SCHEDULE

LAB CYCLE-1:

S.NO	NAME OF THE EXPERIMENT
1	Linear Search
2	Binary Search
3	Bubble Sort
4	Selection Sort
5	Insertion Sort
6	Stack
7	Queue
8	Circular Queue
9	Dequeue

LAB CYCLE-2:

S.NO	NAME OF THE EXPERIMENT
10	Stack using Linked List
11	Queue using Linked List
12	Singly Linked List
13	Polynomial Addition
14	Infix to Postfix Evaluation
15	Implementation of Tree
16	Implementation of Graph

Experiment No: 1

LINEAR SEARCH

PROBLEM DEFINITION:

Write a program to implement Linear Search.

ALGORITHM:

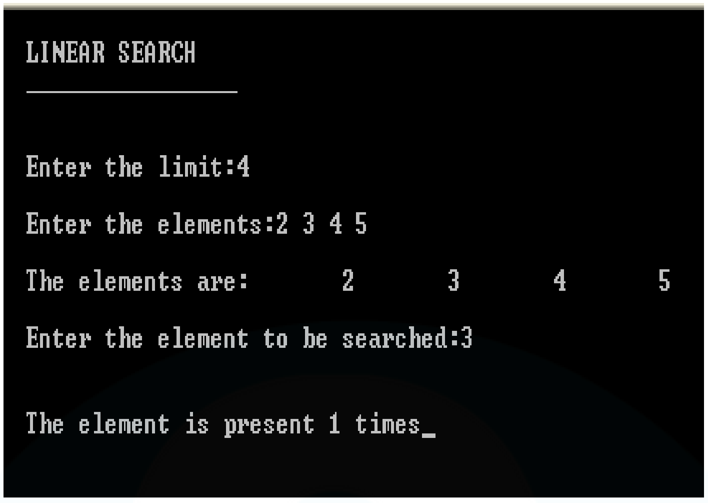
```
Step 1: Start
Step 2: Set count as zero, flag as zero and position as zero
Step 3: Read the limit
Step 4: Read the elements
Step 5: Loop (i<n)
        Step 5.1: Read the elements
        Step 5.2: Incriment i
Step 6: End of loop
Step 7: Loop (i<n)
        Step 7.1: Print the elements
Step 8: Loop ends
Step 9: Printf enter the key elements
Step10: Read the element
Step 11: Loop (i<n)
        Step 11.1: If (a[i]+key)
                Step 11.1.1: Set flag =1
                Step 11.1.2: Set position=1+1
                Step 11.1.3: Increment count
                Step 11.1.4: Print the key found at position
        Step 11.2: If ends
Step 12: Loop ends
Step 13: If (flag=1)
        Step 13.1: Print the key found number of times
Step 14: If ends
Step 15: Else
        Step 15.1: Print element not found
Step 16: Else end
Step 17: Stop
```

PROGRAM DEVELOPMENT:

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

```
#include<conio.h>
void main()
{
 int a[50],i,n,elt,count;
 clrscr();
 printf("\n\tLINEAR SEARCH");
 printf("\n\t_________________\n\n");
 printf("\n\tEnter the limit:");
 scanf("%d",&n);
 printf("\n\tEnter the elements:");
 for(i=0;i<n;i++)
{
scanf("%d",&a[i]);
}
printf("\n\tThe elements are:");
for(i=0;i<n;i++)
{
printf("\t%d",a[i]);
}
printf("\n\n\tEnter the element to be searched:");
scanf("%d",&elt);
count=0;
for(i=0;i<n;i++)
{
if(a[i]==elt)
{
count=count+1;
}
}
printf("\n\n\tThe element is present %d times",count);
getch();
}
```

OUTPUT:



```
LINEAR SEARCH
_____________

Enter the limit:4

Enter the elements:2 3 4 5

The elements are:       2       3       4       5

Enter the element to be searched:3

The element is present 1 times_
```

CONCLUSION:

The algorithm was developed and the program was coded. The program was tested successfully.

Experiment No: 2

BINARY SEARCH

PROBLEM DEFINITION:

Write a program to implement Binary Search.

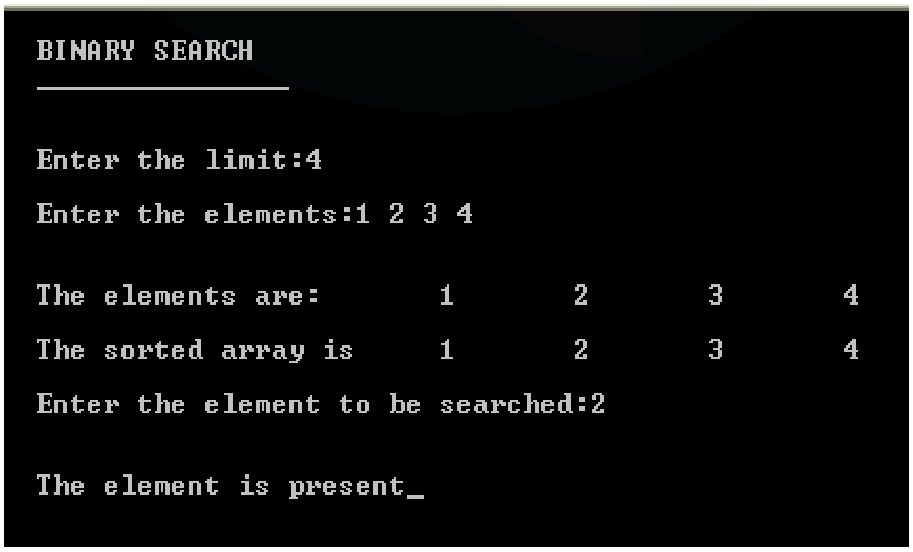
ALGORITHM:

```
Step 1: Start
Step 2: Read the Array size
Step 3: Read the array elements
Step 4: Loop (i<n)
        Step 4.1: Read the array elements
Step 5: Loop ends
Step 6: Loop (i<n)
        Step 6.1: Loop (j<n-1)
        Step 6.2: If (a[i]>a[j+1])
                Step 6.2.1: Set temp as a[i]
                Step 6.2.2: Set a[i] as a[j+1]
                Step 6.2.3: Set a[j+1] as temp
        Step 6.3: Loop ends
Step 7: Loop ends
Step 8: Loop (i<n)
        Step 8.1: Print Sorted array
Step 9: Loop ends
Step10: Set begin as zero
Step11: Set end as n
Step 12: Print enter the key
Step 13: Read key
Step 14: Loop (begin <end)
        Step 14.1: Set mid as (begin+end)/2
        Step 14.2: If (a[mid]==key)
                Step 14.2.1: Print element key found at position mid
        Step 14.3: If ends
        Step 14.4: If (key>a[mid])
                Step 14.4.1: Set begin as mid +1
        Step 14.5: Endif
        Step 14.6: Else
                Step 14.6.1: Set end as mid -1
        Step 14.7: Else end
        Step 14.8: If (begin++ end)
        Step 14.8.1: Print not found
    Step 14.9: If ends
Step 15: Stop
```

PROGRAM DEVELOPMENT:

```
#include<stdio.h>
#include<conio.h>
 void main()
{
int a[50],i,j,n,elt,temp,flag=0,low=0,high,mid;
clrscr();
printf("\n\tBINARY SEARCH");
printf("\n\t________________\n\n");
printf("\n\tEnter the limit:");
scanf("%d",&n);
printf("\n\tEnter the elements:");
for(i=0;i<n;i++)
{
scanf("%d",&a[i]);
}
printf("\n\n\tThe elements are:");
for(i=0;i<n;i++)
{
printf("\t%d",a[i]);
}
for(i=0;i<n-1;i++)
{
for(j=0;j<n-1-i;j++)
{
if(a[j]>a[j+1])
{
temp=a[j];
a[j]=a[j+1];
a[j+1]=temp;
}
}
}
printf("\n\n\tThe sorted array is");
for(i=0;i<n;i++)
{
printf("\t%d",a[i]);
}
printf("\n\n\tEnter the element to be searched:");
scanf("%d",&elt);
high=n-1;
while(low<=high)
{
mid=(low+high)/2;
if(elt<a[mid])
{
high=mid-1;
}
else if(elt>a[mid])
{
low=mid+1;
}
else
{
printf("\n\n\tThe element is present");
flag=1;
break;
}
}
if(flag==0)
{
printf("\n\n\tThe element is not present");
}
 getch();
}
```

OUTPUT:



```
BINARY SEARCH
_______________

Enter the limit:4

Enter the elements:1 2 3 4

The elements are:       1       2       3       4

The sorted array is     1       2       3       4

Enter the element to be searched:2

The element is present_
```

CONCLUSION:

The algorithm was developed and the program was coded. The program was tested successfully.

Experiment No: 3

BUBBLE SORT

PROBLEM DEFINITION:

Write a program to implement Bubble Sort.

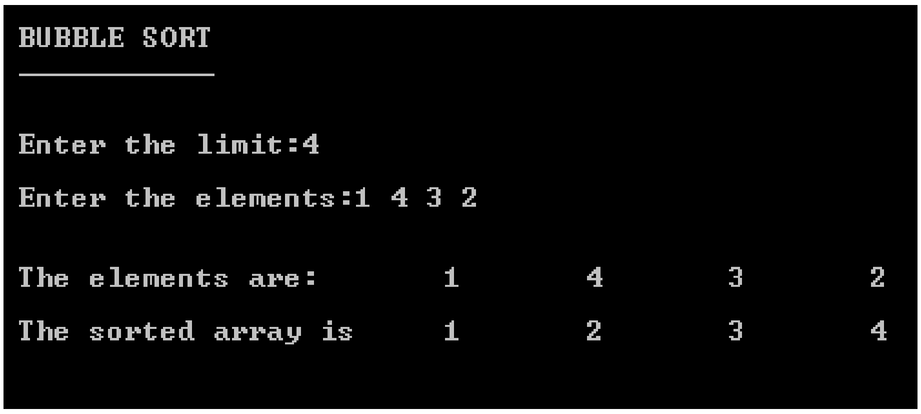
ALGORITHM:

```
Step 1: Start
Step 2: Read the size of arry
Step 3: Read the elements
Step 4: Loop (i<n)
        Step 4.1: Read the elements
Step 5: Loop ends
Step 6: Loop (i<n)
        Step 6.1: Loop (j<n-1)
                Step 6.1.1: If (a[i]>a[j+1])
                Step 6.1.2: Set temp as a[i]
                Step 6.1.3: Set a[j] as a[j+1]
                Step 6.1.4: Set a[j+1] as temp
        Step 6.2: If ends
        Step 6.3: Loop ends
Step 7: Loop ends
Step 8: Print sorted array
Step 9: Loop (i<n)
        Step 9.1: Print array
Step10: Loop ends
Step11: Stop
```

PROGRAM DEVELOPMENT:

```
#include<stdio.h>
#include<conio.h>
 void main()
{
int a[50],i,j,n,temp;
clrscr();
printf("\n\tBUBBLE SORT");
printf("\n\t___________\n\n");
printf("\n\tEnter the limit:");
scanf("%d",&n);
printf("\n\tEnter the elements:");
for(i=0;i<n;i++)
{
scanf("%d",&a[i]);
}
printf("\n\n\tThe elements are:");
for(i=0;i<n;i++)
{
printf("\t%d",a[i]);
}
for(i=0;i<n-1;i++)
{
for(j=0;j<n-1-i;j++)
{
if(a[j]>a[j+1])
{
temp=a[j];
a[j]=a[j+1];
a[j+1]=temp;
}
}
}
printf("\n\n\tThe sorted array is");
for(i=0;i<n;i++)
{
printf("\t%d",a[i]);
}
getch();
}
```

OUTPUT:



```
BUBBLE SORT
___________

Enter the limit:4

Enter the elements:1 4 3 2

The elements are:       1       4       3       2

The sorted array is     1       2       3       4
```

CONCLUSION:

The algorithm was developed and the program was coded. The program was tested successfully.

Experiment No: 4

SELECTION SORT

PROBLEM DEFINITION:

Write a program to implement Selection Sort.

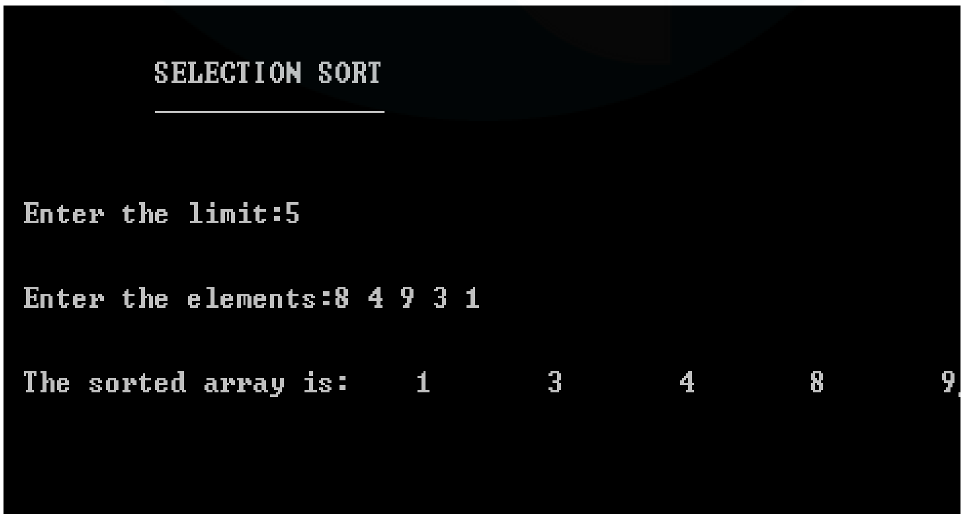
ALGORITHM:

```
Step 1: Start
Step 2: Read the size of the array
Step 3: Read the elements
Step 4: Set I to 0
Step 5: Loop (i<n)
        Step 5.1: Read elements
        Step 5.2: Increment i
Step 6: Loop ends
Step 7: Loop (i<n)
        Step 7.1: Set j=i+1
        Step 7.2: Loop (j<n)
                Step 7.2.1: If (a[j]>a[i])
                        Step 7.2.1.1: Set temp=a[i]
                        Step 7.2.1.2: Set a[i]=a[j]
                        Step 7.2.1.3: Set a[j]=temp
                Step 7.2.2: Endif
                Step 7.2.3: Loop ends
Step 8: Loop ends
Step 9: Loop (i<n)
        Step 9.1: Print the sorted array
Step 10: Loop ends
Step 11: Stop
```

PROGRAM DEVELOPMENT:

```
#include<stdio.h>
#include<conio.h>
void main()
{
int a[50],i,n,j,temp;
clrscr();
printf("\n\n\t\tSELECTION SORT");
printf("\n\t\t______________\n\n");
printf("\n\n\tEnter the limit:");
scanf("%d",&n);
printf("\n\n\tEnter the elements:");
for(i=0;i<n;i++)
{
scanf("%d",&a[i]);
}
for(i=0;i<n-1;i++)
{
for(j=i+1;j<n;j++)
{
if(a[i]>a[j])
{
temp=a[i];
a[i]=a[j];
a[j]=temp;
}
}
}
printf("\n\n\tThe sorted array is:");
for(i=0;i<n;i++)
{
printf("\t%d",a[i]);
}
getch();
}
```

OUTPUT:



```
        SELECTION SORT
        ______________


Enter the limit:5

Enter the elements:8 4 9 3 1

The sorted array is:    1       3       4       8       9
```

CONCLUSION:

The algorithm was developed and the program was coded. The program was tested successfully.

Experiment No: 5

INSERTION SORT

PROBLEM DEFINITION:

Write a program to implement Insertion Sort.

