display the queue
5. Exit

Enter your option: 1

Enter element to be added: 5

***** MAIN MENU *****

1. Insert an element
2. Delete an element
3. Peek
4. Display
5. Exit

Enter your option: 2

The number deleted is 4

***** MAIN MENU *****

1. Insert an element
2. Delete an element
3. Peek
4. Display the queue
5. Exit

Enter your option: 3

The front value in queue is: 6

***** MAIN MENU *****

1. Insert an element
2. Delete an element
3. Peek
4. Display the queue
5. Exit

Enter your option: 4

Queue elements are: 6, 9, 5

***** MAIN MENU *****

1. Insert an element
2. Delete an element
3. Peek
4. Display the queue
5. Exit

Enter your option: 5

Aim :- To write a C program to implement queue ADT using doubly linked list.

Source Code :-

```
#include<stdio.h>
#include<stdlib.h>
struct node{
        int data;
        struct node *prev,*next;
};
struct node *rear=NULL,*front=NULL;
void enque(int x){
        struct node *new=(struct node*) malloc(sizeof(struct node));
        new->data=x;
        new->prev=NULL;
        new->next=NULL;
        if(rear==NULL && front==NULL){
                front=rear=new;
        }
        else{
                rear->next=new;
                new->prev=rear;
                rear=new;
        }
}
int deque(){
        int val;
        if(rear==NULL && front==NULL)
                return-1;
        struct node *temp=front;
        val=front->data;
        front=front->next;
```

```
        if(front==NULL){
                rear=NULL;
        }
        else{
                front->prev=NULL;
        }
        free(temp);
        return val;
}
int peek(){
        if(front==NULL)
                return -1;
        return front->data;
}
void display(){
        if(rear==NULL && front==NULL)
                printf("Queue is Empty\n");
        else{
                printf("Queue elements are: ");
                struct node *temp=front;
                while(temp!=NULL){
                        printf("%d->",temp->data);
                        temp=temp->next;
                }
        }
}
int main(){
        int option,val,x;
        do{
                printf("\n ***** MAIN MENU *****");
                printf("\n 1. Insert an element");
                printf("\n 2. Delete an element");
```

```
printf("\n 3. Peek");
printf("\n 4. Display the queue");
printf("\n 5. Exit");
printf("\n Enter your option: ");
scanf("%od", &option);
switch (option){
        case 1:
            printf("Enter element to be added : ");
            scanf("%d", &x);
            enque(x);
             break;
        case 2:
            val = deque();
            if(val != -1)
                printf("\n The deleted number is: %od", val);
            break;
        case 3:
            val = peek();
            if(val != -1)
                printf("\n The front value in queue is:
                                %od ", val);

            break;
        case 4:
            display();
            break;
    }
} while (option != 5);
```

Output :-

Sample output-1 :

```
***** MAIN MENU *****
1. Insert an element
2. Delete an element
3. Peek
4. Display the queue
5. Exit
Enter your option: 1
Enter element to be added: 7

***** MAIN MENU *****
1. Insert an element
2. Delete an element
3. Peek
4. Display the queue
5. Exit
Enter your option: 1
Enter element to be added: 3

***** MAIN MENU *****
1. Insert an element
2. Delete an element
3. Peek
4. Display the queue
5. Exit
Enter your option: 1
Enter element to be added: 9

***** MAIN MENU *****
1. Insert an element
2. Delete an element
```

3. Peek
4. Display the queue
5. Exit
Enter your option: 4

Queue elements are : 7 -> 3 -> 9 ->

***** MAIN MENU *****

1. Insert an element
2. Delete an element
3. Peek
4. Display the queue
5. Exit
Enter your option: 2

The deleted number is : 7

***** MAIN MENU *****

1. Insert an element
2. Delete an element
3. Peek
4. Display the queue
5. Exit
Enter your option: 3

The front value in queue is : 3

***** MAIN MENU *****

1. Insert an element
2. Delete an element
3. Peek
4. Display the queue
5. Exit
Enter your option: 1

Enter element to be added: 5

***** MAIN MENU *****

1. Insert an element
2. Delete an element
3. Peek
4. Display the queue
5. Exit

Enter your option: 4

Queue elements are: 3 -> 9 -> 5 ->

***** MAIN MENU *****

1. Insert an element
2. Delete an element
3. Peek
4. Display the queue
5. Exit

Enter your option: 5

Sample Output-2:

