KONGU ENGINEERING COLLEGE

(Autonomous)

PERUNDURAI, ERODE- 638 052

2021 - 2022

Transform Yourself

20CSL31

DATA STRUCTURES

LABORATORYLAB RECORD

Name	**: KAPIL K**
Roll No	**: 20CSR086**
Branch	**: COMPUTER SCIENCE AND ENGINEERING**
Semester	**: III SEMESTER**
Year	**: IIND YEAR**
Section	**: B**

KONGU

Assuring the Best

INDEX

Serial No.	Experiment Date	Name Of The Experiment	Marks	Signature of the Staff
1		Implementation of Singly Linked List and Its Operations		
2		Implementation of Doubly Linked List and Its Operations		
3		Implementation of Circular Linked List and Its Operations		
4		Implementation of Polynomial Addition using Linked List		
5		Infix to Postfix Conversion using Stack ADT		
6		Postfix Expression using Array of Stack ADT		
7		Implementation of Reversing Queue using Stack		
8		Implementation of Binary Search Tree Traversals		
9		Implementation of Bubble Sort		
10		Implementation of Graph Traversal Technique Queues		

Kongu Engineering College

(Autonomous)

Perundurai, Erode-638 060

Department of Computer Science and Engineering

20CSL31 – DATA STRUCTURES LABORATORY Laboratory Record

Name KAPIL K Programme Bachelor of Engineering

Branch Computer Science and Engineering Section B Semester III

Roll No. / Register No. 20CSR086

Certified that this is a bonafide record of work done by the above student of the

20CSL31 – DATA STRUCTURES LABORATORY ***Laboratory during the year 2021 – 2022***

Signature of the Lab-in-Charge **Signature of Head of the Department**

Submitted for the Model/End Semester Practical Examination held on _____

Internal Examiner **External Examiner**

EX.NO:01
7/9/21

IMPLEMENTATION OF SINGLY LINKED LIST AND ITS OPERATIONS.

AIM:

To write a program for implementation of singly linked list.