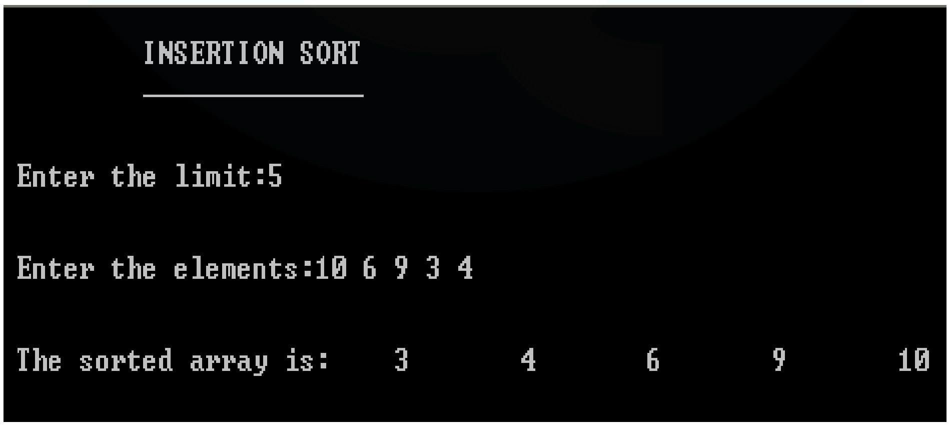
ALGORITHM:

```
Step 1: Start
Step 2: Read the array size
Step 3: Read the elements
Step 4: Loop (i<n)
        Step 4.1: Read array elements
Step 5: Loop ends
Step 6: Set I to 1
Step 7: Loop (i<n)
        Step 7.1: Set temp as a[i]
        Step 7.2: Set j as i-1
        Step 7.3: Loop (a[i]>temp & j>=0)
                Step 7.3.1: Set a[j+1]=a[j]
                Step 7.3.2: Decrement j
        Step 7.4: End loop
        Step 7.5: Set a[j+1] as temp
Step 8: End loop
Step 9: Loop (i<n)
        Step 9.1: Print the sorted array
Step10: Loop ends
Step 11: Stop
```

PROGRAM DEVELOPMENT:

```
#include<stdio.h>
#include<conio.h>
void main()
{
int a[50],i,n,j,temp;
clrscr();
printf("\n\t\tINSERTION SORT\n");
printf("\t\t_____________\n");
printf("\n\n\tEnter the limit:");
scanf("%d",&n);
printf("\n\n\tEnter the elements:");
for(i=0;i<n;i++)
{
scanf("%d",&a[i]);
}
for(i=1;i<n;i++)
{
temp=a[i];
j=i-1;
while(temp<a[j]&&j>=0)
{
a[j+1]=a[j];
 j--;
 }
a[j+1]=temp;
}
printf("\n\n\tThe sorted array is:");
for(i=0;i<n;i++)
{
printf("\t%d",a[i]);
}
getch();
}
```

OUTPUT:



```
        INSERTION SORT
        ______________

Enter the limit:5

Enter the elements:10 6 9 3 4

The sorted array is:    3       4       6       9       10
```

CONCLUSION:

The algorithm was developed and the program was coded. The program was tested successfully.

Experiment No: 6

STACK

PROBLEM DEFINITION:

Write a program to implement Stack.

ALGORITHM:

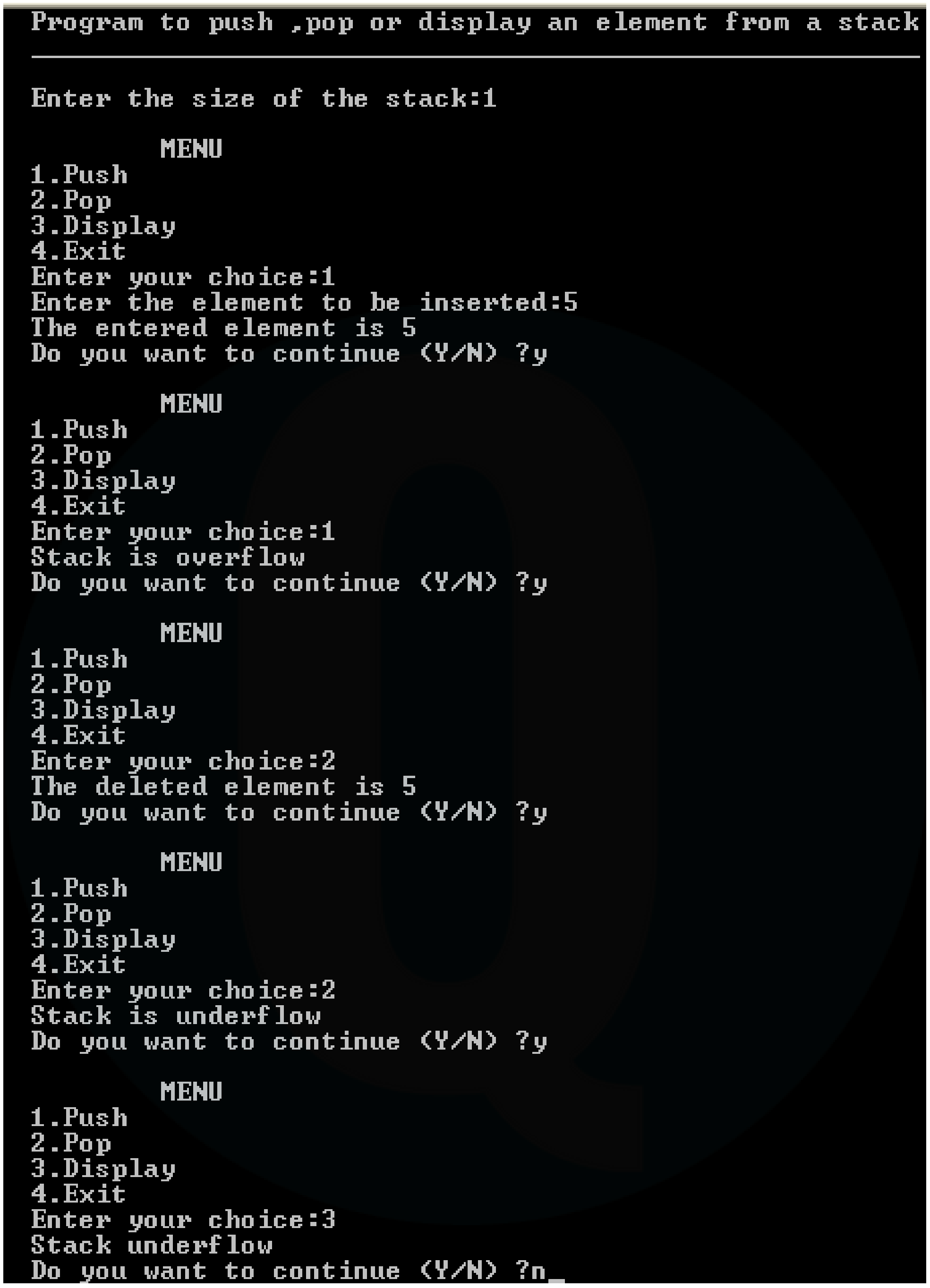
```
Step 1: Start
Step 2: Set top to -1
Step 3: Print size of stack
Step 4: If (push)
    Step 4.1: If (top=size-1)
        Step 4.1.1: Print overflow
        Step 4.1.2: Endif
        Step 4.1.3: Else
            Step 4.1.3.1: Decrement top
            Step 4.1.3.2: Read one element
            Step 4.1.3.3: Set stack[top] to element
        Step 4.1.4: Else end
Step 5: If (pop)
    Step 5.1: If (top=-1)
        Step 5.1.1: Print overflow
        Step 5.1.2: Endif
        Step 5.1.3: Else
            Step 5.1.3.1: Set item to stack[top]
            Step 5.1.3.2: Set top to top-1
            Step 5.1.3.3: Else end
Step 6: If (display)
    Step 6.1: Set i as top
    Step 6.2: Loop (i<top)
        Step 6.2.1: Print stack
        Step 6.2.2: Increment i
    Step 6.3: Loop ends
Step 7: Stop
```

PROGRAM DEVELOPMENT:

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

```
#include<conio.h>
void main()
{
int st[50],top=-1,k,n,i,si;
char ch;
clrscr();
printf("\n\tProgram to push ,pop or display an element from a stack");
printf("\n\t_____________________________________________________________\n\n");
printf("\n\tEnter the size of the stack:") ;
scanf("%d",&si);
do
{
printf("\n\t\tMENU");
printf("\n\t\t_____\n");
printf("\n\n\t1.Push\n\n\t2.Pop\n\n\t3.Display\n\n\t4.Exit");
printf("\n\n\tEnter your choice:");
scanf("%d",&n);
if(n==1)
{
if(top==si-1)
{
printf("\n\tStack is overflow");
else
{
printf("\n\tEnter the element to be inserted:");
scanf("%d",&k);
top++;
st[top]=k;
printf("\n\tThe entered element is %d",st[top]);
}
}
else if(n==2)
{
if(top==-1)
{
printf("\n\tStack is underflow");
}
else
{
printf("\n\tThe deleted element is %d",st[top]);
top--;
}
}
else if(n==3)
{
if(top==-1)
{
printf("\n\tStack underflow");
}
else
{
printf("\n\tThe stack is:");
for(i=top;i>=0;i--)
{
printf("\t\t%d",st[i]);
}
}
}
else
{
break;
}
printf("\n\n\tDo you want to continue (Y/N) ?");
scanf(" %c",&ch);
}
while((ch=='Y')||(ch=='y'));
getch();
}
```

OUTPUT:



```
Program to push ,pop or display an element from a stack
________________________________________________________

Enter the size of the stack:1

        MENU
1.Push
2.Pop
3.Display
4.Exit
Enter your choice:1
Enter the element to be inserted:5
The entered element is 5
Do you want to continue (Y/N) ?y

        MENU
1.Push
2.Pop
3.Display
4.Exit
Enter your choice:1
Stack is overflow
Do you want to continue (Y/N) ?y

        MENU
1.Push
2.Pop
3.Display
4.Exit
Enter your choice:2
The deleted element is 5
Do you want to continue (Y/N) ?y

        MENU
1.Push
2.Pop
3.Display
4.Exit
Enter your choice:2
Stack is underflow
Do you want to continue (Y/N) ?y

        MENU
1.Push
2.Pop
3.Display
4.Exit
Enter your choice:3
Stack underflow
Do you want to continue (Y/N) ?n_
```

CONCLUSION:

The algorithm was developed and the program was coded. The program was tested successfully.

Experiment No: 7

QUEUE

PROBLEM DEFINITION:

Write a program to implement Queue.

ALGORITHM:

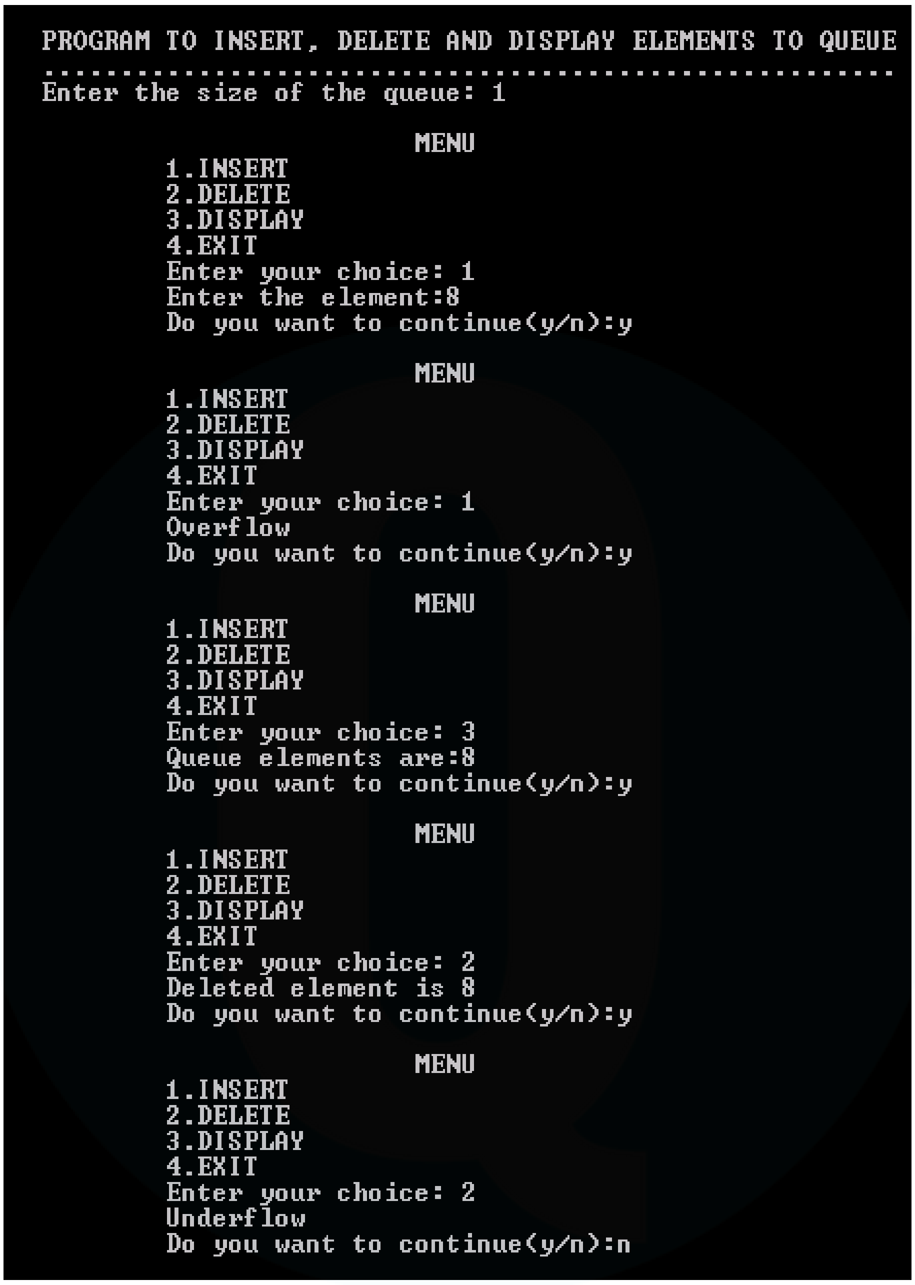
```
Step 1: Start
Step 2: Set front and rear to -1
Step 3: Read size of queue
Step 4: If (insertion)
        Step 4.1: If (rear=size-1)
                Step 4.1.1: Print overflow
        Step 4.2: End if
        Step 4.3: Else
                Step 4.3.1: Print enter the element
                Step 4.3.2: Read the element
                Step 4.3.3: If (front=-1& rear=-1)
                        Step 4.3.3.1: Set front=0
                        Step 4.3.3.2: Increment rear
                        Step 4.3.3.3: Read element to queue [rear]
                        Step 4.3.3.4: Endif
                Step 4.3.4: End else
Step 5: If (deletion)
        Step 5.1: If (front=-1)
                Step 5.1.1: Print underflow
        Step 5.2: End if
        Step 5.3: Else
                Step 5.3.1: Print deleted element is queue [front]
                Step 5.3.2: If (front=rear)
                        Step 5.3.2.1: Set front and rear as -1
                        Step 5.3.2.2: Endif
                        Step 5.3.2.3: Else
                                Step 5.3.2.3.1: Increment front
                                Step 5.3.2.3.2: End Else
        Step 5.4: Endif
Step 6: If (display)
        Step 6.1: If (front & rear are -1)
                Step 6.1.1: Print underflow
                Step 6.1.2: End if
```

```
            Step 6.1.3: Else
                  Step 6.1.3.1: Set i as zero
                  Step 6.1.3.2: Loop (i<rear)
                        Step 6.1.3.2.1: Print queue[i]
                        Step 6.1.3.2.2: Loop ends
            Step 6.1.4: End else
Step 7: End if
Step 8: Stop
```

PROGRAM DEVELOPMENT:

```
#include<stdio.h>
#include<conio.h>
void main()
{
int no,i,front=-1,rear=-1,k,queue[50],element,n;
char c;
clrscr();
printf("\n\n\n\tPROGRAM TO INSERT, DELETE AND DISPLAY ELEMENTS TO QUEUE");
printf("\n\t......................................................\n\n\n");
printf("\n\n\tEnter the size of the queue: ");
scanf("%d",&n);
do
{
printf("\n\n\n\t\t\t\tMENU\n\n");
printf("\t\t1.INSERT\n\n\t\t2.DELETE\n\n\t\t3.DISPLAY\n\n\t\t4.EXIT\n\n\t\tEnter your choice: ");
scanf("%d",&no);
if(no==1)
{
if(rear==(n-1))
printf("\n\t\tOverflow");
else
{
printf("\n\n\t\tEnter the element: ");
scanf("%d",&element);
if(front==-1&&rear==-1)
front=0;
rear++;
queue[rear]=element;
}
}
if(no==2)
{
if(front==-1)
printf("\n\t\tUnderflow\n");
else
{
k=queue[front];
printf("\n\t\tDeleted element is %d ",k);
}
if(front==rear)
front=rear=-1;
else
front++;
}
if(no==3)
{
if(front==-1&&rear==-1)
printf("\n\t\tUnderflow\n");
else
{
printf("\n\t\tQueue elements are\n ");
for(i=front;i<=rear;i++)
{
printf("\n\n\t\t%d",queue[i]);
}
}
}
if(no==4)
break;
printf("\n\n\t\tDo you want to continue(y/n) ");
scanf(" %c",&c);
}
while(c=='y'||c=='Y');
getch();
}
```

OUTPUT:



```
PROGRAM TO INSERT, DELETE AND DISPLAY ELEMENTS TO QUEUE
.......................................................
Enter the size of the queue: 1

                        MENU
        1.INSERT
        2.DELETE
        3.DISPLAY
        4.EXIT
        Enter your choice: 1
        Enter the element:8
        Do you want to continue(y/n):y

                        MENU
        1.INSERT
        2.DELETE
        3.DISPLAY
        4.EXIT
        Enter your choice: 1
        Overflow
        Do you want to continue(y/n):y

                        MENU
        1.INSERT
        2.DELETE
        3.DISPLAY
        4.EXIT
        Enter your choice: 3
        Queue elements are:8
        Do you want to continue(y/n):y

                        MENU
        1.INSERT
        2.DELETE
        3.DISPLAY
        4.EXIT
        Enter your choice: 2
        Deleted element is 8
        Do you want to continue(y/n):y

                        MENU
        1.INSERT
        2.DELETE
        3.DISPLAY
        4.EXIT
        Enter your choice: 2
        Underflow
        Do you want to continue(y/n):n
```

CONCLUSION:

The algorithm was developed and the program was coded. The program was tested successfully.