***** MAIN MENU *****

1. Insert an element
2. Delete an element
3. Peek
4. Display the queue
5. Exit

Enter your option: 4

QUEUE IS EMPTY

***** MAIN MENU *****

1. Insert an element
2. Delete an element

3. Peek
4. Display the queue
5. Exit

Enter your option: 1

Enter element to be added: 5

***** MAIN MENU *****
1. Insert an element
2. Delete an element
3. Peek
4. Display the queue
5. Exit

Enter your option: 1

Enter element to be added: 3

***** MAIN MENU *****
1. Insert an element
2. Delete an element
3. Peek
4. Display the queue
5. Exit

Enter your option: 1

Enter element to be added: 7

***** MAIN MENU *****
1. Insert an element
2. Delete an element
3. Peek
4. Display
5. Exit

Enter your option: 2

```
The number deleted is : 5

***** MAIN MENU *****
1. Insert an element
2. Delete an element
3. Peek
4. Display the queue
5. Exit
Enter your option : 2
The number deleted is : 3

***** MAIN MENU*****
1. Insert an element
2. Delete
3. Peek
4. Display the queue
5. Exit
Enter your option. 3
The front value in Queue is: 7->
***** MAIN MENU*****
1. Insert an element
2. Delete
3. Peek
4. Display the queue
5. Exit
Enter your option : 5
```

Aim:- To write a C program to implement Circular queue ADT using array

Source Code:-

```
#include<stdio.h>
#define N 10
int rear=-1, front=-1, queue[N];
void enque(int x){
    if((rear+1)%N==front)
        printf("Overflow\n");
    else if(rear==-1 && front==-1){
        front=rear=0;
        queue[rear]=x;
    }
    else{
        rear=(rear+1)%N;
        queue[rear]=x;
    }
}
int deque(){
    int data;
    if(rear==-1 && front==-1){
        printf("Underflow\n");
        return -1;
    }
    data=queue[front];
    if(front==rear)
        front=rear=-1;
    else
        front=(front+1)%N;
```

```
    return data;
}
void display(){
    if(rear== -1  && front==-1)
        printf(" Circular Queue is Empty\n");
    else{
        int i=front;
        printf(" Queue elements are \n");
        while( i!=rear) {
            printf("%d ",queue[i]);
            i=(i+1)%N;
        }
        printf("%d ", queue[i]);
    }
}
int peek(){
    if (front== -1)
        return -1;
    return queue[front];
}
int main(){
    int option, val, x;
    do{
        printf("\n ***** MAIN MENU *****");
        printf ("\n 1. Insert an element");
        printf ("\n 2. Delete an element");
        printf ("\n 3. Peek");
        printf (" \n 4. Display the Queue");
        printf ("\n 5. Exit");
        printf (" \n 6. Enter your option ");
```

```
scanf("%d", &option);
switch(option){
        case 1:
            printf("Enter element to be added : ");
            scanf("%d", &x);
            enque(x);
            break;
        case 2:
            printf("In The deleted number is : ");
            val = deque();
            if( val != 1)
                printf("%d", val);
            break;
        case 3:
            val = peek();
            if (val != -1)
                printf("In The front value in queue is:
                    %d", val);
            break;
        Case 4:
            display();
            break;
    }
} while (option != 5);
}
```

Output:

Sample Output-1:

```
***** MAIN MENU ***** *
1. Insert an element
2. Delete an element
3. Peek
```