CODE:

```
#include <stdio.h>
#include<stdlib.h>
struct node
{
   int data;
   struct node *next;
};
struct node *head,*temp;
void insert_first();
void insert_random();
void insert_last();
void del_first();
void del_random();
void del_last();
void search();
void display();
int main()
{
   int op,option;
   head=(struct node*)malloc(sizeof(struct node));
   head->next=NULL;
   while(option!=2)
   {
      printf("Enter any option\n1.Insert at first\n2.Insert at random\n3.Insert at end\n4.Delete at
first\n5.Delete at random\n6.Delete at end\n7.search\n8.display\n");
      scanf("%d",&op);
      switch(op)
      {
         case 1:
            insert_first();
            display();
            break;
         case 2:
            insert_random();
            display();
            break;
         case 3:
            insert_last();
            display();
            break;
```

```
        case 4:
            del_first();
            display();
            break;
        case 5:
            del_random();
            display();
            break;
        case 6:
            del_last();
            display();
            break;
        case 7:
            search();
            break;
        case 8:
            display();
            break;
    }
    printf("Do you want to continue?\n1.Yes\n2.No\n");
    scanf("%d",&option);
  }
  return 0;
}
void insert_first()
{
  int data;
  struct node *new;
  new=(struct node*)malloc(sizeof(struct node));
  printf("Enter the data\n");
  scanf("%d",&data);
  new->data=data;
  new->next=NULL;
  if(head->next==NULL)
  {
    head->next=new;
  }
  else
  {
    new->next=head->next;
    head->next=new;
        printf("Data inserted successfully!!\n");
  }

}

void insert_last()
{
```

```
    int data;
    struct node *new;
    new=(struct node*)malloc(sizeof(struct node));
    printf("Enter the data\n");
    scanf("%d",&data);
    new->data=data;
    new->next=NULL;
    if(head->next==NULL)
    {
       head->next=new;
    }
    else
    {
       temp=head->next;
       while(temp->next!=NULL)
       {
          temp=temp->next;
       }
       temp->next=new;
            printf("Data inserted successfully!!\n");
    }

}
void insert_random()
{
    int data,pos,i;
    struct node *new;
    new=(struct node*)malloc(sizeof(struct node));

    printf("Enter the position where you need to insert a node\n");
    scanf("%d",&pos);
    temp=head;
    for(i=0;i<pos-1&&temp->next!=NULL;i++)
    {
       temp=temp->next;
    }
    if(temp->next!=NULL)
    {
       printf("Enter the data\n");
       scanf("%d",&data);
       new->data=data;
       new->next=NULL;
       new->next=temp->next;
       temp->next=new;
            printf("Data inserted successfully!!\n");
    }
    else
    {
```

```
      printf("Position is not available\n");

   }

}
void del_first()
{
   if(head->next==NULL)
   {
     printf("The list is empty!!\n");

   }
   temp=head->next;
   head->next=temp->next;
   free(temp);
   printf("Data Deleted successfully!!\n");
}
void del_last()
{
   if(head->next==NULL)
   {
     printf("The list is empty!!\n");

   }
   struct node *del;
   temp=head;
   while(temp->next->next!=NULL)
   {
      temp=temp->next;
   }
   del=temp->next;
   temp->next=NULL;
   free(del);
   printf("Data Deleted successfully!!\n");
}
void del_random()
{
   if(head->next==NULL)
   {
     printf("The list is empty!!\n");

   }
   int i,pos;
   struct node *del;
   printf("Enter the position where you need to delete a node\n");
   scanf("%d",&pos);
   temp=head;
   for(i=0;i<pos-1&&temp->next!=NULL;i++)
```

```
   {
      temp=temp->next;
   }
   if(temp->next!=NULL)
   {
     del=temp->next;
     temp->next=del->next;
     free(del);
printf("Data Deleted successfully!!\n");
   }
   else
   {
      printf("Position is not available\n");

   }

}
void search()
{
   int search;
   printf("Enter the data that needs to be searched\n");
   scanf("%d",&search);
   temp=head->next;
   while(temp->data!=search&&temp->next!=NULL)
   {
      temp=temp->next;
   }
   if(temp->data==search)
   {
      printf("Data is found\n");
   }
   else
   {
      printf("Data is not found\n");
   }
}
void display()
{
   temp=head->next;
```

```
   if(head->next==NULL)
   {
     printf("The list is empty!!\n");

   }
   printf("The data of list are:\n");
```

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

OUTPUT:

```
Enter any option
1.Insert at first
2.Insert at random
3.Insert at end
4.Delete at first
5.Delete at random
6.Delete at end
7.search
8.display
1
Enter the data
3
The data of list are:
3
Do you want to continue?
1.Yes
2.No
1
Enter any option
1.Insert at first
2.Insert at random
3.Insert at end
4.Delete at first
5.Delete at random
6.Delete at end
7.search
8.display
3
Enter the data
5
Data inserted successfully!!
The data of list are:
3
5
Do you want to continue?
```

```
5
Do you want to continue?
1.Yes
2.No
1
Enter any option
1.Insert at first
2.Insert at random
3.Insert at end
4.Delete at first
5.Delete at random
6.Delete at end
7.search
8.display
2
Enter the position where you need to insert a node
2
Enter the data
6
Data inserted successfully!!
The data of list are:
3
6
5
Do you want to continue?
1.Yes
2.No
1
Enter any option
1.Insert at first
2.Insert at random
3.Insert at end
4.Delete at first
5.Delete at random
6.Delete at end
```

```
5.Delete at random
6.Delete at end
7.search
8.display
7
Enter the data that needs to be searched
9
Data is not found
Do you want to continue?
1.Yes
2.No
1
Enter any option
1.Insert at first
2.Insert at random
3.Insert at end
4.Delete at first
5.Delete at random
6.Delete at end
7.search
8.display
8
The data of list are:
3
6
5
Do you want to continue?
1.Yes
2.No
1
Enter any option
1.Insert at first
2.Insert at random
3.Insert at end
4.Delete at first
```

```
3.Insert at end
4.Delete at first
5.Delete at random
6.Delete at end
7.search
8.display
4
Data Deleted successfully!!
The data of list are:
6
5
Do you want to continue?
1.Yes
2.No
1
Enter any option
1.Insert at first
2.Insert at random
3.Insert at end
4.Delete at first
5.Delete at random
6.Delete at end
7.search
8.display
6
Data Deleted successfully!!
The data of list are:
6
Do you want to continue?
1.Yes
2.No
2

...Program finished with exit code 0
```

RESULT:

Thus a program for implementing singly linked list is done successfully.