Experiment No: 8

CIRCULAR QUEUE

PROBLEM DEFINITION:

Write a program to implement Circular Queue.

ALGORITHM:

```
Step 1: Start
Step 2: Set front=0&rear=0
Step 3: Read size of queue
Step 4: If (insertion)
        Step 4.1: Read the element
        Step 4.2: If (front and rear are zero)
                Step 4.2.1: Set front&rear to 1
                Step 4.2.2: Read element to queue [rear]
        Step 4.3: Endif
        Step 4.4: Else
                Step 4.4.1: Set next as rear mod size+1
                Step 4.4.2: If (next=front)
                        Step 4.4.2.1: Print overflow
                        Step 4.4.2.2: End if
                Step 4.4.3: Else
                        Step 4.4.3.1: Set rear as next
                        Step 4.4.3.2: Read element to queue [rear]
                        Step 4.4.3.3: End else
                Step 4.4.5: End else
Step 5: End if
Step 6: If (deletion)
        Step 6.1: If (front=0)
        Step 6.2: Print underflow
        Step 6.3: Endif
        Step 6.4: Else
                Step 6.4.1: Print deleted element as queue [front]
                Step 6.4.2: If (front =rear)
                Step 6.4.3: Set front and rear as 0
                Step 6.4.4: Endif
        Step 6.5: Else
                Step 6.5.1: Set front as front mmode size+1
                Step 6.5.2: End else
                Step 6.5.3: End else
Step 7: End if
Step 8: If (display)
        Step 8.1: If (front & rear =0)
                Step 8.1.1: (Print under flow)
        Step 8.2: End if
        Step 8.3: Else
                Step 8.3.1: If (front <rear)
                        Step 8.3.1.1: Set i as front
                        Step 8.3.1.2: Loop (i<rear)
                                        Step 8.3.1.2.1: Print queue[i]
                                        Step 8.3.1.2.2: Increment i
                        Step 8.3.1.3: Loop ends
                Step 8.3.2: Endif
                Step 8.3.3: Else
                        Step 8.3.3.1: Loop (i<rear)
                                Step 8.3.3.1.1: Print queue[i]
                                Step 8.3.3.1.2: Increment i
                        Step 8.3.3.2: Loop ends
                        Step 8.3.3.3: Set I as 1
                        Step 8.3.3.4: Loop (i<rear)
                                Step 8.3.3.4.1: Print queue[i]
                                Step 8.3.3.4.2: Increment i
                        Step 8.3.3.5: Loop ends
                Step 8.3.4: Else end
Step 9: Endif
Stop 10: Stop
```

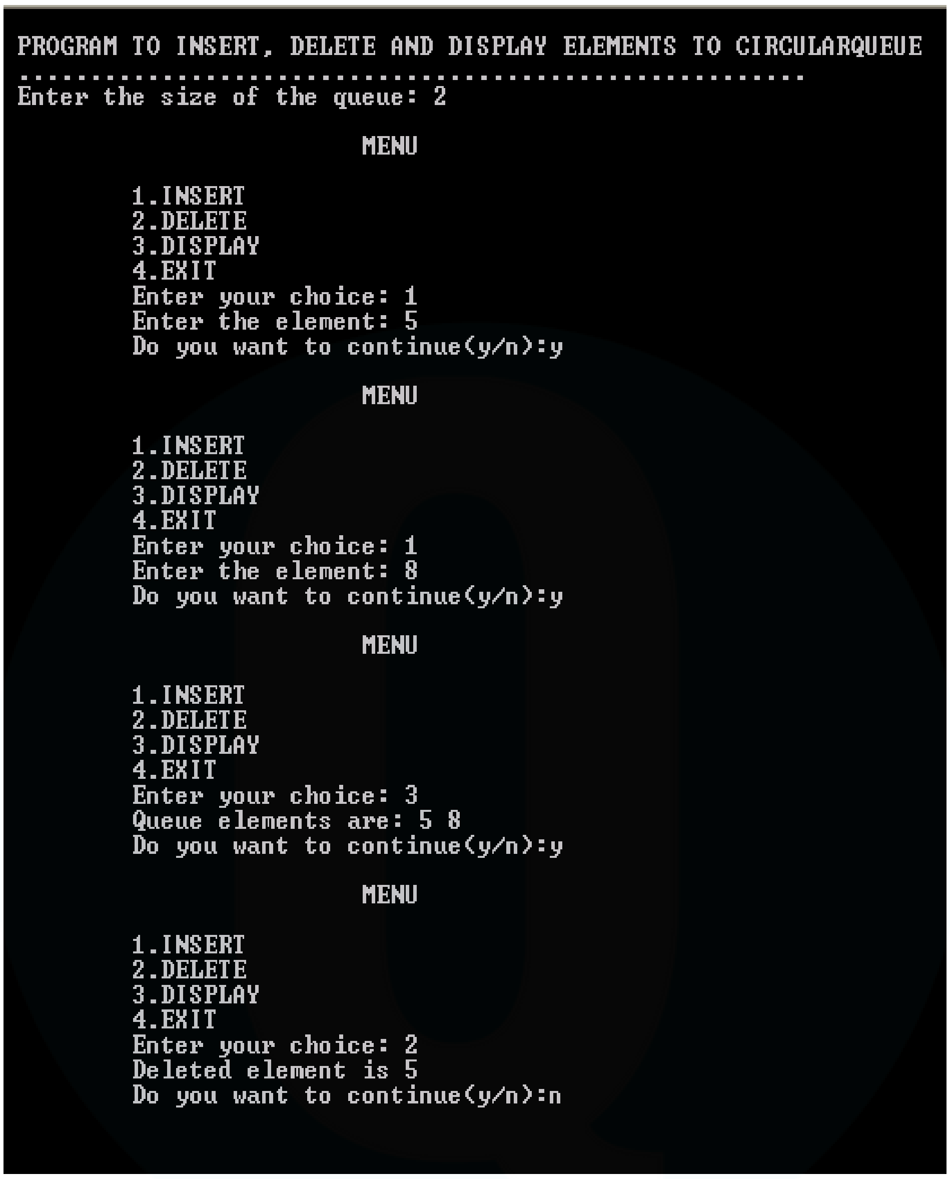
PROGRAM DEVELOPMENT:

```
#include<stdio.h>
#include<conio.h>
void main()
{
int no,i,front=0,rear=0,k,queue[50],element,size,next=1;
char c;
clrscr();
printf("\n\tPROGRAM TO INSERT, DELETE AND DISPLAY ELEMENTS TO
CIRCULARQUEUE");
printf("\n\t.....................................................");
printf("\n\tEnter the size of the queue: ");
scanf("%d",&size);
do
{
```

```
printf("\n\t\t\t\tMENU\n\n");
printf("\t\t1.INSERT\n\t\t2.DELETE\n\t\t3.DISPLAY\n\t\t4.EXIT\n\t\tEnter your choice: ");
scanf("%d",&no);
if(no==1)
{
if(rear==size-1)
{
printf("\n overflow ");
}
printf("\t\tEnter the element: ");
scanf("%d",&element);
if(front==0&&rear==0)
{
front=rear=1;
queue[rear]=element;
}
else
{
next=(rear%size)+1;
if(next==front)
printf("\t\tOverflow\n");
else
{
rear=next;
queue[rear]=element;
}
}
}
if(no==2)
{
if(front==0)
printf("\t\tUnderflow\n");
else
{
k=queue[front];
printf("\t\tDeleted element is %d\n",k);
}
if(front==rear)
front=rear=0;
else
front=(front%size)+1;
}
if(no==3)
{
```

```
if(front==0&&rear==0)
printf("\t\tUnderflow\n");
else
{
printf("\t\tQueue elements are");
if(front<rear)
{
for(i=front;i<=rear;i++)
printf("\n\t%d\t",queue[i]);
}
else
{
for(i=front;i<=size;i++)
printf("\n\t\t%d",queue[i]);
for(i=1;i<=rear;i++)
printf("\n\t\t%d",queue[i]);
}
}
}
if(no==4)
break;
printf("\t\tDo you want to continue(y/n):");
scanf(" %c",&c);
}
while(c=='y'||c=='Y');
getch();
}
```

OUTPUT:



```
PROGRAM TO INSERT, DELETE AND DISPLAY ELEMENTS TO CIRCULARQUEUE
.......................................................
Enter the size of the queue: 2

                         MENU

        1.INSERT
        2.DELETE
        3.DISPLAY
        4.EXIT
        Enter your choice: 1
        Enter the element: 5
        Do you want to continue(y/n):y

                         MENU

        1.INSERT
        2.DELETE
        3.DISPLAY
        4.EXIT
        Enter your choice: 1
        Enter the element: 8
        Do you want to continue(y/n):y

                         MENU

        1.INSERT
        2.DELETE
        3.DISPLAY
        4.EXIT
        Enter your choice: 3
        Queue elements are: 5 8
        Do you want to continue(y/n):y

                         MENU

        1.INSERT
        2.DELETE
        3.DISPLAY
        4.EXIT
        Enter your choice: 2
        Deleted element is 5
        Do you want to continue(y/n):n
```

CONCLUSION:

The algorithm was developed and the program was coded. The program was tested successfully.

Experiment No: 9

DEQUEUE

PROBLEM DEFINITION:

Write a program to implement Dequeue.

ALGORITHM:

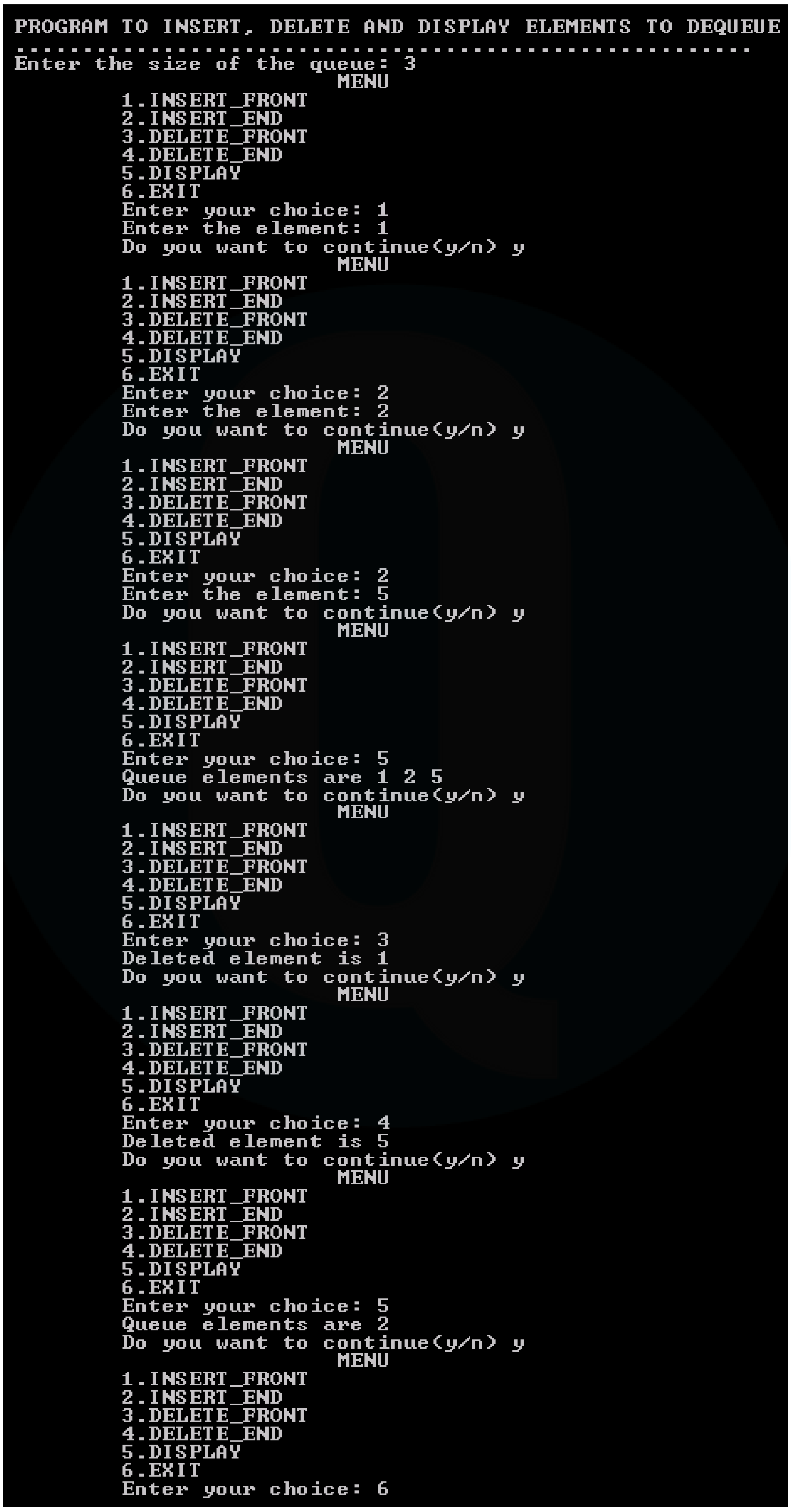
```
Step 1: Start
Step 2: Read choice
Step 3: If (insert front)
        Step 3.1: If front =0
                Step 3.1.1: Print overflow
        Step 3.2: Else
                Step 3.2.1: Read element
        Step 3.3: End if
        Step 3.4: If (front= -1)
                Step 3.4.1: Set front =rear=0
                Step 3.4.2: Insert element
        Step 3.5: Else if (front!=0)
                Step 3 5.1: Set front= front-1
                Step 3.5.2: Insert element
        Step 3.6: End if
Step 4: If (insert end)
        Step 4.1: If (rear= n-1)
                Step 4.1.1: Print ‘overflow’
        Step 4.2: Else
                Step 4.2.1: Read element
                Step 4.2.2: If (front =-1)
                        Step 4.2.2.1: Set front=rear=0
                Step 4.2.3: Else
                        Step 4.2.3.1: Rear+ +
                Step 4.2.4: End if
                Step 4.2.5: Insert element
        Step 4.3: End if
Step 5: If (delete front)
        Step 5.1: If (f= -1)
                Step 5.1.1: Print overflow
        Step 5.2: Else
                Step 5.2.1: If (f=rear)
                Step 5.2.2: F=rear=-1
        Step 5.3: End if
Step 6: If (insertion at end)
        Step 6.1: If (f=-1)
                Step 6.1.1: Print under flow
        Step 6.2: Else
                Step 6.2.1: If (f=rear)
                        Step 6.2.1.1: F=rear=-1
                Step 6.2.2: End if
                Step 6.2.3: Rrear =rear-1
        Step 6.3: End if
Step 7: Stop.
```