4. Display the queue
5. Exit
Enter your option: 1
Enter element to be added: 3

***** MAIN MENU *****
1. Insert an element
2. Delete an element
3. Peek
4. Display the queue
5. Exit
Enter your option: 1
Enter element to be added: 6

***** MAIN MENU *****
1. Insert an element
2. Delete an element
3. Peek
4. Display the queue
5. Exit
Enter your option: 1
Enter element to be added: 9

***** MAIN MENU *****
1. Insert an element
2. Delete an element
3. Peek
4. Display the queue
5. Exit
Enter your option: 4
Queue elements are: 3 6 9

```
***** MAIN MENU *****
1. Insert an element
2. Delete an element
3. Peek
4. Display the queue
5. Exit
Enter your option: 2
The deleted number is : 3
***** MAIN MENU *****
1. Insert an element
2. Delete an element
3. Peek
4. Display the queue
5. Exit
Enter your option: 2
The deleted number is : 6
***** MAIN MENU *****
1. Insert an element
2. Delete an element
3. Peek
4. Display the queue
5. Exit
Enter your option: 2
The deleted number is : 9
***** MAIN MENU *****
1. Insert an element
2. Delete an element
3. Peek
4. Display the queue
5. Exit
```

Enter your option: 4

QUEUE IS EMPTY:

***** MAIN MENU *****

1. Insert an element
2. Delete an element
3. Peek
4. Display the queue
5. Exit

Enter your option: 5

Sample Output-2:

***** MAIN MENU *****

1. Insert an element
2. Delete an element
3. Peek
4. Display the queue
5. Exit

Enter your option: 1

Enter element to be added: 2

***** MAIN MENU *****

1. Insert an element
2. Delete an element
3. Peek
4. Display the queue
5. Exit

Enter your option: 1

Enter element to be added: 4

***** MAIN MENU *****

1. Insert an element
2. Delete an element

3. Peek
4. Display the queue
5. Exit
Enter your option: 1
Enter element to be added: 6
***** MAIN MENU *****
1. Insert an element
2. Delete an element
3. Peek
4. Display the queue
5. Exit
Enter your option: 2
The deleted number is: 2
***** MAIN MENU *****
1. Insert an element
2. Delete an element
3. Peek
4. Display the queue
5. Exit
Enter your option: 1
Enter element to be added: 8
***** MAIN MENU *****
1. Insert an element
2. Delete an element
3. Peek
4. Display the queue
5. Exit
Enter your option: 3

The front value in queue is : 4

***** MAIN MENU *****

1. Insert an element
2. Delete an element
3. Peek
4. Display the queue
5. Exit

Enter your option : 4

Queue elements are : 4 6 8

***** MAIN MENU *****

1. Insert an element
2. Delete an element
3. Peek
4. Display the queue
5. Exit

Enter your option : 5

Aim:- To write a C program to implement insertion sort.

Source Code:-

```
#include<stdio.h>
int insertion (int arr[], int n) {
        for (int i= 1, i<n; i++){
                int gap= i, key = arr[i];
                while (gap >0  && arr[gap-1] >key){
                        arr[gap]= arr[gap-1];
                        gap = gap-1;
                }
                arr[gap]= key;
        }
}
int main() {
        int i,n;
        printf ("\n Enter no.of array elements : ");
        scanf (" %od", &n);
        int arr[n];
        for(i=0 ; i<n; i++){
                printf ("\n Enter array element : ");
                scanf ("%od", &arr[i]);
        }
        printf("\n array elements before sorting : ");
        for (i=0; i<n ; i++)
                printf ("%od ", arr[i]);
        insertion (arr, n);
        printf (" array elements after sorting : ");
        for(i=0; i<n; i++)
                printf (" %od", arr[i]);
        return 0;
}
```

Output:-

Sample Output-1:-

```
Enter no.of array elements:5
Enter array element:7
Enter array element:6
Enter array element:5
Enter array element:2
Enter array element:3
array elements before sorting: 7 6 5 2 3
array elements after sorting: 2 3 5 6 7
```