EX.NO:02

14/9/21

IMPLEMENTATION OF DOUBLY LINKED LIST AND ITS OPERATIONS.

AIM:

To write a program for implementing doubly linked list and its operations.

CODE:

```
#include <stdio.h>
#include<stdlib.h>
struct node
{
   struct node *prev;
```

```
   int data;
   struct node *next;
};
struct node *head,*temp;
void insert_first();
void insert_random();
void insert_last();
void del_first();
void del_random();
void del_last();
void search();
void display();
int main()
{
   int op,option;
   while(option!=2)
   {
      printf("Enter any option\n1.Insert at first\n2.Insert at random\n3.Insert at end\n4.Delete at
first\n5.Delete at random\n6.Delete at end\n7.search\n8.display\n");
      scanf("%d",&op);
      switch(op)
      {
         case 1:
            insert_first();
            display();
            break;
         case 2:
            insert_random();
            display();
            break;
         case 3:
            insert_last();
            display();
            break;
         case 4:
            del_first();
            display();
            break;
         case 5:
            del_random();
            display();
            break;
         case 6:
            del_last();
            display();
            break;
         case 7:
            search();
```

```
            break;
        case 8:
            display();
            break;
      }
      printf("Do you want to continue?\n1.Yes\n2.No\n");
      scanf("%d",&option);
   }
   return 0;
}
void insert_first()
{
   int data;
   struct node *new;
   new=(struct node*)malloc(sizeof(struct node));
   printf("Enter the data\n");
   scanf("%d",&data);
   new->data=data;
   new->prev=NULL;
   new->next=NULL;
   if(head==NULL)
   {
     head=new;
   }
   else
   {
      temp=head;
      new->next=temp;
      temp->prev=new;
      head=new;
             printf("Data inserted successfully!!\n");
   }

}

void insert_last()
{
   int data;
   struct node *new;
   new=(struct node*)malloc(sizeof(struct node));
   printf("Enter the data\n");
   scanf("%d",&data);
   new->prev=NULL;
   new->data=data;
   new->next=NULL;
   if(head==NULL)
   {
      head=new;
```

```
    }
    else
    {
        temp=head;
        while(temp->next!=NULL)
        {
            temp=temp->next;
        }
        temp->next=new;
        new->prev=temp;
                printf("Data inserted successfully!!\n");
    }

}
void insert_random()
{
    int data,pos,i;
    struct node *new;
    new=(struct node*)malloc(sizeof(struct node));
    printf("Enter the position where you need to insert a node\n");
    scanf("%d",&pos);
    temp=head;
    for(i=1;i<pos-1&&temp->next!=NULL;i++)
    {
        temp=temp->next;
    }
    if(temp->next!=NULL)
    {
        printf("Enter the data\n");
        scanf("%d",&data);
        new->data=data;
        new->next=NULL;
        new->prev=NULL;
        temp->next->prev=new;
        new->next=temp->next;
        temp->next=new;
                printf("Data inserted successfully!!\n");
    }
    else
    {
        printf("Position is not available\n");

    }

}
void del_first()
{
    if(head==NULL)
```

```
    {
      printf("The list is empty!!\n");
    }
    temp=head;
    head=temp->next;
    head->prev=NULL;
    temp->next=NULL;
    free(temp);
    printf("Data Deleted successfully!!\n");
}
void del_last()
{
    if(head==NULL)
    {
      printf("The list is empty!!\n");
    }
    struct node *del;
    temp=head;
    while(temp->next->next!=NULL)
    {
       temp=temp->next;
    }
    del=temp->next;
    temp->next=NULL;
    del->prev=NULL;
    free(del);
    printf("Data Deleted successfully!!\n");
}
void del_random()
{
    if(head==NULL)
    {
      printf("The list is empty!!\n");
    }
    int i,pos;
    struct node *del;
    printf("Enter the position where you need to delete a node\n");
    scanf("%d",&pos);
    temp=head;
    for(i=1;i<pos-1&&temp->next!=NULL;i++)
    {
       temp=temp->next;
    }
    if(temp->next!=NULL)
    {
       del=temp->next;
       temp->next=del->next;
       del->prev=temp;
```

```
      free(del);
      printf("Data Deleted successfully!!\n");
   }
   else
   {
      printf("Position is not available\n");
   }

}
void search()
{
   int ele,i=1;
   printf("Enter the data that needs to be searched\n");
   scanf("%d",&ele);
   temp=head;
   while(temp->data!=ele&&temp->next!=NULL)
   {
      temp=temp->next;
      i++;
   }
   if(temp->data==ele)
   {
      printf("Data is found\n");
      printf("The position of %d is %d\n",ele,i);
   }
   else
   {
      printf("Data is not found\n");
   }
}
void display()
{
   temp=head;
   if(head==NULL)
   {
     printf("The list is empty!!\n");

   }
   printf("The data of list are:\n");
   while(temp!=NULL)
   {
      printf("%d\n",temp->data);
      temp=temp->next;
   }
}
```