PROGRAM DEVELOPMENT:

```
#include<stdio.h>
#include<conio.h>
void main()
{
int no,i,front=-1,rear=-1,k,queue[50],element,n;
char c;
clrscr();
printf("\n\tPROGRAM TO INSERT, DELETE AND DISPLAY ELEMENTS TO
DEQUEUE\n");
printf("\t.....................................................");
printf("\n\tEnter the size of the queue: ");
scanf("%d",&n);
do
{
printf("\t\t\t\tMENU\n");
printf("\t\t1.INSERT_FRONT\n\t\t2.INSERT_END\n\t\t3.DELETE_FRONT\n\t\t4.DELETE_E
ND\n\t\t5.DISPLAY\n\t\t6.EXIT\n\t\tEnter your choice: ");
scanf("%d",&no);
if(no==1)
{
if(front==0)
printf("\t\tOverflow\n");
else
{
printf("\t\tEnter the element: ");
scanf("%d",&element);
if(front==-1)
{front=0;
rear=0;
queue[front]=element;
}
else if(front!=0)
{
front=front-1;
queue[front]=element;
}
}
}
if(no==2)
{
if(rear==n-1)
printf("\t\tOverflow\n");
else
{
printf("\t\tEnter the element: ");
scanf("%d",&element);
if(front==-1)
{
front=rear=0;
queue[rear]=element;
}
else
{
rear=rear+1;
queue[rear]=element;
}
}
}
if(no==3)
{
if(front==-1)
printf("\t\tUnderflow\n");
else
{
k=queue[front];
printf("\t\tDeleted element is %d ",k);
printf("\n");
if(front==rear)
front=rear=-1;
else
front++;
}
}
```

```
if(no==4)
{
if(rear==-1)
printf("\t\tUnderflow\n");
else
{
k=queue[rear];
printf("\t\tDeleted element is %d ",k);
printf("\n");
if(front==rear)
front=rear=-1;
else
rear--;
}
}
if(no==5)
{
if(front==-1&&rear==-1)
printf("\t\tUnderflow\n");
else
{
printf("\t\tQueue elements are ");
for(i=front;i<=rear;i++)
printf("%d ",queue[i]);
printf("\n");
}
}
if(no==6)
break;
printf("\t\tDo you want to continue(y/n) ");
scanf(" %c",&c);
}
while(c=='y'||c=='Y');
getch();
}
```

OUTPUT:



```
PROGRAM TO INSERT, DELETE AND DISPLAY ELEMENTS TO DEQUEUE
-------------------------------------------------------
Enter the size of the queue: 3
                        MENU
        1.INSERT_FRONT
        2.INSERT_END
        3.DELETE_FRONT
        4.DELETE_END
        5.DISPLAY
        6.EXIT
        Enter your choice: 1
        Enter the element: 1
        Do you want to continue(y/n) y
                        MENU
        1.INSERT_FRONT
        2.INSERT_END
        3.DELETE_FRONT
        4.DELETE_END
        5.DISPLAY
        6.EXIT
        Enter your choice: 2
        Enter the element: 2
        Do you want to continue(y/n) y
                        MENU
        1.INSERT_FRONT
        2.INSERT_END
        3.DELETE_FRONT
        4.DELETE_END
        5.DISPLAY
        6.EXIT
        Enter your choice: 2
        Enter the element: 5
        Do you want to continue(y/n) y
                        MENU
        1.INSERT_FRONT
        2.INSERT_END
        3.DELETE_FRONT
        4.DELETE_END
        5.DISPLAY
        6.EXIT
        Enter your choice: 5
        Queue elements are 1 2 5
        Do you want to continue(y/n) y
                        MENU
        1.INSERT_FRONT
        2.INSERT_END
        3.DELETE_FRONT
        4.DELETE_END
        5.DISPLAY
        6.EXIT
        Enter your choice: 3
        Deleted element is 1
        Do you want to continue(y/n) y
                        MENU
        1.INSERT_FRONT
        2.INSERT_END
        3.DELETE_FRONT
        4.DELETE_END
        5.DISPLAY
        6.EXIT
        Enter your choice: 4
        Deleted element is 5
        Do you want to continue(y/n) y
                        MENU
        1.INSERT_FRONT
        2.INSERT_END
        3.DELETE_FRONT
        4.DELETE_END
        5.DISPLAY
        6.EXIT
        Enter your choice: 5
        Queue elements are 2
        Do you want to continue(y/n) y
                        MENU
        1.INSERT_FRONT
        2.INSERT_END
        3.DELETE_FRONT
        4.DELETE_END
        5.DISPLAY
        6.EXIT
        Enter your choice: 6
```

CONCLUSION:

The algorithm was developed and the program was coded. The program was tested successfully.

Experiment No: 10

STACK USING LINKED LIST

PROBLEM DEFINITION:

Write a program to implement Stack using Linked List.

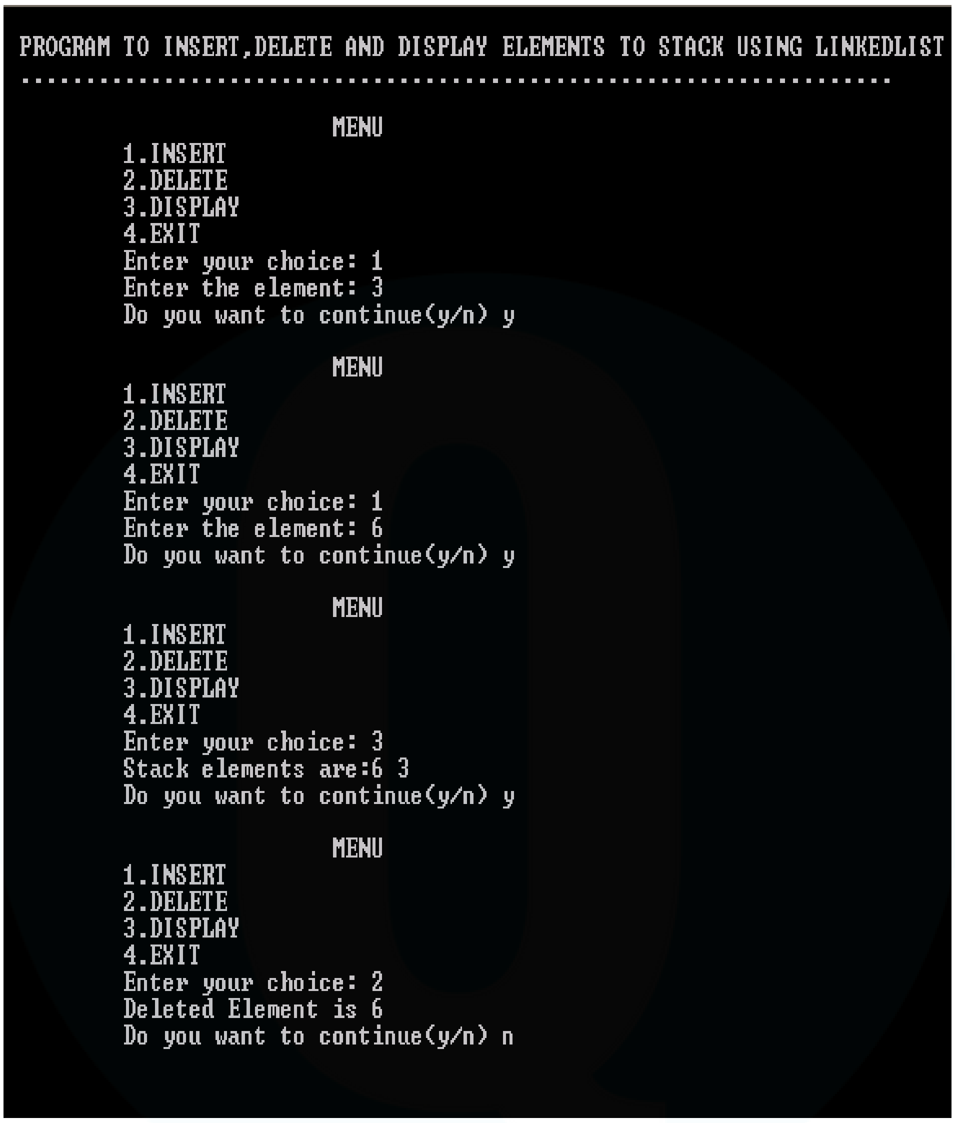
ALGORITHM:

```
Step 1: Start
Step 2: Read the operation
Step 3: If (insertion)
        Step 3.1: New = Getnode (Node)
                Step 3.1.1: If (new=null) then
                        Step 3.1.1.1: Print insufficient memory"
                        Step 3.1.1.2: Exit
Step 4: Else
        Step 4.1: Ptr=header
        Step 4.2: While (ptr->link#Null) do
                Step 4.2.1: Ptr=ptr->link
                Step 4.2.2: End while
                Step 4.2.3: Ptr->link=new
                Step 4.2.4: New->data=x
                Step 4.2.5: Endif
                Step 4.2.6: Endif
Step 5: If (Deletion)
        Step 5.1: Ptr=header
        Step 5.2: If (ptr->link==Null) then
                Step 5.2.1: Printf'empty list"
                Step 5.2.2.: Exit
        Step 5.3: Else
                Step 5.3.1: While (ptr->link#null)do
                Step 5.3.2:Ptr1=ptr
                Step 5.3.3: Ptr=ptr->link
                Step 5.3.4: Endwhile
                Step 5.3.5:Ptr1->link=null
                Step 5.3.6: Return node (ptr)
                Step 5.3.7: Endif
Step 6: Endif
Step 7: If (display)
        Step 7.1: Ptr=header->link
        Step 7.2: While (ptr#Null)do
        Step 7.3: Display (ptr->data)
        Step 7.4: Ptr=ptr->link
        Step 7.5: Endwhile
Step 8: Endif
Step 9: Stop
```

PROGRAM DEVELOPMENT:

```
#include<stdio.h>
#include<conio.h>
#include<alloc.h>
typedef struct node
{
int data;
struct node *link;
}NODE;
NODE *start=NULL,*top=NULL,*s,*ptr,*disply;
void main()
{
int no,item;
char c;
clrscr();
printf("\n\tPROGRAM TO INSERT, DELETE AND DISPLAY ELEMENTS TO STACK
USING LINKED LIST");
printf("\n\t.....................................................\n");
do
{
printf("\t\t\t\tMENU\n");
printf("\t\t1.INSERT\n\t\t2.DELETE\n\t\t3.DISPLAY\n\t\t4.EXIT\n\t\tEnter your choice: ");
scanf("%d",&no);
if(no==1)
{
ptr=(NODE*)malloc(sizeof(NODE));
printf("\n\t\tEnter the element: ");
scanf("%d",&item);

ptr->data=item;
ptr->link=NULL;
if(start==NULL)
{
start=ptr;
top=ptr;
}
else
{
ptr->link=top;
top=ptr;
}
}
if(no==2)
{
if(top==NULL)
printf("\n\t\tStack is empty");
else
{
s=top;
printf("\n\t\tDeleted Element is %d",top->data);
top=top->link;
free(s);
if(top==NULL)
start=NULL;
}
}
if(no==3)
{
if(top==NULL)
printf("\n\t\tStack is empty");
else
{
printf("\n\t\tStack elements are:");
disply=top;
while(disply!=NULL)
{
item=disply->data;
printf("\t %d\n",item);
disply=disply->link;
}
}
}
if(no==4)
break;
printf("\n\t\tDo you want to continue(y/n) ");
scanf(" %c",&c);
}
while(c=='y'||c=='Y');
getch();
}
```

OUTPUT:



```
PROGRAM TO INSERT,DELETE AND DISPLAY ELEMENTS TO STACK USING LINKEDLIST
..................................................................

                        MENU
       1.INSERT
       2.DELETE
       3.DISPLAY
       4.EXIT
       Enter your choice: 1
       Enter the element: 3
       Do you want to continue(y/n) y

                        MENU
       1.INSERT
       2.DELETE
       3.DISPLAY
       4.EXIT
       Enter your choice: 1
       Enter the element: 6
       Do you want to continue(y/n) y

                        MENU
       1.INSERT
       2.DELETE
       3.DISPLAY
       4.EXIT
       Enter your choice: 3
       Stack elements are:6 3
       Do you want to continue(y/n) y

                        MENU
       1.INSERT
       2.DELETE
       3.DISPLAY
       4.EXIT
       Enter your choice: 2
       Deleted Element is 6
       Do you want to continue(y/n) n
```

CONCLUSION:

The algorithm was developed and the program was coded. The program was tested successfully.

Experiment No: 11

QUEUE USING LINKED LIST

PROBLEM DEFINITION:

Write a program to implement Queue using Linked List.

ALGORITHM:

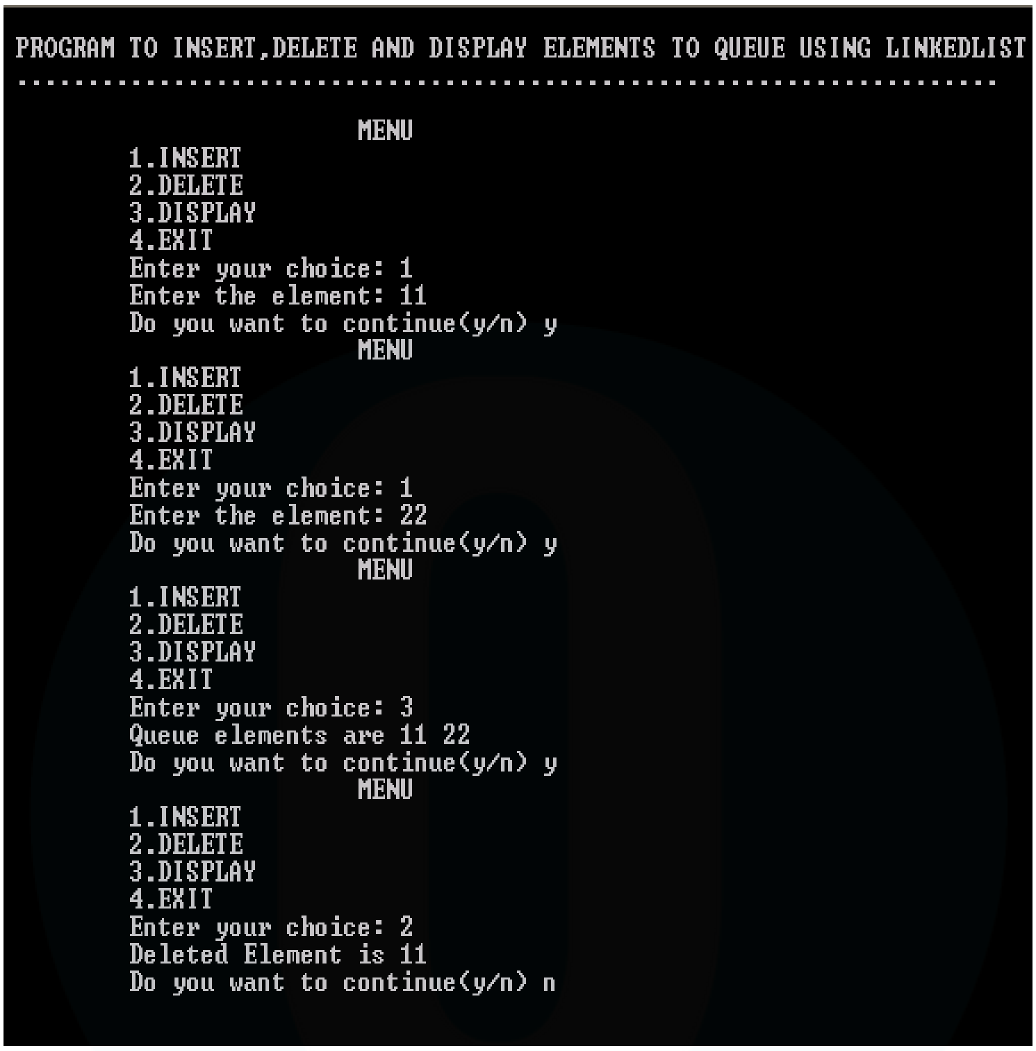
```
Step 1: Start
Step 2: Read the option
Step 3: If (insertion)
    Step 3.1: New = getnode (node)
    Step 3.2: Ptr=header
    Step 3.3: While (ptr->link#Null) do
        Step 3.3.1: Ptr=ptr->link
    Step 3.4: End while
    Step 3.5: Ptr->link=new
    Step 3.6: New->data=x
Step 4: End if
Step 5: If (deletion)
    Step 5.1: Ptr=header->link
    Step 5.2: If (ptr=null)
        Step 5.2.1: Print Empty list
        Step 5.2.2: Exit
    Step 5.3: Else
        Step 5.3.1:Ptr1=ptr->link
        Step 5.3.2: Header->link=ptr1
        Step 5.3.3: Return node (ptr)
        Step 5.3.4: Endif
        Step 5.3.5: Endif
Step 6: If (display)
    Step 6.1: Ptr=header->link
    Step 6.2: While (ptr#null) do
        Step 6.2.1: Display ptr->data
        Step 6.2.2: Ptr=ptr->link
        Step 6.2.3: Endwhile
Step 7: End if
Step 8: Stop
```

PROGRAM DEVELOPMENT:

```
#include<stdio.h>
#include<conio.h>
#include<alloc.h>
typedef struct node
{
int data;
struct node *link;
}NODE;
NODE *front=NULL,*rear=NULL,*s,*ptr,*disply;
void main()
{
int no,item;
char c;
clrscr();
printf("\n\tPROGRAM TO INSERT,DELETE AND DISPLAY ELEMENTS TO QUEUE
USING LINKEDLIST");
printf("\t\t..............................................................\n\n");
do
{
printf("\t\t\t\tMENU");
printf("\n\t\t1.INSERT\n\t\t2.DELETE\n\t\t3.DISPLAY\n\t\t4.EXIT\n\t\tEnter your choice: ");
scanf("%d",&no);
if(no==1)
{
ptr=(NODE*)malloc(sizeof(NODE));
printf("\t\tEnter the element: ");
scanf("%d",&ptr->data);
ptr->link=NULL;
if(rear==NULL)
{
front=ptr;
rear=ptr;
}
else
{
rear->link=ptr;
rear=ptr;
}
}
if(no==2)
{
```

```
if(front==NULL)
printf("\t\tStack is empty\n");
else
{
s=front;
printf("\t\tDeleted Element is %d\n",front->data);
front=front->link;
free(s);
if(front==NULL)
rear=NULL;
}
}
if(no==3)
{
if(front==NULL)
printf("\t\tQueue is empty\n");
else
{
printf("\t\tQueue elements are");
disply=front;
while(disply!=NULL)
{
printf(" %d",disply->data);
disply=disply->link;
}
printf("\n");
}
}
if(no==4)
break;
printf("\t\tDo you want to continue(y/n) ");
scanf(" %c",&c);
}
while(c=='y'||c=='Y');
getch();
}
```

OUTPUT:



```
PROGRAM TO INSERT,DELETE AND DISPLAY ELEMENTS TO QUEUE USING LINKEDLIST
.....................................................................

                        MENU
        1.INSERT
        2.DELETE
        3.DISPLAY
        4.EXIT
        Enter your choice: 1
        Enter the element: 11
        Do you want to continue(y/n) y
                        MENU
        1.INSERT
        2.DELETE
        3.DISPLAY
        4.EXIT
        Enter your choice: 1
        Enter the element: 22
        Do you want to continue(y/n) y
                        MENU
        1.INSERT
        2.DELETE
        3.DISPLAY
        4.EXIT
        Enter your choice: 3
        Queue elements are 11 22
        Do you want to continue(y/n) y
                        MENU
        1.INSERT
        2.DELETE
        3.DISPLAY
        4.EXIT
        Enter your choice: 2
        Deleted Element is 11
        Do you want to continue(y/n) n
```

CONCLUSION:

The algorithm was developed and the program was coded. The program was tested successfully.