Sample output-2:-

```
Enter no.of array elements: 6
Enter array element:6
Enter array element:3
Enter array element:1
Enter array element:5
Enter array element:8
Enter array element:3
array elements before sorting: 6 3 1 5 8 3
array elements after sorting: 1 3 3 5 6 8
```

Aim :- To write a C program for implementing the Merge Sort

Source Code :

```
#include<stdio.h>
int merge(int arr[], int l, int mid, int r){
        int a[r],i=l, j=mid+1, k=l;
        while(i<=mid && j<=r){
                if(arr[i]<arr[j])
                        a[k++]=arr[i++];
                else
                        a[k++]=arr[j++];
        }
        while(i<=mid)
                a[k++]=arr[i++];
        while(j<=r)
                a[k++]=arr[j++];

        for(i=l; i<=r; i++)
                arr[i]=a[i];
}
int mergesort(int arr[], int l, int r){
        if(l<r){
                int mid=(l+r)/2;
                mergesort(arr,l,mid);
                mergesort(arr,mid+1,r);
                merge(arr,l,mid,r);
        }
}
int main(){
        int i,n;
        printf("\n Enter no of array element : ");
        scanf("%d",&n);
        int arr[n];
```

```
for(i=0; i<n ;i++){
        printf("\nEnter array elements : ");
        scanf ("%d", &arr[i]);
}
printf("\n array elements before sorting : ");
for(i=0; i<n ; i++)
        printf(" %d ", arr[i]);
for(
mergesort (arr , 0,n-1);
printf("\n array elements after before Sorting : ");
for(i=0; i<n; i++)
        printf(" %d ", arr[i]);
return 0;
}
```

Output:-

Sample output-1:

```
Enter no of array elements : 5
Enter array element: 8
Enter array element: 3
Enter array element: 2
Enter array element: 1
Enter array element: 8
array elements before sorting: 8 3 2 1 8
array elements after sorting: 1 2 3 8 8
```

Sample output-2:

```
Enter no of array elements : 6
Enter array element : 1
Enter array element : 6
Enter array element : 3
```

Enter array element : 2
Enter array element : 7
Enter array element : 8
array before sorting : 1 6 3 2 7 8
array elements after sorting : 1 2 3 6 7 8

Aim:- To write a C program to implement Quick sort

Source Code:-

```
#include <stdio.h>
void swap(int *a, int *b){
    int temp = *a;
    *a = *b;
    *b = temp;
}
int partition(int arr[], int low, int high){
    int pivot = arr[high];
    int i = low - 1;
    for(int j = low; j <= high - 1; j++){
        if(arr[j] < pivot){
            i++;
            swap(&arr[i], arr[j]);
        }
    }
    swap(&arr[i+1], &arr[high]);
    return (i+1);
}
void quicksort(int arr[], int low, int high){
    if(low < high){
        int pi = partition(arr, low, high);
        quicksort(arr, low, pi-1);
        quicksort(arr, pi+1, high);
    }
}
```

```
int main(){
    int i,n;
    printf("\nEnter no.of array elements : ");
    scanf("%d", &n);
    int arr[n];
    for(i=0; i<n; i++){
        printf("\nEnter array element : ");
        scanf("%d", &arr[i]);
    }
    printf("\n arrary elements before sorting : ");
    for(i=0; i<n; i++)
        printf("%d ", &arr[i]);
    quicksort(arr, 0, n-1);
    printf("\n array elements after sorting : ");
    for(i=0; i<n; i++)
        printf("%d ", arr[i]);
    return 0;
}
```

Output :-

Sample Output-1:

```
Enter no.of array elements : 5
Enter array element : 3
Enter array element : 2
Enter array element : 6
Enter array element : 1
Enter array element : 6
array elements before sorting: 3 2 6 1 6
array elements after sorting: 1 2 3 6 6
```

Sample Output-2 :-

```
Enter no.of array element : 6
Enter array element: 6
Enter array element: 4
Enter array element: 9
Enter array element: 3
Enter array element: 1
Enter array element: 8
array elements before sorting: 6 4 9 3 1 8
array elements after sorting: 1 3 4 6 8 9
```

Aim:- To write a C program to implement selection sort.