OUTPUT:

```
nter any option
.Insert at first
.Insert at random
.Insert at end
.Delete at first
.Delete at random
.Delete at end
.search
.display

nter the data

he data of list are:

o you want to continue?
.Yes
.No

nter any option
.Insert at first
.Insert at random
.Insert at end
.Delete at first
.Delete at random
.Delete at end
.search
.display

nter the data

ata inserted successfully!!
he data of list are:

o you want to continue?
```

33°C Light rain 2:10 PM

```
Do you want to continue?
1.Yes
2.No
1
Enter any option
1.Insert at first
2.Insert at random
3.Insert at end
4.Delete at first
5.Delete at random
6.Delete at end
7.search
8.display
2
Enter the position where you need to insert a node
2
Enter the data
6
Data inserted successfully!!
The data of list are:
3
6
5
Do you want to continue?
1.Yes
2.No
1
Enter any option
1.Insert at first
2.Insert at random
3.Insert at end
4.Delete at first
5.Delete at random
6.Delete at end
```

33°C Light rain 2:10 PM

```
5.Delete at random
6.Delete at end
7.search
8.display
7
Enter the data that needs to be searched
9
Data is not found
Do you want to continue?
1.Yes
2.No
1
Enter any option
1.Insert at first
2.Insert at random
3.Insert at end
4.Delete at first
5.Delete at random
6.Delete at end
7.search
8.display
8
The data of list are:
8
6
5
Do you want to continue?
1.Yes
2.No
1
Enter any option
1.Insert at first
2.Insert at random
3.Insert at end
4.Delete at first
```

```
3.Insert at end
4.Delete at first
5.Delete at random
6.Delete at end
7.search
8.display
4
Data Deleted successfully!!
The data of list are:
6
5
Do you want to continue?
1.Yes
2.No
1
Enter any option
1.Insert at first
2.Insert at random
3.Insert at end
4.Delete at first
5.Delete at random
6.Delete at end
7.search
8.display
6
Data Deleted successfully!!
The data of list are:
6
Do you want to continue?
1.Yes
2.No
2

...Program finished with exit code 0
```

RESULT:

Thus a program for doubly linked list is done successfully.

EX.NO:03
16/9/21

IMPLEMENTATION OF CIRCULAR LINKED LIST AND ITS OPERATIONS.

AIM:

TO WRITE A PROGRAM FOR CIRCULAR LINKED LIST AND ITS OPERATIONS.