Experiment No: 12

SINGLY LINKED LIST

PROBLEM DEFINITION:

Write a program to implement Singly Linked List.

ALGORITHM:

```
Step 1: Start
Step 2: Read the option
Step 3: If (traverse)
        Step 3.1: Ptr=header->link
        Step 3.2: While (ptr!=Null)do
                Step 3.2.1: Print ptr->data
                Step 3.2.2: Ptr=ptr->link
        Step 3.3: End while
        Step 3.4: Stop
Step 4: Elseif (insertion at front)
        Step 4.1: New = getnode (Node)
        Step 4.2: If (new=null)
                Step 4.2.1: Insertion not possible
                Step 4.2.2: Exit
        Step 4.3: Else
                Step 4.3.1: New->link=header->link
                Step 4.3.2: Header->link=new
        Step 4.4: Endif
        Step 4.5: Stop
Step 5: Elseif (insertion at end)
        Step 5.1: New = getnode (node)
        Step 5.2: If (new=null) then
                Step 5.2.1: Print insufficient memory
                Step 5.2.2: Exit
        Step 5.3: Else
                Step 5.3.1: Ptr=header
                Step 5.3.2: While (ptr->link#null)
                        Step 5.3.2.1: Ptr=ptr->link
                Step 5.3.3: End while
                Step 5.3.4: Ptr->link=new
                Step 5.3.5: New->data=x
                Step 5.3.6: New->link=null
        Step 5.4: Endif
        Step 5.5: Stop
Step 6: Else if (inset at any position)
        Step 6.1: New = getnode (node)
        Step 6.2: If (new==null)
                Step 6.2.1: Printf insufficient memory
                Step 6.2.2: Exit
        Step 6.3: Else
                Step 6.3.1: Ptr=header
                Step 6.3.2: While (ptr->data#key) and (ptr->link#null)
                        Step 6.3.2.1: Ptr=ptr->link
                Step 6.3.3: End while
                Step 6.3.4: If (ptr->link=null)
                        Step 6.3.4.1: Print key not available
                        Step 6.3.4.2: Exit
                Step 6.3.5: Else
                        Step 6.3.5.1: New->link=ptr->link
                        Step 6.3.5.2: New->data=v
                        Step 6.3.5.3: Ptr->link =new
                Step 6.3.6: Endif
        Step 6.4: Endif
        Step 6.5: Stop
Step 7: Elseif (deletefront)
        Step 7.1: Ptr=header->link
        Step 7.2: If (ptr=null) then
                Step 7.2.1: Print Empty list
                Step 7.2.2: Exit
        Step 7.3: Else
                Step 7.3.1: Ptr1=ptr->link
                Step 7.3.2: Header->link=ptr1
                Step 7.3.3: Return node (ptr)
        Step 7.4: Endif
        Step 7.5: Stop
Step 8: Else if (delete end)
        Step 8.1: Ptr=header->link
        Step 8.2: If (ptr=null) then
                Step 8.2.1: Print empty list
                Step 8.2.2: Exit
        Step 8.3: Else
                Step 8.3.1: While (ptr->link#Null)
                        Step 8.3.1.1: Ptr1=ptr
                        Step 8.3.1.2: Ptr=ptr->link
                Step 8.3.2: End while
                Step 8.3.3: Ptr->link=null
                Step 8.3.4: Return node(ptr)
                Step 8.3.5: Endif
                Step 8.3.6: Stop
Step 9: Elseif (delete any)
        Step 9.1: Ptr1=header
        Step 9.2: Ptr=ptr1->link
        Step 9.3: If (ptr==null) then
                Step 9.3.1: Print Empty list
                Step 9.3.2: Exit
        Step 9.4: Else
                Step 9.4.1: While (ptr#null) and (ptr->data#key) do
                        Step 9.4.1.1: Ptr1=ptr
                        Step 9.4.1.2: Ptr=ptr->link
                Step 9.4.2: End while
                Step 9.4.3: If (ptr==null) then
                        Step 9.4.3.1: Print Key not present
                        Step 9.4.3.2: Exit
                Step 9.4.4: Else
                        Step 9.4.4.1: Ptr1->link=ptr->link
                        Step 9.4.4.2: Return node (ptr)
                Step 9.4.5: End if
                Step 9.4.6: Stop
Step 9: Endif
Step 10: Stop
```

PROGRAM DEVELOPMENT:

```
#include<stdio.h>
#include<conio.h>
#include<alloc.h>
void traverse();
void insertfront();
void insertend();
void insertany();
void deletefront();
void deleteend();
void deleteany();
typedef struct node
{
int data;
struct node *link;
}NODE;
NODE *header=NULL,*newptr=NULL,*ptr,*ptr1;
void main()
```

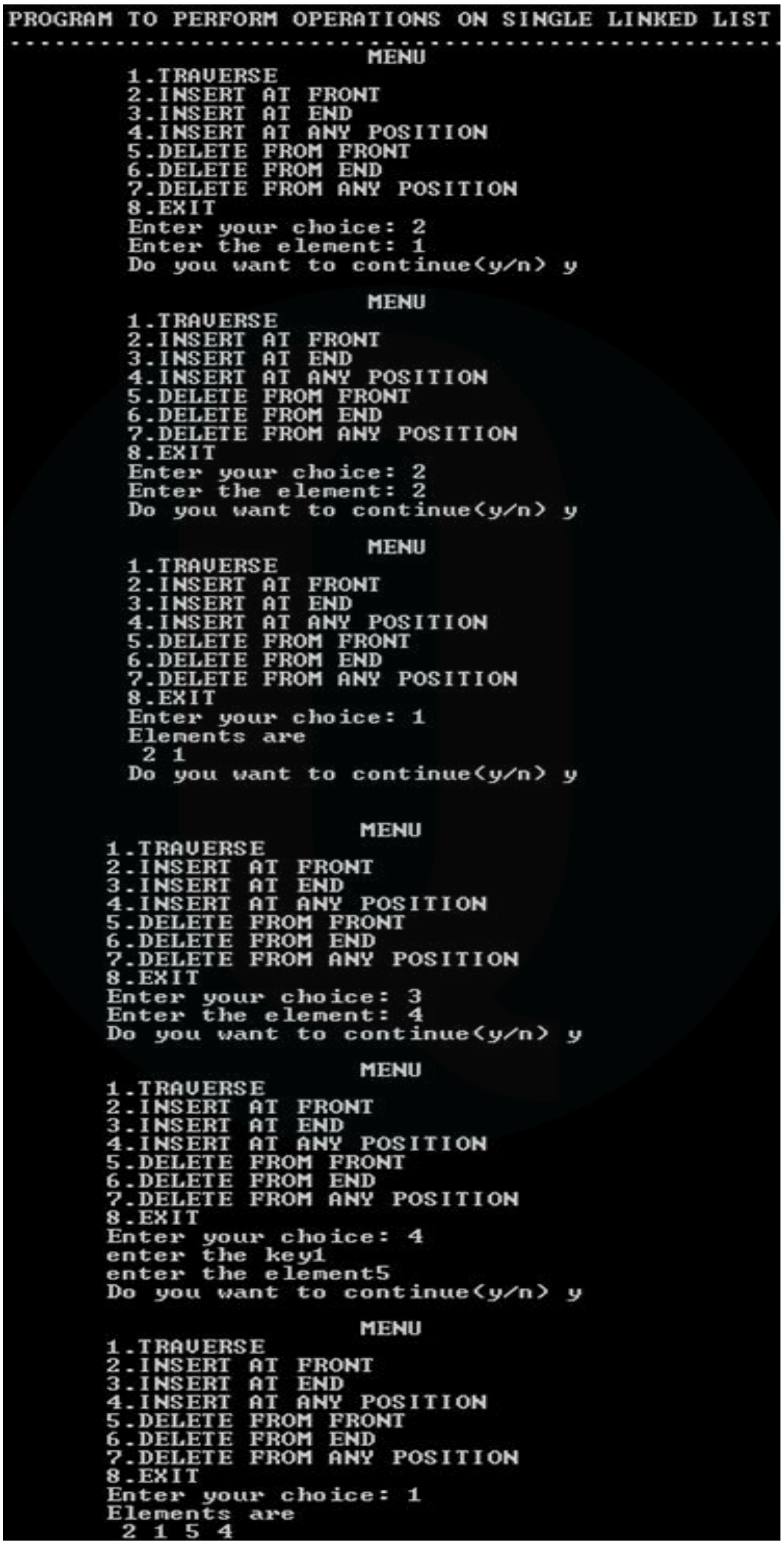
```
{
int no,item;
char c;
clrscr();
printf("\n\tPROGRAM TO PERFORM OPERATIONS ON SINGLE LINKED LIST");
printf("\n\t.......................................................");
do
{
printf("\n\t\t\t\tMENU\n\n");
printf("\t\t1.TRAVERSE\n\t\t2.INSERT AT FRONT\n\t\t3.INSERT AT END\n\t\t4.INSERT AT
ANY POSITION\n\t\t5.DELETE FROM FRONT\n\t\t6.DELETE FROM END\n\t\t7.DELETE
FROM ANY POSITION\n\t\t8.EXIT\n\t\tEnter your choice: ");
scanf("%d",&no);
if(no==8)
break;
switch(no)
{
case 1:
            traverse();
            break;
case 2:
            insertfront();
            break;
case 3:
            insertend();
            break;
case 4:
            insertany();
            break;
case 5:
            deletefront();
            break;
case 6:
            deleteend();
            break;
case 7:
            deleteany();
            break;
default :
           printf("\t\tINVALID ENTRY");
           break;
}
printf("\t\tDo you want to continue(y/n) ");
scanf(" %c",&c);
```

```
}
while(c=='y'||c=='Y');
getch();
}
void insertfront()
{
newptr=(NODE*)malloc(sizeof(NODE));
printf("\t\tEnter the element: ");
scanf("%d",&newptr->data);
newptr->link=NULL;
if(newptr==NULL)
printf("\t\tInsufficient memmory");
else
{
newptr->link=header->link;
header->link=newptr;
}
}
void insertend()
{
newptr=(NODE*)malloc(sizeof(NODE));
if(newptr==NULL)
printf("\t\tInsufficient memmory");
else
{
printf("\t\tEnter the element: ");
scanf("%d",&newptr->data);
newptr->link=NULL;
ptr=header;
while(ptr->link!=NULL)
ptr=ptr->link;
newptr->link=ptr->link;
ptr->link=newptr;
}
}
void insertany()
{
int key;
newptr=(NODE*)malloc(sizeof(NODE));
if(newptr==NULL)
printf("\t\tInsufficient memmory");
else
{
printf("\t\tenter the key");
```

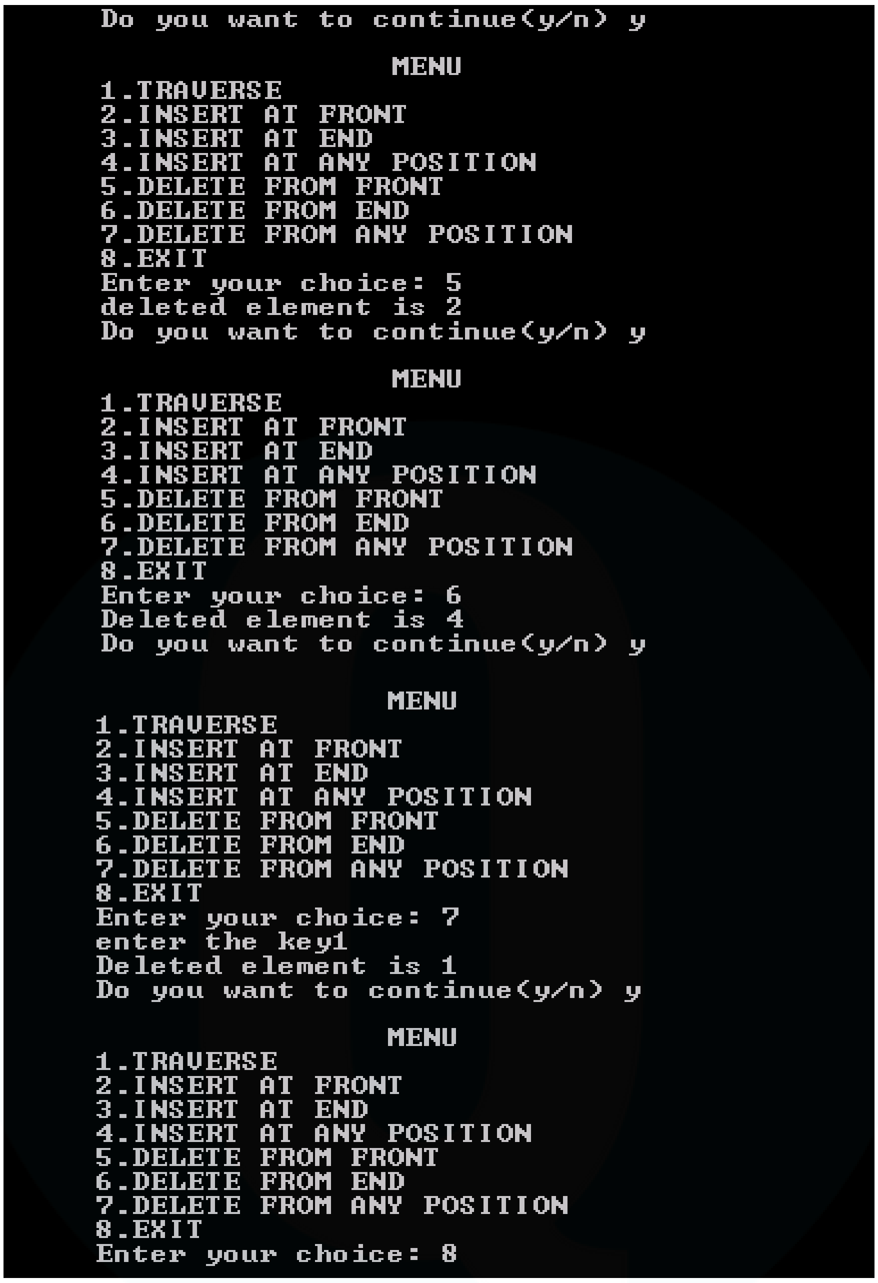
```
scanf("%d",&key);
printf("\t\tenter the element");
scanf("%d",&newptr->data);
ptr=header->link;
while(ptr->data!=key&&ptr!=NULL)
ptr=ptr->link;
if(ptr==NULL)
printf("\t\tkey is not found");
else
{
newptr->link=ptr->link;
ptr->link=newptr;
}
}
}
void deletefront()
{
ptr=header->link;
ptr1=ptr->link;
if(ptr==NULL)
printf("\t\tEmpty list");
else
{
header->link=ptr1;
printf("\t\tdeleted element is %d",ptr->data);
free(ptr);
}
printf("\n");
}
void deleteend()
{
ptr=header;
ptr1=ptr->link;
if(ptr1==NULL)
printf("\t\tEmpty list");
else
{
while(ptr1->link!=NULL)
{
ptr=ptr->link;
ptr1=ptr1->link;
}
ptr->link=NULL;
printf("\t\tDeleted element is %d",ptr1->data);
```

```
free(ptr1);
}
printf("\n");
}
void deleteany()
{
int key;
ptr=header;
ptr1=ptr->link;
if(ptr1==NULL)
printf("\t\tEmpty list");
else
{
printf("\t\tenter the key");
scanf("%d",&key);
while(ptr1->data!=key&&ptr1!=NULL)
{
ptr=ptr1;
ptr1=ptr1->link;
}
if(ptr1==NULL)
printf("\t\tKey not found");
else if(ptr1->data==key)
{
ptr->link=ptr1->link;
printf("\t\tDeleted element is %d",ptr1->data);
free(ptr1);}
printf("\n");
}}
void traverse()
{
if(header->link==NULL)
printf("\t\tlist is empty\n");
else
{
printf("\t\tElements are\n");
ptr=header->link;
printf("\t\t");
while(ptr!=NULL)
{
printf(" %d",ptr->data);
ptr=ptr->link;
}
printf("\n");}}
```