Source Code:-

```
#include <stdio.h>
int selection(int arr[], int n){
        for(int i=0; i<n; i++){
                int m=i;
                for(int j=i+1; j<n; j++){
                        if(arr[j]<arr[m])
                                m=j;
                }
                int temp=arr[m];
                arr[m]=arr[i];
                arr[i]=temp;
        }
}
int main(){
        int i,n;
        printf("\n Enter no. of array elements : ");
        scanf("%d", &n);
        int arr[n];
        for(int j=0; j<n; j++){
                printf("\n Enter array element : ");
                scanf("%d", &arr[j]);
        }
        printf("\n array elements before sorting : ");
        for(i=0; i<n; i++){
                printf("%d ", arr[i]);
        }
        selection(arr,n);
        printf("\nArray elements after sorting : ");
        for(i=0; i<n; i++)
                printf("%d", arr[i]);
        return 0;
}
```

Output :-

Sample Output - 1 :-

```
Enter no.of array elements: 5
Enter array element: 5
Enter array element: 2
Enter array element: 1
Enter array element: 6
Enter array element: 7
Array elements before sorting: 5 2 1 6 7
Array elements after sorting: 1 2 5 6 7
```

Sample output -2:-

```
Enter no.of array elements: 6
Enter array element: 5
Enter array element: 4
Enter array element: 3
Enter array element: 2
Enter array element: 1
Enter array element: 0
array elements before sorting: 5 4 3 2 1 0
array elements after sorting: 0 1 2 3 4 5
```

Aim :- To write a C program that uses functions to perform
(a) to create a Binary Search tree (BST)
(b) to Insert data in BST
(c) to traverse the BST recursively in postorder
(d) to delete an element in BST.

Source Code :-

```
#include <stdio.h>
#include<stdlib.h>
struct node{
        int data;
        struct node * left,* right;
};
struct node *root= NULL;
struct node * newNode (int item){
        struct node *temp;
        temp=(struct node *)malloc(sizeof(struct node));
        temp->data =item;
        temp->left = temp->right =NULL;
        return temp;
}
struct node *insert (struct node *node, int key){
        if ( node== NULL)
                return newNode(key);
        if ( key< node->data)
                node->left= insert (node->left, key);
        else if
                node->right = insert ( node->right, key);
        return node;
}
```


break;

```
struct node * minValueNode (struct node * node){
        struct node *current =node;
        while (current && current -> left != NULL)
                current = current -> left;
        return current;
}
struct node *delete (struct node * root, int key){
        if (root == NULL)
                return root;
        if (key < root -> data)
                root -> left = delete (root -> left, key);
        else if (key > root -> data)
                root -> right = delete (root -> right, key);
        else {
                if (root -> left == NULL){
                struct node * temp= root -> right;
                free (root);
                return temp;
        }
        else if (root -> right = NULL){
                struct node * temp = root -> left;
                free(root);
                return temp;
        }
        struct node *temp= minValueNode (root -> right);
        root -> data = temp -> data
        root -> right = del (root -> right, temp -> data);
        }
        return root;
}
```

```
void postorder (struct node *root){
    if (root != NULL) {
        postorder(root->left);
        postorder(root->right);
        printf ("%d", root->data);
    }
}
int main() {
    int option, x, k;
    do{
        printf ("\n ***** MAIN MENU**** *");
        printf ("\n1: Create a BINARY SEARCH TREE");
        printf ("\n2: Display the BST in post order");
        printf ("\n 3: Delete a node in BST");
        printf ("\n 4: Insert a node in BST");
        printf ("\n 5: Exit");
        printf ("\n Enter your option : ");
        scanf ("%d", &option);
        switch (option) {
            case 1:
                do{
                    printf ("Enter the value : ");
                    scanf ("%d", &x);
                    root = insert (root, x);
                    printf ("If want to stop enter '0':");
                    scanf ("%d", &k);
                } while (k != 0);
                break;
            case 2:
                printf ("\n POST ORDER : ");
                post order (root);
                break;
```

```
            case 3:
                printf("Enter the value to delete : ");
                scanf("%d",&x);
                root = delete(root, x);
                break;
            case 4:
                printf("Enter the value to insert : ");
                scanf("%d", &x);
                root = insert(root, x);
                break;
        }
    }while(option!=5);
}
```