CODE:

```
#include<stdio.h>
#include<stdlib.h>
struct node
{
int data;
struct node *next;
};
struct node *head,*temp,*p;
int j=0;
void insert_first()
{
int data;
struct node*new;
new=malloc(sizeof(struct node*));
printf("enter the data of new node:");
scanf("%d",&data);
new->data=data;
new->next=NULL;
if(head->next==NULL)
{
head->next=new;
new->next=new;
j++;
printf("Data inserted successfully!!\n");
}
else
{
do
{
temp=temp->next;
}
while(temp->next!=head->next);
new->next=head->next;
head->next=new;
temp->next=new;
j++;
printf("Data inserted successfully!!\n");
}
}
void insert_random()
{
int data,pos,i;
```

```
struct node*new;
new=malloc(sizeof(struct node*));
if(head->next==NULL)
{
printf("The list is empty\n");
}
else
{
printf("enter the position of new node:");
scanf("%d",&pos);
if(pos==1||pos==j-1)
{
printf("Enter postion at middle\n");
}
else
{
temp=head->next;
for(int i=1;i<pos-1&&temp->next!=head->next;i++)
{
temp=temp->next;
}
if(temp->next!=head->next)
{
printf("enter the data of new node:");
scanf("%d",&data);
new->data=data;
new->next=temp->next;
temp->next=new;
j++;
printf("Data inserted successfully!!\n");
}
else
{
printf("position not available\n");
}
}
}
}
void insert_last()
{
int data;
struct node*new;
new=malloc(sizeof(struct node*));
printf("enter the data of new node:");
scanf("%d",&data);
new->data=data;
new->next=NULL;
if(head->next==NULL)
```

```
{
j++;
head->next=new;
new->next=new;
}
else
{
temp=head->next;
while(temp->next!=head->next)
{
temp=temp->next;
}
temp->next=new;
new->next=head->next;
j++;
printf("Data inserted successfully!!\n");
}
}
void delete_first()
{
struct node *p;
if(head->next==NULL)
{
printf("The list is empty\n");
}
else
{
temp=head->next;
while(temp->next!=head->next)
{
temp=temp->next;
}
temp->next=head->next->next;
p=head->next;
head->next=p->next;
free(p);
j--;
printf("Data deleted successfully!!\n");
}
}
void delete_random()
{
int pos;
struct node *del;
printf("enter the position to delete the node\n");
scanf("%d",&pos);
if(pos==1||pos==j)
{
```

```
printf("Enter postion at middle\n");
}
else
{
temp=head->next;
for(int i=1;i<pos-1&&temp->next!=head->next;i++)
{
temp=temp->next;
}
if(temp->next!=head->next)
{
del=temp->next;
temp->next=del->next;
del->next=NULL;
free(del);
j--;
printf("Data deleted successfully!!\n");
}
else
{
printf("position not available\n");
}
}
}
void delete_end()
{
if(head->next==NULL)
{
printf("The list is empty\n");
}
else
{
temp=head;
while(temp->next->next!=head->next)
{
temp=temp->next;
}
p=temp->next;
temp->next=head->next;
free(p);
p->next=NULL;
j--;
printf("Data deleted successfully!!\n");
}
}
void search()
{
int s;
```

```
printf("Enter the data for searching:");
scanf("%d",&s);
temp=head->next;
while(temp->data!=s && temp->next!=head->next)
{
   temp=temp->next;
}
if(temp->data==s)
{
   printf("Data is found\n");
}
else
{
   printf("Data not found\n");
}
}
void display()
{
if(head->next==NULL)
{
 printf("The list is empty!!\n");
}
else
{
temp=head->next;
printf("The elements are:\n");
do
{
printf("%d\n",temp->data);
temp=temp->next;
}
while(temp!=head->next);
}
}
void data()
{
   int pos,data;
   struct node *new;
   new=(struct node*)malloc(sizeof(struct node));
   printf("Enter the position");
   scanf("%d",&pos);
   temp=head->next;
   for(int i=1;i<pos;i++)
   {
      temp=temp->next;
   }
   new->data=(temp->data)*2;
   temp=head->next;
```

```
    while(temp->next!=head->next)
    {
    temp=temp->next;
    }
    temp->next=new;
    new->next=head->next;
    j++;
    printf("Data inserted successfully!!\n");
}
int main()
{
int opt,op;
head=malloc(sizeof(struct node*));
head->next=NULL;
while(op!=2)
{
printf("1.Insert at first\n2.Insert at random\n3.Insert at end\n4.Delete at first\n5.Delete at
random\n6.Delete at end\n7.search\n8.Display\n9.data\n");
printf("Enter any option:");
scanf("%d",&opt);
switch(opt)
{
case 1:
insert_first();
display();
break;
case 2:
insert_random();
display();
break;
case 3:
insert_last();
display();
break;
case 4:
delete_first();
display();
break;
case 5:
delete_random();
display();
break;
case 6:
delete_end();
display();
break;
case 7:
search();
```

```
break;
case 8:
display();
break;
case 9:
data();
display();
break;
}
printf("Do you want to continue?\n1.yes\n2.no\n");
scanf("%d",&op);
}
return 0;
}
```