OUTPUT:



```
PROGRAM TO PERFORM OPERATIONS ON SINGLE LINKED LIST
-------------------------------------------------------
                        MENU
        1.TRAVERSE
        2.INSERT AT FRONT
        3.INSERT AT END
        4.INSERT AT ANY POSITION
        5.DELETE FROM FRONT
        6.DELETE FROM END
        7.DELETE FROM ANY POSITION
        8.EXIT
        Enter your choice: 2
        Enter the element: 1
        Do you want to continue(y/n) y

                        MENU
        1.TRAVERSE
        2.INSERT AT FRONT
        3.INSERT AT END
        4.INSERT AT ANY POSITION
        5.DELETE FROM FRONT
        6.DELETE FROM END
        7.DELETE FROM ANY POSITION
        8.EXIT
        Enter your choice: 2
        Enter the element: 2
        Do you want to continue(y/n) y

                        MENU
        1.TRAVERSE
        2.INSERT AT FRONT
        3.INSERT AT END
        4.INSERT AT ANY POSITION
        5.DELETE FROM FRONT
        6.DELETE FROM END
        7.DELETE FROM ANY POSITION
        8.EXIT
        Enter your choice: 1
        Elements are
         2 1
        Do you want to continue(y/n) y


                        MENU
       1.TRAVERSE
       2.INSERT AT FRONT
       3.INSERT AT END
       4.INSERT AT ANY POSITION
       5.DELETE FROM FRONT
       6.DELETE FROM END
       7.DELETE FROM ANY POSITION
       8.EXIT
       Enter your choice: 3
       Enter the element: 4
       Do you want to continue(y/n) y

                        MENU
       1.TRAVERSE
       2.INSERT AT FRONT
       3.INSERT AT END
       4.INSERT AT ANY POSITION
       5.DELETE FROM FRONT
       6.DELETE FROM END
       7.DELETE FROM ANY POSITION
       8.EXIT
       Enter your choice: 4
       enter the key1
       enter the element5
       Do you want to continue(y/n) y

                        MENU
       1.TRAVERSE
       2.INSERT AT FRONT
       3.INSERT AT END
       4.INSERT AT ANY POSITION
       5.DELETE FROM FRONT
       6.DELETE FROM END
       7.DELETE FROM ANY POSITION
       8.EXIT
       Enter your choice: 1
       Elements are
        2 1 5 4
```



```
Do you want to continue(y/n) y

                MENU
1.TRAVERSE
2.INSERT AT FRONT
3.INSERT AT END
4.INSERT AT ANY POSITION
5.DELETE FROM FRONT
6.DELETE FROM END
7.DELETE FROM ANY POSITION
8.EXIT
Enter your choice: 5
deleted element is 2
Do you want to continue(y/n) y

                MENU
1.TRAVERSE
2.INSERT AT FRONT
3.INSERT AT END
4.INSERT AT ANY POSITION
5.DELETE FROM FRONT
6.DELETE FROM END
7.DELETE FROM ANY POSITION
8.EXIT
Enter your choice: 6
Deleted element is 4
Do you want to continue(y/n) y

                MENU
1.TRAVERSE
2.INSERT AT FRONT
3.INSERT AT END
4.INSERT AT ANY POSITION
5.DELETE FROM FRONT
6.DELETE FROM END
7.DELETE FROM ANY POSITION
8.EXIT
Enter your choice: 7
enter the key1
Deleted element is 1
Do you want to continue(y/n) y

                MENU
1.TRAVERSE
2.INSERT AT FRONT
3.INSERT AT END
4.INSERT AT ANY POSITION
5.DELETE FROM FRONT
6.DELETE FROM END
7.DELETE FROM ANY POSITION
8.EXIT
Enter your choice: 8
```

CONCLUSION:

The algorithm was developed and the program was coded. The program was tested successfully.

Experiment No: 13

POLYNOMIAL ADDITION

PROBLEM DEFINITION:

Write a program to implement Polynomial Addition.

ALGORITHM:

```
Step 1: Start
Step 2: Read coefficient &exponent of 2 polynomials
Step 3: Compare the exponents from 1st node
Step 4: If (poly1->coeff>poly2->coeff)
        Step 4.1: Poly->coeff=poly1->coeff
        Step 4.2: Poly->exp=poly1->exp
        Step 4.3: Poly1=poly1->link
        Step 4.4: Poly->link = getnode (node)
        Step 4.5: Poly=poly->link
        Step 4.6: Exit
Step 5: Elseif (poly->exp<poly2->exp)
        Step 5.1: Poly->coeff=poly2->coeff
        Step 5.2: Poly->exp=poly2->exp
        Step 5.3:Poly2=poly2->link
        Step 5.4: Poly->link = getnode (node)
        Step 5.5: Poly=poly->link
        Step 5.6: Exit
Step 6: Else
        Step 6.1: Poly->coeff=poly2->coeff+poly1->coeff
        Step 6.2: Poly->exp=poly1->exp
        Step 6.3: Poly2=poly2->link
        Step 6.4: Poly1=poly1->link
        Step 6.5: Poly->link = getnode (node)
        Step 6.6: Poly=poly->link
        Step 6.7: Exit
Step 7: If (poly1->link!=Null)
        Step 7.1: While (poly1->link!=Null)
                Step 7.1.1: Poly->coeff=poly1->coeff
                Step 7.1.2: Poly->exp=poly1->exp
                Step 7.1.3: Poly1=poly1->link
                Step 7.1.4: Poly->link=getnode (node)
                Step 7.1.5: Poly=polylink
        Step 7.2: End while
Step 8: Else if
        Step 8.1: While (poly2link!=Null)
                Step 8.1.1: Poly->coeff=poly2->coeff
                Step 8.1.2: Poly->exp=poly2->exp
                Step 8.1.3: Poly2=poly2->link
                Step 8.1.4: Poly->link = getnode (node)
                Step 8.1.5: Poly=poly->link
        Step 8.2: End while
Step 9: End if
Step 10: Stop
```

PROGRAM DEVELOPMENT:

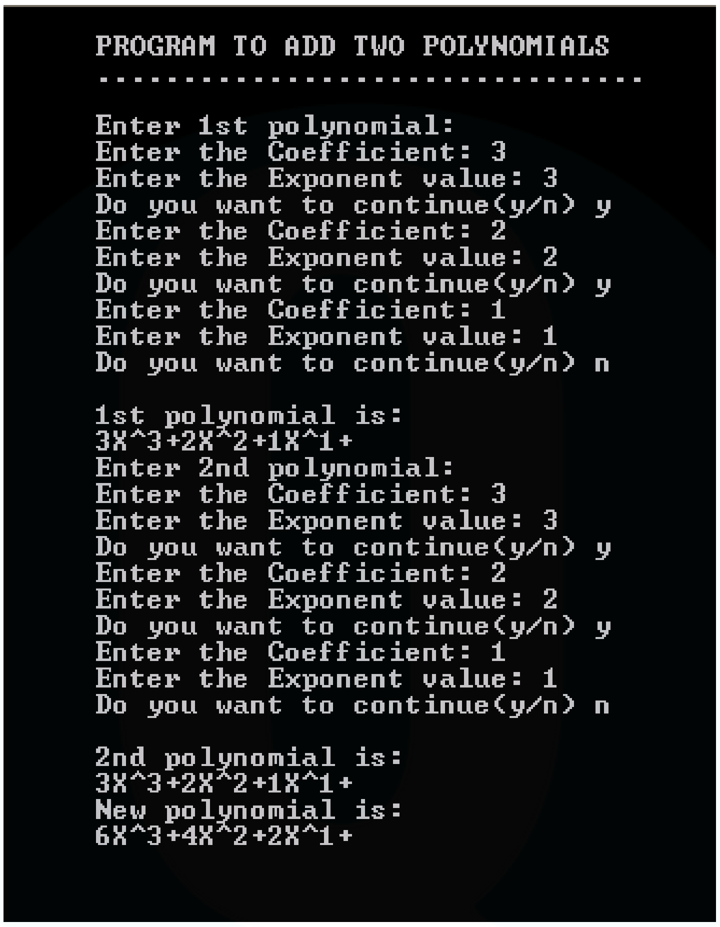
```
#include<stdio.h>
#include<conio.h>
#include<alloc.h>
typedef struct node
{
int coeff;
int exp;
struct node*link;
}NODE;
NODE *poly1=NULL,*poly2=NULL,*poly=NULL;
void create(NODE*);
void show(NODE*);
void polyadd(NODE*,NODE*,NODE*);
void main()
{
clrscr();
printf("\n\t\tPROGRAM TO ADD TWO POLYNOMIALS\n");
printf("\t\t...............................\n");
poly=(NODE*)malloc(sizeof(NODE));
poly1=(NODE*)malloc(sizeof(NODE));
poly2=(NODE*)malloc(sizeof(NODE));
printf("\n\t\tEnter 1st polynomial: ");
create(poly1);
printf("\n\t\t1st polynomial is:  ");
show(poly1);
printf("\n\t\tEnter 2nd polynomial: ");
create(poly2);
printf("\n\t\t2nd polynomial is: ");
show(poly2);
polyadd(poly1,poly2,poly);
```

```
printf("\n\t\tNew polynomial is: ");
show(poly);
getch();
}
void create(NODE*ptr)
{
char c;
printf("\n");
do
{
printf("\t\tEnter the Coefficient: ");
scanf("%d",&ptr->coeff);
printf("\t\tEnter the Exponent value: ");
scanf("%d",&ptr->exp);
ptr->link=(NODE*)malloc(sizeof(NODE));
ptr=ptr->link;
ptr->link=NULL;
printf("\t\tDo you want to continue(y/n) ");
scanf(" %c",&c);
}
while(c=='y'||c=='Y');
}
void show(NODE*ptr)
{
printf("\n\t\t");
while(ptr->link!=NULL)
{
if(ptr->exp==0)
printf("%d",ptr->coeff);
else
printf("%dX^%d+",ptr->coeff,ptr->exp);
ptr=ptr->link;
}
}
void polyadd(NODE*ptr1,NODE*ptr2,NODE*ptr)
{
while(ptr1->link!=NULL&&ptr2->link!=NULL)
{
if(ptr1->exp>ptr2->exp)
{
ptr->coeff=ptr1->coeff;
ptr->exp=ptr1->exp;
ptr1=ptr1->link;
ptr->link=(NODE*)malloc(sizeof(NODE));
```

```
ptr=ptr->link;
ptr->link=NULL;
}
else if(ptr1->exp<ptr2->exp)
{
ptr->coeff=ptr2->coeff;
ptr->exp=ptr2->exp;
ptr2=ptr2->link;
ptr->link=(NODE*)malloc(sizeof(NODE));
ptr=ptr->link;
ptr->link=NULL;
}
else
{
ptr->coeff=ptr1->coeff+ptr2->coeff;
ptr->exp=ptr1->exp;
ptr1=ptr1->link;
ptr2=ptr2->link;
ptr->link=(NODE*)malloc(sizeof(NODE));
ptr=ptr->link;
ptr->link=NULL;
}
}
if(ptr1->link!=NULL)
{
while(ptr1->link!=NULL)
{
ptr->coeff=ptr1->coeff;
ptr->exp=ptr1->exp;
ptr1=ptr1->link;
ptr->link=(NODE*)malloc(sizeof(NODE));
ptr=ptr->link;
ptr->link=NULL;
}
}
else if(ptr2->link!=NULL)
{
while(ptr2->link!=NULL)
{
ptr->coeff=ptr2->coeff;
ptr->exp=ptr2->exp;
ptr2=ptr2->link;
ptr->link=(NODE*)malloc(sizeof(NODE));
ptr=ptr->link;
```

```
ptr->link=NULL;
}
}
}
```

OUTPUT:



```
PROGRAM TO ADD TWO POLYNOMIALS
------------------------------

Enter 1st polynomial:
Enter the Coefficient: 3
Enter the Exponent value: 3
Do you want to continue(y/n) y
Enter the Coefficient: 2
Enter the Exponent value: 2
Do you want to continue(y/n) y
Enter the Coefficient: 1
Enter the Exponent value: 1
Do you want to continue(y/n) n

1st polynomial is:
3X^3+2X^2+1X^1+
Enter 2nd polynomial:
Enter the Coefficient: 3
Enter the Exponent value: 3
Do you want to continue(y/n) y
Enter the Coefficient: 2
Enter the Exponent value: 2
Do you want to continue(y/n) y
Enter the Coefficient: 1
Enter the Exponent value: 1
Do you want to continue(y/n) n

2nd polynomial is:
3X^3+2X^2+1X^1+
New polynomial is:
6X^3+4X^2+2X^1+
```

CONCLUSION:

The algorithm was developed and the program was coded. The program was tested successfully.

Experiment No: 14

INFIX TO POSTFIX EVALUATION

PROBLEM DEFINITION:

Write a program to implement Infix to Postfix Evaluation.