Output :-

Sample Output-1 :-

```
***** MAIN MENU *****
1: Create a BINARY SEARCH TREE
2: DISPLAY the BST in postorder
3: Delete a node in BST
4: Insert a node in BST
5: Exit
Enter your option: 1
Enter the value: 9
If u want to enter stop more value enter '0': 1
Enter the value: 5
If u want to stop enter '0': 1
Enter the value: 11
If u want to stop enter '0': 1
Enter the value: 10
If u want to stop enter '0': 1
Enter the value: 7
```

Enter the value : 2

If u want to stop enter '0': 0

** * * * * MAIN MENU * * * * *

1: Create a BINARY SEARCH TREE

2: Display the BST in postorder

3: Delete a node in BST

4: Insert a node in BST

5: Exit

Enter your option: 2

POST ORDER: 2 7 5 10 11 9

* * * * * MAIN MENU * * * * *

1: Create a BINARY SEARCH TREE

2: DISPLAY the BST in post order

3: Delete a node in BST

4: Insert a node in BST

5: Exit

Enter your option: 3

Enter element to delete : 9

* * * * * MAIN MENU * * * * *

1: Create a BINARY SEARCH TREE

2: Display the BST in postorder

3: Delete a node in BST

4: Insert a node in BST

5: Exit

Enter your option: 2

POST ORDER: 2 7 5 11 10

** *** MAIN MENU *****
1: Create a BINARY SEARCH TREE
2: Display the BST in postorder
3: Display the BST in p
3: Delete a node in BST
4: Insert a node in BST
5 Exit
Enter your option: 4
Enter the value to in insert : 8

***** MAIN MENU *****
1: Create a BINARY SEARCH TREE
2: Display the BST in postorder
3: Delete a node in BST
4: Insert a node in BST
5: Exit
Enter your option: 2
POST ORDER : 2 8 7 5 11 10

***** MAIN MENU **** *
1: Create a BINARY SEARCH TREE
2: Display the BST in postorder
3: Delete a node in BST
4: Insert a node in BST
5: Exit
Enter your option: 5

Sample Output-2:-

```
***** MAIN MENU *****
1: Create a BINARY SEARCH TREE
2: Display the BST in postorder
3: Delete a node in BST
4: Insert a node in BST
5: Exit
Enter your option: 1
Enter the value: 10
If u want to stop enter '0': 1
Enter the value: 15
If u want to stop enter '0': 1
Enter the value: 5
If u want to stop enter '0': 1
Enter the value: 2
If u want to stop enter '0': 1
Enter the value: 7
If u want to stop enter '0': 1
Enter the value: 80
If u want to stop enter '0': 1
Enter the value: 17
If u want to stop enter '0': 0

***** MAIN MENU *****
1: Create a BINARY SEARCH TREE
2: Display the BST in postorder
3: Delete a node in BST
4: Insert a node in BST
```

5: Exit
Enter your option : 2

Post ORDER: 2 7 5 17 20 15 10

***** MAIN MENU *****
1: Create a BINARY SEARCH TREE
2: Display the BST in postorder
3: Delete a node in BST
4: Insert a node in BST
5: Exit
Enter your option: 3
Enter element to delete : 9