OUTPUT:

```
1.Insert at first
2.Insert at random
3.Insert at end
4.Delete at first
5.Delete at random
6.Delete at end
7.search
8.Display
9.data
Enter any option:1
enter the data of new node:1
Data inserted successfully!!
The elements are:
1
Do you want to continue?
1.yes
2.no
1
1.Insert at first
2.Insert at random
3.Insert at end
4.Delete at first
5.Delete at random
6.Delete at end
7.search
8.Display
9.data
Enter any option:3
enter the data of new node:3
Data inserted successfully!!
The elements are:
1
3
Do you want to continue?
1.yes
```

```
1.yes
2.no
1
1.Insert at first
2.Insert at random
3.Insert at end
4.Delete at first
5.Delete at random
6.Delete at end
7.search
8.Display
9.data
Enter any option:2
enter the position of new node:2
enter the data of new node:2
Data inserted successfully!!
The elements are:
1
2
3
Do you want to continue?
1.yes
2.no
1
1.Insert at first
2.Insert at random
3.Insert at end
4.Delete at first
5.Delete at random
6.Delete at end
7.search
8.Display
9.data
Enter any option:3
enter the data of new node:4
```

```
Enter any option:3
enter the data of new node:4
Data inserted successfully!!
The elements are:
1
2
3
4
Do you want to continue?
1.yes
2.no
1
1.Insert at first
2.Insert at random
3.Insert at end
4.Delete at first
5.Delete at random
6.Delete at end
7.search
8.Display
9.data
Enter any option:5
enter the position to delete the node
2
Data deleted successfully!!
The elements are:
1
3
4
Do you want to continue?
1.yes
2.no
1
1.Insert at first
2.Insert at random
```

33°C Light rain 3:23 PM

```
1.Insert at first
2.Insert at random
3.Insert at end
4.Delete at first
5.Delete at random
6.Delete at end
7.search
8.Display
9.data
Enter any option:4
Data deleted successfully!!
The elements are:
3
4
Do you want to continue?
1.yes
2.no
1
1.Insert at first
2.Insert at random
3.Insert at end
4.Delete at first
5.Delete at random
6.Delete at end
7.search
8.Display
9.data
Enter any option:6
Data deleted successfully!!
The elements are:
3
Do you want to continue?
1.yes
2.no
1
```

33°C Light rain 3:23 PM

```
3.Insert at end
4.Delete at first
5.Delete at random
6.Delete at end
7.search
8.Display
9.data
Enter any option:6
Data deleted successfully!!
The elements are:
3
Do you want to continue?
1.yes
2.no
1
1.Insert at first
2.Insert at random
3.Insert at end
4.Delete at first
5.Delete at random
6.Delete at end
7.search
8.Display
9.data
Enter any option:7
Enter the data for searching:8
Data not found
Do you want to continue?
1.yes
2.no
2


...Program finished with exit code 0
Press ENTER to exit console.
```

RESULT:

Thus a program for circular linked list is done successfully.

EX.NO:04
21/09/21

IMPLEMENTATION OF POLYNOMIAL ADDITION USING LINKED LIST.

AIM:

To write a program for implementation of polynomial addition using linked list.