ALGORITHM:

```
Step 1: Start
Step 2: Read the expression
Step 3: For I from 0 to length
    Step 3.1: Set ch as a[i]
    Step 3.2: If (ch='c')
        Step 3.2.1: Push (ch)
    Step 3.3: If (ch=')')
        Step 3.3.1: While (stk [top]!='(')
            Step 3.3.1.1: Read stk [top]=post[j++]
            Step 3.3.1.2: Decrement top
        Step 3.3.2: End loop
        Step 3.3.3: Decrement top
    Step 3.4: If (ch='+' or ch='-' or ch='*' orch='/')
        Step 3.4.1: If (top=-1 or stk[top]='(')
            Step 3.4.1.1: Push (ch)
        Step 3.4.2: Else
            Step 3.4.2.1: x=priority (ch)
            Step 3.4.2.2: y=priority (stk [top])
            Step 3.4.2.3: If(y>=x)
                Step 3.4.2.3.1: Read stk [top] to post [j++]
                Step 3.4.2.3.2: Decrement top
                Step 3.4.2.3.3: Push (ch)
            Step 3.4.2.4: Else
                Step 3.4.2.4.1: Push (ch)
            Step 3.4.2.5: End if
        Step 3.4.3: End if
    Step 3.5: If (ch is an alphabet)
        Step 3.5.1: Read ch to post [j++]
    Step 3.6: End if
Step 4: End loop
Step 5: While (stk [top] !='\0')
    Step 5.1: Read stk [top] to post [j++]
    Step 5.2: Decrement top
Step 6: End loop
Step7: Assign post[i] as '\0'
Step 8: Print "Post fix expression as post"
Step9: Stop
```

Eval-Postfix ()

```
Step 1: Start
Step 2: Find postfix expression for given expression
Step 3: For I from 0 to length of post
        Step 3.1: Set ch as post[i]
        Step 3.2: If ch is an alphabet
                Step 3.2.1: Print "Enter the value for ch"
                Step 3.2.2: Read the value to c.
                Step 3.3.3: Push C to another stk.
        Step 3.3: Else
                Step 3.3.1: Set o1 as stk [top]
                Step 3.3.2: Decrement top
                Step 3.3.3: Set o2 as stk [top]
                Step 3.3.4: Decrement top
                Step 3.3.5: If(ch==   +)
                        Step 3.3.5.1: x=o1+o2
                Step 3.3.6: If(ch= -)
                        Step 3.3.6.1: x=o1-o2
                Step 3.3.7: If (ch=*)
                        Step 3.3.7.1: x=o1*o2
                Step 3.3.8: If (ch=/)
                        Step 3.3.8.1: x=o1/o2
                Step 3.3.9: End if
                Step 3.3.10: Push (x)
        Step 3.4: End if
Step 4: End for
Step 5: Print value as stk [top]
Step 6: Stop
```

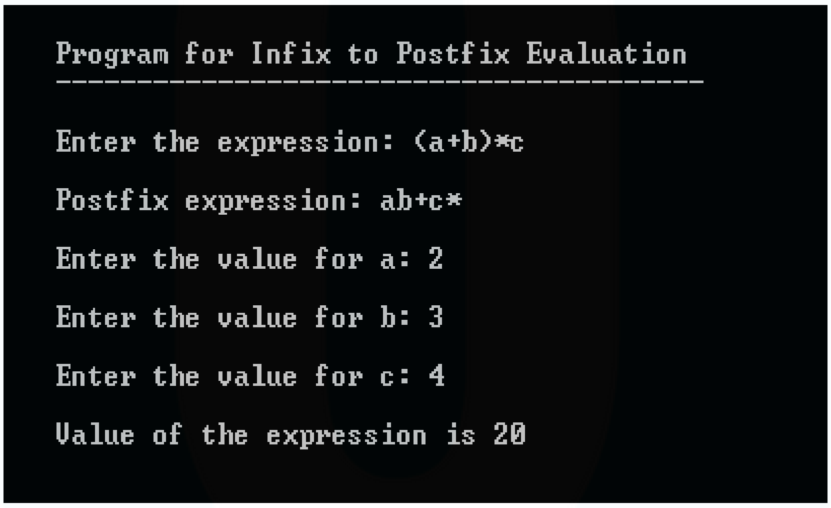
PROGRAM DEVELOPMENT:

```
#include <stdio.h>
#include<conio.h>
#include<string.h>
void push(char);
void push1(int);
int priority(char);
void read();
```

```
int top=-1,top1=-1,j=0,i,x,y;
char stk[50],stk1[50],a[50],ch,post[50];
void main()
{
clrscr();
printf("\n\n\tProgram for Infix to Postfix Evaluation");
printf("\n\t-----------------------------------------\n");
printf("\n\tEnter the expression: ");
gets(a);
for(i=0;a[i]!='\0';i++)
{
ch=a[i];
switch(ch)
{
case '(':push(ch);
break;
case')':while (stk[top]!='(')
{
post[j++]=stk[top];
top--;
}
top--;
break;
case '+':
case '-':
case '^':
case '/':
case '*': if (top== -1||stk[top]=='(')
push(ch);
else
{x=priority(ch);
y=priority(stk[top]);
if(y>=x)
{
post[j++]=stk[top];
top--;
push(ch);}
else
push(ch);
}
break;
default:
if(isalpha(ch))
post[j++]=ch;
break;
}
}
while(stk[top]!='\0')
{
post[j++]=stk[top];
```

```
top--;
}
post[j]='\0';

printf("\n\tPostfix expression: ");
puts(post);
read();
getch();
}
void push(char ch)
{
top++;
stk[top]=ch;
}
void push1(int ch)
{top1++;
stk1[top1]=ch;
}
int priority(char c)
{
if (c=='t'||c=='-')
return 1;
else if(c=='*'||c=='/')
return 2;
else if(c=='^')
return 3;
else
return 0;
}
void read()
{
int c,o1,o2;
for(i=0;post[i]!='\0';i++)
{ch=post[i];
if(isalpha(ch))
{printf("\n\tEnter the value for %c: ",ch);
scanf("%d", &c);
push1(c);
}
else {
o1=stk1[top1];
top1--;
o2=stk1[top1];
top1--;
switch(ch)
{
case '+':x=o1+o2;
break;
case'-':x=o1-o2;
break;
case'*':x=o1*o2;
break;
case'/':x=o1/o2;
break;
case'^':x=o1^o2;
break;
default:
break;
}
push1(x);
}
}
printf("\n\tValue of the expression is %d",stk1[top1]);
}
```

OUTPUT:



```
Program for Infix to Postfix Evaluation
----------------------------------------

Enter the expression: (a+b)*c

Postfix expression: ab+c*

Enter the value for a: 2

Enter the value for b: 3

Enter the value for c: 4

Value of the expression is 20
```

CONCLUSION:

The algorithm was developed and the program was coded. The program was tested successfully.

Experiment No: 15

IMPLEMENTATION OF TREE

PROBLEM DEFINITION:

Write a program to implement Tree.

ALGORITHM:

```
Step 1: Start
Step 2: Ptr=new node ()
Step 3: Repeat the following steps until info!=0
Step 4: Enter the ptr->data
Step 5: Ptr->left=NULL and ptr->right=NULL
Step 6: If (root=NULL)
        Step 6.1: Root =ptr
        Step 6.2: Start=root
Step 7: Else
        Step 7.1: Root=start
        Step 7.2: While (root!=NULL)
                Step 7.2.1: If (ptr->data<root->data)
                        Step 7.2.1.1: If (root->left=NULL)
                                Step 7.2.1.1.1: Root->left=ptr
                                Step 7.2.1.1.2: Exit while loop
                Step 7.2.2: Root=root->left
                Step 7.2.3: Else
                        Step 7.2.3.1: If (root->right==NULL)
                        Step 7.2.3.2: Root->right=ptr
                        Step 7.2.3.3: Exit while loop
                        Step 7.2.3.4: Root=root->right
                Step 7.2.4: End if
        Step 7.3: End While
Step 8: End if
Step 9: Stop
```

Preorder ()

```
Step 1: Start
Step 2: If (p=NULL) ie start= NULL
        Step 2.1: Tree traversal not possible
Step 3: Else
        Step 3.1: Print p->data
        Step 3.2: Preorder (p->left)
        Step 3.3: Preorder (p->right)
Step 4: End if
Step 5: Stop
```

Inorder ()

```
Step 1: Start
Step 2: If (p=NULL)
        Step 2.1: Tree traversal not possible
Step 3: Else
        Step 3.1: Inorder (p->left)
        Step 3.2: Print p->data
        Step 3.3: Inorder (p->right)
Step 4: End if
Step 5: Stop
```

Postorder ()

```
Step 1: Start
Step 2: If (p==NULL)
        Step 2.1: Tree traversal not possible
Step 3: Else
        Step 3.1: Postorder (p->left)
        Step 3.2: Postorder (p->right)
        Step 3.3: Print p->data
Step 4: End if
Step 5: Stop
```

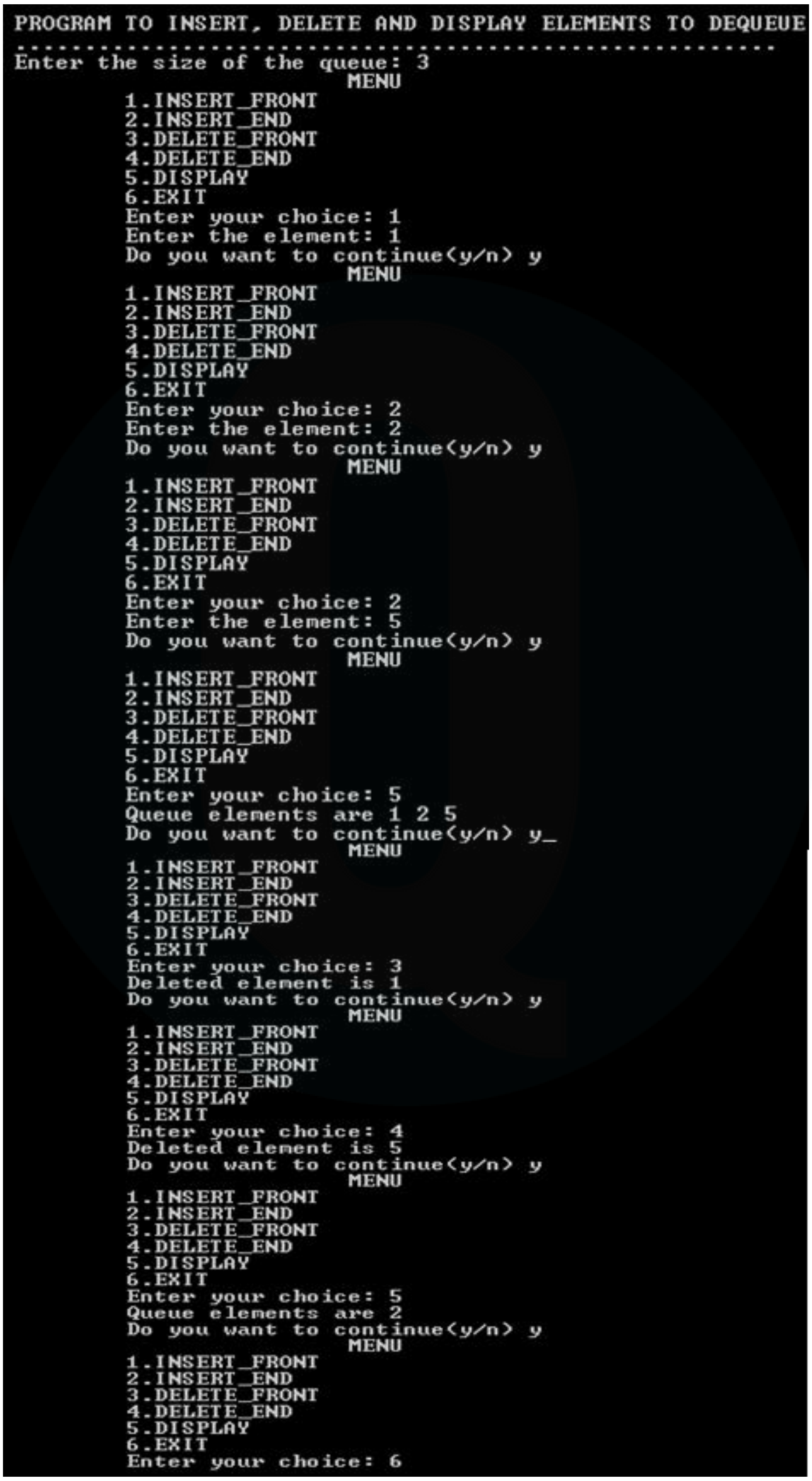
PROGRAM DEVELOPMENT:

```
# include <stdio.h>
# include <conio.h>
# include <stdlib.h>
 typedef struct BST {
int data;
struct BST *lchild, *rchild;
} node;
void insert(node *, node *);
void inorder(node *);
void preorder(node *);
void postorder(node *);
node *search(node *, int, node **);
```

```
int f=0;
void main() {
int choice;
char ans = 'N';
int key;
node *new_node, *root, *tmp, *parent;
node *get_node();
root = NULL;
clrscr();
printf("\n\n\tProgram For Binary Search Tree");
printf("\n\n\t------------------------------");
do {
printf("\n");
printf("\n\tMenu");
printf("\n\t----");
printf("\n\t1.Create");
printf("\n\t2.Search");
printf("\n\t3.Traversals");
printf("\n\t4.Exit");
printf("\n\tEnter your choice: ");
scanf("%d", &choice);
switch (choice) {
case 1:
do {
new_node = get_node();
printf("\tEnter The Element: ");
scanf("%d", &new_node->data);
if (root == NULL) /* Tree is not Created */
root = new_node;
else
insert(root, new_node);
printf("\tWant To enter More Elements?(y/n): ");
scanf(" %c",ans);
ans = getch();
} while (ans == 'y');
break;
case 2:
printf("\tEnter Element to be searched: ");
scanf("%d", &key);
tmp = search(root, key, &parent);
if(tmp!=NULL)
{
printf("\n\tParent of node %d is %d\n", tmp->data, parent->data);
}
break;
case 3:
 if (root == NULL)
printf("Tree Is Not Created");
else {
printf("\n\tThe Inorder display : ");
inorder(root);
printf("\n\tThe Preorder display : ");
preorder(root);
printf("\n\tThe Postorder display : ");
postorder(root);
}
printf("\n");
break;
}
} while (choice != 4);
/*
 Get new Node
 */
node *get_node() {
node *temp;
temp = (node *) malloc(sizeof(node));
temp->lchild = NULL;
temp->rchild = NULL;
return temp;
/*
 This function is for creating a binary search tree
 */
void insert(node *root, node *new_node) {
if (new_node->data < root->data) {
if (root->lchild == NULL)
root->lchild = new_node;
else
insert(root->lchild, new_node);
}
if (new_node->data > root->data) {
if (root->rchild == NULL)
root->rchild = new_node;
else
insert(root->rchild, new_node);
}
}
/*
 This function is for searching the node from
 binary Search Tree
 */
node *search(node *root, int key, node **parent) {
node *temp;
temp = root;
while (temp != NULL) {
if (temp->data == key) {
f=1;
printf("\tElement %d is Present", temp->data);
return temp;
}
*parent = temp;
if (temp->data > key)
temp = temp->lchild;
else
temp = temp->rchild;
}
if(f==0)
{
printf("\nElement not [present");
return NULL;
}
}
/*
 This function displays the tree in inorder fashion
 */
void inorder(node *temp) {
if (temp != NULL) {
inorder(temp->lchild);
printf("%d", temp->data);
inorder(temp->rchild);
}
}
/*
 This function displays the tree in preorder fashion
 */
void preorder(node *temp) {
if (temp != NULL) {
printf("%d", temp->data);
preorder(temp->lchild);
preorder(temp->rchild);
}
}
/*
```

```
 This function displays the tree in postorder fashion
 */
void postorder(node *temp) {
if (temp != NULL) {
postorder(temp->lchild);
postorder(temp->rchild);
printf("%d", temp->data);
 }
}
```

OUTPUT:



```
PROGRAM TO INSERT, DELETE AND DISPLAY ELEMENTS TO DEQUEUE
.............................................................
Enter the size of the queue: 3
                        MENU
        1.INSERT_FRONT
        2.INSERT_END
        3.DELETE_FRONT
        4.DELETE_END
        5.DISPLAY
        6.EXIT
        Enter your choice: 1
        Enter the element: 1
        Do you want to continue(y/n) y
                        MENU
        1.INSERT_FRONT
        2.INSERT_END
        3.DELETE_FRONT
        4.DELETE_END
        5.DISPLAY
        6.EXIT
        Enter your choice: 2
        Enter the element: 2
        Do you want to continue(y/n) y
                        MENU
        1.INSERT_FRONT
        2.INSERT_END
        3.DELETE_FRONT
        4.DELETE_END
        5.DISPLAY
        6.EXIT
        Enter your choice: 2
        Enter the element: 5
        Do you want to continue(y/n) y
                        MENU
        1.INSERT_FRONT
        2.INSERT_END
        3.DELETE_FRONT
        4.DELETE_END
        5.DISPLAY
        6.EXIT
        Enter your choice: 5
        Queue elements are 1 2 5
        Do you want to continue(y/n) y_
                        MENU
        1.INSERT_FRONT
        2.INSERT_END
        3.DELETE_FRONT
        4.DELETE_END
        5.DISPLAY
        6.EXIT
        Enter your choice: 3
        Deleted element is 1
        Do you want to continue(y/n) y
                        MENU
        1.INSERT_FRONT
        2.INSERT_END
        3.DELETE_FRONT
        4.DELETE_END
        5.DISPLAY
        6.EXIT
        Enter your choice: 4
        Deleted element is 5
        Do you want to continue(y/n) y
                        MENU
        1.INSERT_FRONT
        2.INSERT_END
        3.DELETE_FRONT
        4.DELETE_END
        5.DISPLAY
        6.EXIT
        Enter your choice: 5
        Queue elements are 2
        Do you want to continue(y/n) y
                        MENU
        1.INSERT_FRONT
        2.INSERT_END
        3.DELETE_FRONT
        4.DELETE_END
        5.DISPLAY
        6.EXIT
        Enter your choice: 6
```

CONCLUSION:

The algorithm was developed and the program was coded. The program was tested successfully.

Experiment No: 16

IMPLEMENTATION OF GRAPH

PROBLEM DEFINITION:

Write a program to implement Graph.