***** MAIN MENU *****
1: Create a BINARY SEARCH TREE
2: Display the BST in postorder
3: Delete a node in BST
4: Insert a node in BST
5: Exit
Enter your option: 3
Enter element to delete: 17
***** MAIN MENU *****
1: Create a BINARY SEARCH TREE
2: DISPLAY the BST in postorder
3: Delete a node in BST
4: Insert a node in BST
5: Exit
Enter your option: 2

POST ORDER: 2 7 5 20 15 10

***** MAIN MENU *****

1: Create a BINARY SEARCH TREE

2: Display the BST in postorder

3: Delete a node in BST

4: Insert a node in BST

5: Exit

Enter your option: 4

Enter the value to insert: 9

***** MAIN MENU *****

1: Create a BINARY SEARCH TREE

2: DISPLAY the BST in postorder

3: Delete a node in BST

4: Insert a node in BST

5: Exit

Enter your option: 2

POST ORDER: 2 9 7 5 20 15 10

***** MAIN MENU *****

1: Create a BINARY SEARCH TREE

2: DISPLAY the BST in postorder

3: Delete a node in BST

4: Insert a node in BST

5: Exit

Enter your option: 5

Aim:- To write a C program for implementing depth first traversal and breadth first traversal of a graph;

Source Code:

```
#include<stdio.h>
#define SIZE 10
int G[10][10], visitied[10], n;
int queue[SIZE], rear=-1, front=-1;
void DFS(int i){
    int j;
    printf("%c ", i+65);
    visited[i]=1;
    for(j=0; j<n; j++){
        if(!visited[j] && G[i][j]==1)
            DFS(j);
    }
}
void enque(int x){
    if(rear==SIZE-1)
        printf("Overflow\n");
    else if(rear==-1 && front==-1)
        front=rear=0;
        front queue[rear]=x;
    }else
        queue[++rear]=x;
}
int deque(){
    int data;
    if(rear==-1 && front==-1){
        printf("Underflow\n");
        return -1;
    }
```

```
        else if (front==rear){
                data=queue[front];
                front=rear=-1;
        }
        else
                data=queue[front++];
        return data;
}
void bfs(int n,int adj[][n],int start){
        int j,k;
        enque(start);
        visited[start]=1;
        while(front!=1 && rear!=1){
                k=deque();
                printf("%c ",k+65);
                for(j=0;j<n;j++){
                        if(adj[k][j]==1 && visited[j]==0){
                                enque(j);
                                visited[j]=1;
                        }
                }
        }
}
void bft(int n,int adj[][n]){
        for(int i=0;i<n;i++){
                if(visited[i]==0)
                        bfs(n,adj,i);
        }
}
```

```
void main(){
    int i,j;
    printf("\nEnter number of vertices: ");
    scanf("%od", &n);
    printf("\n Enter the adjacency matrix of the graph:\n");
    for(i=0; i<n; i++){
        for(j=0; j<n; j++)
            scanf("%d", &G[i][j];
    }
    for(i=0; i<n; i++)
        visitied[i]=0;
    printf("\n DFS of the graph: ");
    DFS(0);
    for(i=0; i<n; i++)
        visited[i]=0;
    printf("\n BFS of the graph : ");
    bfs(n, G);
}
```

Output:-

Sample Output:

```
Enter the number of vertices : 6

Enter adjacency matrix of the graph:
1 0 1 0 1 1
0 1 1 0 1 0
1 1 0 0 1 0
0 0 1 1 0 1
1 1 1 0 1 0
1 0 1 1 0 1
DFS of the graph: A C B E F D
BFS of the graph: A C E F D B
```

Sample output - 2 :

Enter number of vertices : 5

Enter the adjacency matrix of the graph:

1 1 0 1 1
0 0 0 0 0
1 1 1 1 1
0 1 0 1 0
1 0 1 0 1

DFS of the graph: A B D E C
BFS of the graph : A B D E C