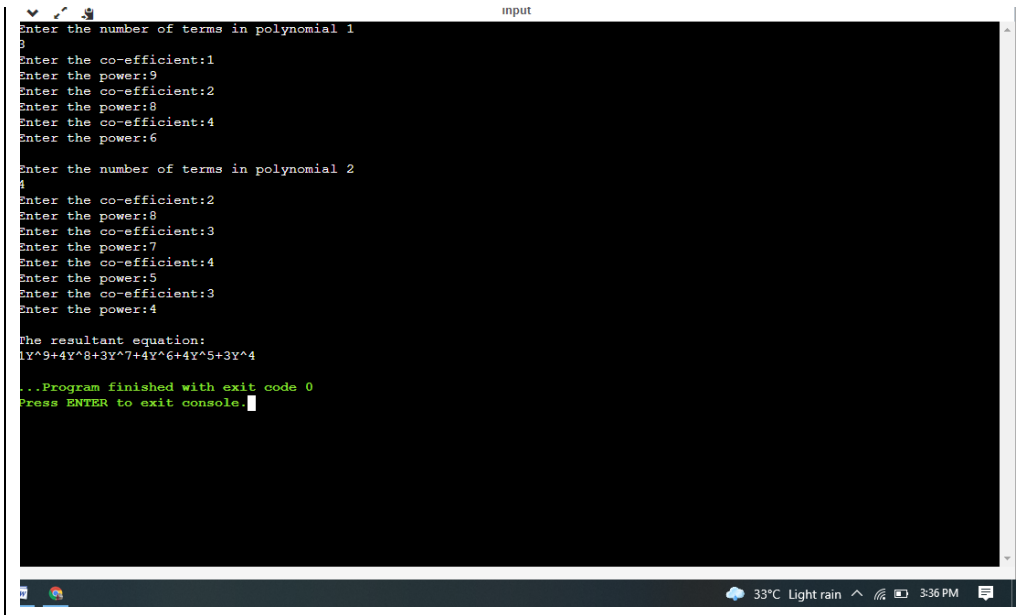
CODE:

```
#include<stdio.h>
#include<stdlib.h>
struct node
{
   int coef;
   int powe;
   struct node *next;
};
struct node *head1,*head2,*temp,*temp1,*temp2,*tempn,*new,*nhead,*tempr;
struct node *get();
int main()
{
   printf("Enter the polynomial 1\n");
   head1=get();
   printf("Enter the polynomial 2\n");
   head2=get();
   temp1=head1;
   temp2=head2;
   while(temp1!=NULL&&temp2!=NULL)
   {
      new=(struct node*)malloc(sizeof(struct node));
      if(temp1->powe>temp2->powe)
      {
         new->powe=temp1->powe;
         new->coef=temp1->coef;
         new->next=NULL;
         temp1=temp1->next;
      }
      else if(temp1->powe<temp2->powe)
      {
         new->powe=temp2->powe;
         new->coef=temp2->coef;
         new->next=NULL;
         temp2=temp2->next;
      }
      else
      {
         new->powe=temp1->powe;
```

```
        new->coef=temp1->coef+temp2->coef;
        new->next=NULL;
        temp1=temp1->next;
        temp2=temp2->next;
    }
    if(nhead==NULL)
    {
        nhead=new;
        tempn=new;
    }
    else
    {
        tempn->next=new;
        tempn=tempn->next;
    }
}
if(temp1==NULL)
{
    while(temp2!=NULL)
    {
        new=(struct node*)malloc(sizeof(struct node));
        new->powe=temp2->powe;
        new->coef=temp2->coef;
        new->next=NULL;
        temp2=temp2->next;
        tempn->next=new;
        tempn=tempn->next;
    }
}
else if(temp2==NULL)
{
    while(temp1!=NULL)
    {
        new=(struct node*)malloc(sizeof(struct node));
        new->powe=temp1->powe;
        new->coef=temp1->coef;
        new->next=NULL;
        temp1=temp1->next;
        tempn->next=new;
        tempn=tempn->next;
    }
}
tempr=nhead;
printf("The resultant equation:\n");
while(tempr->next!=NULL)
{
    printf("(%d)x^%d+",tempr->coef,tempr->powe);
    tempr=tempr->next;
```

```
    }
    printf("(%d)x^%d",tempr->coef,tempr->powe);
}



struct node* get()
{
struct node *dhead=NULL;
int n,co,po,i;
printf("Enter the number of terms\n");
scanf("%d",&n);
for(i=0;i<n;i++)
{
struct node *new=malloc(sizeof(struct node));
printf("Enter the co-efficient:");
scanf("%d",&co);
printf("Enter the power:");
scanf("%d",&po);
new->coef=co;
new->powe=po;
new->next=NULL;
if(dhead==NULL)
{
dhead=new;
temp=dhead;
}
else
{
temp->next=new;
temp=temp->next;
}

}
return dhead;
}
```

OUTPUT:



RESULT:

Thus a program for implementation of polynomial addition is done successfully.