ALGORITHM:

```
Step 1: Start
Step 2: Enter the data
Step 3: Store the data in an array
Step 4: To find BFS goto step 5
Step 5: Set i=0,j=0
Step 6: Repeat the steps 6 to 6.3 till i<n
        Step 6.1: If vis[i]=0 then vis[i]=I and print data[i]
        Step 6.2: Repeat the steps 6.2 to 6.3 till j<n
        Step 6.3:I f adj[i][j]=1&vis[j]=0,then set vis[j]=1& print data[j]
Step 7: To find DFS goto Step 8
Step 8: Vist[x] = 1
        Step 8.1: Print data[x]
Step 9: Repeat the following steps till j<=n
        Step 9.1: If adj[x][j]=1 and vis[j]=0
        Step 9.2: Dfs (j)
Step 10: Stop
```

PROGRAM DEVELOPMENT:

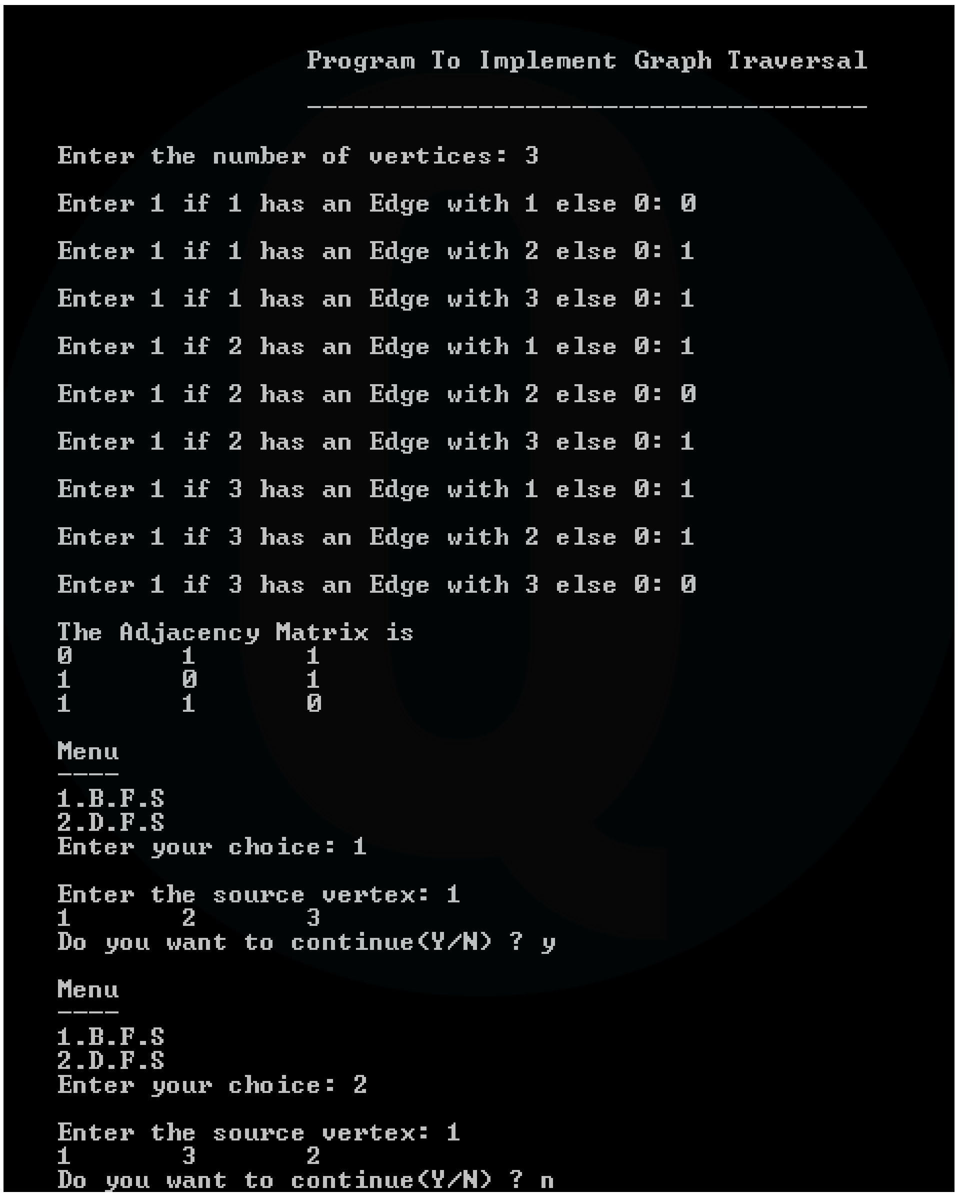
```
#include<stdio.h>
int q[20],top=-1,front=-1,rear=-1,a[20][20],vis[20],stack[20];
int delete();
void add(int item);
void bfs(int s,int n);
void dfs(int s,int n);
void push(int item);
int pop();

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'));
}
```

```
//**************BFS(breadth-first search) code**************//
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);
}
}

//***************DFS(depth-first search) code*****************//
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:



```
                    Program To Implement Graph Traversal
                    ------------------------------------

    Enter the number of vertices: 3

    Enter 1 if 1 has an Edge with 1 else 0: 0

    Enter 1 if 1 has an Edge with 2 else 0: 1

    Enter 1 if 1 has an Edge with 3 else 0: 1

    Enter 1 if 2 has an Edge with 1 else 0: 1

    Enter 1 if 2 has an Edge with 2 else 0: 0

    Enter 1 if 2 has an Edge with 3 else 0: 1

    Enter 1 if 3 has an Edge with 1 else 0: 1

    Enter 1 if 3 has an Edge with 2 else 0: 1

    Enter 1 if 3 has an Edge with 3 else 0: 0

    The Adjacency Matrix is
    0       1       1
    1       0       1
    1       1       0

    Menu
    ----
    1.B.F.S
    2.D.F.S
    Enter your choice: 1

    Enter the source vertex: 1
    1       2       3
    Do you want to continue(Y/N) ? y

    Menu
    ----
    1.B.F.S
    2.D.F.S
    Enter your choice: 2

    Enter the source vertex: 1
    1       3       2
    Do you want to continue(Y/N) ? n
```

CONCLUSION:

The algorithm was developed and the program was coded. The program was tested successfully.

VIVA VOICE QUESTIONS AND ANSWERS

1) What is data structure?

Data structures refer to the way data is organized and manipulated. It seeks to find ways to make data access more efficient. When dealing with data structure, we not only focus on one piece of data, but rather different set of data and how they can relate to one another in an organized manner.

2) Differentiate file structure from storage structure.

Basically, the key difference is the memory area that is being accessed. When dealing with the structure that resides the main memory of the computer system, this is referred to as storage structure. When dealing with an auxiliary structure, we refer to it as file structures.

3) When is a binary search best applied?

A binary search is an algorithm that is best applied to search a list when the elements are already in order or sorted. The list is search starting in the middle, such that if that middle value is not the target search key, it will check to see if it will continue the search on the lower half of the list or the higher half. The split and search will then continue in the same manner.

4) What is a linked list?

A linked list is a sequence of nodes in which each node is connected to the node following it. This forms a chain-like link of data storage.

5) How do you reference all the elements in a one-dimension array?

To do this, an indexed loop is used, such that the counter runs from 0 to the array size minus one. In this manner, we are able to reference all the elements in sequence by using the loop counter as the array subscript.

6) In what areas do data structures applied?

Data structures are important in almost every aspect where data is involved. In general, algorithms that involve efficient data structure is applied in the following areas: numerical analysis, operating system, A.I., compiler design, database management, graphics, and statistical analysis, to name a few.

7) What is LIFO?

LIFO is short for Last In First Out, and refers to how data is accessed, stored and retrieved. Using this scheme, data that was stored last, should be the one to be extracted first. This also means that in order to gain access to the first data, all the other data that was stored before this first data must first be retrieved and extracted.

8) What is a queue?

A queue is a data structure that can simulates a list or stream of data. In this structure, new elements are inserted at one end and existing elements are removed from the other end.

9) What are binary trees?

A binary tree is one type of data structure that has two nodes, a left node and a right node. In programming, binary trees are actually an extension of the linked list structures.

10) Which data structures is applied when dealing with a recursive function?

Recursion, which is basically a function that calls itself based on a terminating condition, makes use of the stack. Using LIFO, a call to a recursive function saves the return address so that it knows how to return to the calling function after the call terminates.

11) What is a stack?

A stack is a data structure in which only the top element can be accessed. As data is stored in the stack, each data is pushed downward, leaving the most recently added data on top.

12) Explain Binary Search Tree

A binary search tree stores data in such a way that they can be retrieved very efficiently. The left subtree contains nodes whose keys are less than the node’s key value, while the right subtree contains nodes whose keys are greater than or equal to the node’s key value. Moreover, both subtrees are also binary search trees.

13) What are multidimensional arrays?

Multidimensional arrays make use of multiple indexes to store data. It is useful when storing data that cannot be represented using a single dimensional indexing, such as data representation in a board game, tables with data stored in more than one column.

14) Are linked lists considered linear or non-linear data structures?

It actually depends on where you intend to apply linked lists. If you based it on storage, a linked list is considered non-linear. On the other hand, if you based it on access strategies, then a linked list is considered linear.

15) How does dynamic memory allocation help in managing data?

Aside from being able to store simple structured data types, dynamic memory allocation can combine separately allocated structured blocks to form composite structures that expand and contract as needed.

16) What is FIFO?

FIFO is short for First-in, First-out, and is used to represent how data is accessed in a queue. Data has been inserted into the queue list the longest is the one that is removed first.

17) What is an ordered list?

An ordered list is a list in which each node's position in the list is determined by the value of its key component, so that the key values form an increasing sequence, as the list is traversed.

18) What is merge sort?

Merge sort takes a divide-and-conquer approach to sorting data. In a sequence of data, adjacent ones are merged and sorted to create bigger sorted lists. These sorted lists are then merged again to form an even bigger sorted list, which continuous until you have one single sorted list.

19) Differentiate NULL and VOID.

Null is actually a value, whereas Void is a data type identifier. A variable that is given a Null value simply indicates an empty value. Void is used to identify pointers as having no initial size.

20) What is the primary advantage of a linked list?

A linked list is a very ideal data structure because it can be modified easily. This means that modifying a linked list works regardless of how many elements are in the list.

21) What is the difference between a PUSH and a POP?

Pushing and popping applies to the way data is stored and retrieved in a stack. A push denotes data being added to it, meaning data is being "pushed" into the stack. On the other hand, a pop denotes data retrieval, and in particular refers to the topmost data being accessed.

22) What is a linear search?

A linear search refers to the way a target key is being searched in a sequential data structure. Using this method, each element in the list is checked and compared against the target key, and is repeated until found or if the end of the list has been reached.

23) How does variable declaration affect memory allocation?

The amount of memory to be allocated or reserved would depend on the data type of the variable being declared. For example, if a variable is declared to be of integer type, then 32 bits of memory storage will be reserved for that variable.

24) What is the advantage of the heap over a stack?

Basically, the heap is more flexible than the stack. That's because memory space for the heap can be dynamically allocated and de-allocated as needed. However, memory of the heap can at times be slower when compared to that stack.

25) What is a postfix expression?

A postfix expression is an expression in which each operator follows its operands. The advantage of this form is that there is no need to group sub-expressions in parentheses or to consider operator precedence.

26) What is Data abstraction?

Data abstraction is a powerful tool for breaking down complex data problems into manageable chunks. This is applied by initially specifying the data objects involved and the operations to be performed on these data objects without being overly concerned with how the data objects will be represented and stored in memory.

27) How do you insert a new item in a binary search tree?

Assuming that the data to be inserted is a unique value (that is, not an existing entry in the tree), check first if the tree is empty. If it's empty, just insert the new item in the root node. If it's not empty, refer to the new item's key. If it's smaller than the root's key, insert it into the root's left subtree, otherwise, insert it into the root's right subtree.

28) How does a selection sort work for an array?

The selection sort is a fairly intuitive sorting algorithm,, though not necessarily efficient. To perform this, the smallest element is first located and switched with the element at subscript zero, thereby placing the smallest element in the first position. The smallest element remaining in the subarray is then located next with subscripts 1 through n-1 and switched with the element at subscript 1, thereby placing the second smallest element in the second position. The steps are repeated in the same manner till the last element.

29) How do signed and unsigned numbers affect memory?

In the case of signed numbers, the first bit is used to indicate whether positive or negative, which leaves you with one bit short. With unsigned numbers, you have all bits available for that number. The effect is best seen in the number range (unsigned 8 bit number has a range 0-255, while 8-bit signed number has a range -128 to +127.

30) What is the minimum number of nodes that a binary tree can have?

A binary tree can have a minimum of zero nodes, which occurs when the nodes have NULL values. Furthermore, a binary tree can also have 1 or 2 nodes.

31) What are dynamic data structures?

Dynamic data structures are structures that expand and contract as a program runs. It provides a flexible means of manipulating data because it can adjust according to the size of the data.

32) In what data structures are pointers applied?

Pointers that are used in linked list have various applications in data structure. Data structures that make use of this concept include the Stack, Queue, Linked List and Binary Tree.

33) Do all declaration statements result in a fixed reservation in memory?

Most declarations do, with the exemption of pointers. Pointer declaration does not allocate memory for data, but for the address of the pointer variable. Actual memory allocation for the data comes during run-time.

34) What are ARRAYS?

When dealing with arrays, data is stored and retrieved using an index that actually refers to the element number in the data sequence. This means that data can be accessed in any order. In programming, an array is declared as a variable having a number of indexed elements.

35) What is the minimum number of queues needed when implementing a priority queue?

The minimum number of queues needed in this case is two. One queue is intended for sorting priorities while the other queue is intended for actual storage of data.

36) Which sorting algorithm is considered the fastest?

There are many types of sorting algorithms: quick sort, bubble sort, balloon sort, radix sort, merge sort, etc. Not one can be considered the fastest because each algorithm is designed for a particular data structure and data set. It would depend on the data set that you would want to sort.

37) Differentiate STACK from ARRAY.

Data that is stored in a stack follows a LIFO pattern. This means that data access follows a sequence wherein the last data to be stored will the first one to be extracted. Arrays, on the other hand, does not follow a particular order and instead can be accessed by referring to the indexed element within the array.

38) Give a basic algorithm for searching a binary search tree.

1. If the tree is empty, then the target is not in the tree, end search
2. If the tree is not empty, the target is in the tree
3. Check if the target is in the root item
4. If target is not in the root item, check if target is smaller than the root’s value
5. If target is smaller than the root’s value, search the left subtree
6. Else, search the right subtree

39) What is a Dequeue?

A dequeue is a double-ended queue. This is a structure wherein elements can be inserted or removed from either end.

40) What is a bubble sort and how do you perform it?

A bubble sort is one sorting technique that can be applied to data structures such as an array. It works by comparing adjacent elements and exchanges their values if they are out of order. This method lets the smaller values “bubble” to the top of the list, while the larger value sinks to the bottom.

41) What are the parts of a linked list?

A linked list typically has two parts: the head and the tail. Between the head and tail lie the actual nodes, with each node being linked in a sequential manner.

42) How does selection sort work?

Selection sort works by picking the smallest number from the list and placing it at the front. This process is repeated for the second position towards the end of the list. It is the simplest sort algorithm.

43) What is a graph?

A graph is one type of data structure that contains a set of ordered pairs. These ordered pairs are also referred to as edges or arcs, and are used to connect nodes where data can be stored and retrieved.

44) Differentiate linear from non linear data structure.

Linear data structure is a structure wherein data elements are adjacent to each other. Examples of linear data structure include arrays, linked lists, stacks and queues. On the other hand, non-linear data structure is a structure wherein each data element can connect to more than two adjacent data elements. Examples of non linear data structure include trees and graphs.

45) What is an AVL tree?

An AVL tree is a type of binary search tree that is always in a state of partially balanced. The balance is measured as a difference between the heights of the subtrees from the root. This self-balancing tree was known to be the first data structure to be designed as such.

46) What are doubly linked lists?

Doubly linked lists are a special type of linked list wherein traversal across the data elements can be done in both directions. This is made possible by having two links in every node, one that links to the next node and other one that links to the previous node.

47) What is Huffman’s algorithm?

Huffman’s algorithm is associated in creating extended binary trees that has minimum weighted path lengths from the given weights. It makes use of a table that contains frequency of occurrence for each data element.

48) What is Fibonacci search?

Fibonacci search is a search algorithm that applies to a sorted array. It makes use of a divide-and-conquer approach that can greatly reduce the time needed in order to reach the target element.

49) Briefly explain recursive algorithm.

Recursive algorithm targets a problem by dividing it into smaller, manageable sub-problems. The output of one recursion after processing one sub-problem becomes the input to the next recursive process.

50) How do you search for a target key in a linked list?

To find the target key in a linked list, you have to apply sequential search. Each node is traversed and compared with the target key, and if it is different, then it follows the link to the next node. This traversal continues until either the target key is found or if the last node is reached.