EX.NO:05
23/9/21

INFIX TO POSTFIX CONVERSTION USING STACK ADT

AIM:

To write a program for infix to postfix converstion.

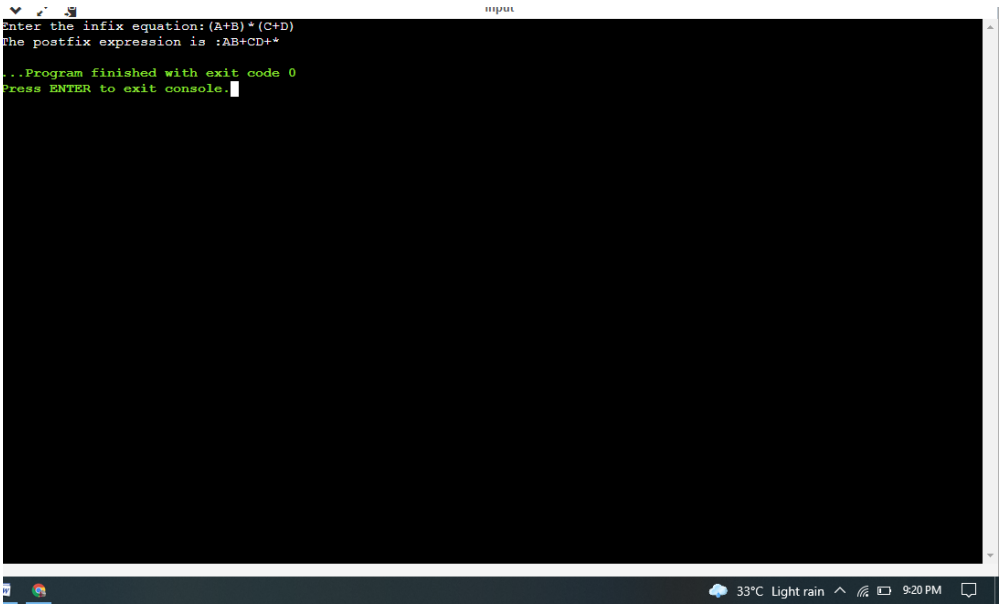
CODE:

```
#include<stdio.h>
#include<string.h>
char stack[25];
int top=-1;
void push(char a)
{
   top++;
   stack[top]=a;
}

void pop()
{
   if(stack[top]=='(')
   {
      top--;
   }
   else
   {
      printf("%c",stack[top]);
      top--;
   }
}
int pre(char c)
{
   switch(c)
   {
      case '^':
         return 3;
         break;
      case '%':
                return 2;
                break;
      case '*':
         return 2;
         break;
      case '/':
```

```
        return 2;
        break;
      case '+':
        return 1;
        break;
      case '-':
        return 1;
        break;
    }
}

int main()
{
    char arr[20];
    int i,size=0;
    printf("Enter the infix equation:");
    scanf("%s",arr);
    size=strlen(arr);
    printf("The postfix expression is :");
    for(i=0;i<size;i++)
    {
      if((arr[i]>='a' && arr[i]<='z') || (arr[i]>='A' && arr[i]<='Z'))
      {
        printf("%c",arr[i]);
      }
      else
      {
        if(arr[i]=='(')
        {
          push(arr[i]);
        }
        else if(arr[i]==')')
        {
          while(stack[top]!='(')
          {
            pop();
          }
          pop();
        }
        else if(top==-1)
        {
          push(arr[i]);
        }
        else if(stack[top]=='(')
        {
          push(arr[i]);
        }
        else if(pre(stack[top])<pre(arr[i]))
```

```
        {
            push(arr[i]);
        }
        else
        {
            while(pre(arr[i])<=pre(stack[top])&&top!=-1)
            {
                pop();
            }
            push(arr[i]);
        }
    }
  }
  while(top!=-1)
  {
    pop();
  }
}
```

OUTPUT:



RESULT:

Thus program for infix to postfix conversion is done successfully.

EX.NO:06
23/9/21

IMPLEMENT THE APPLICATION FOR EVALUATING POSTFIX EXPRESSIONS USING ARRAY OF STACK ADT

AIM:

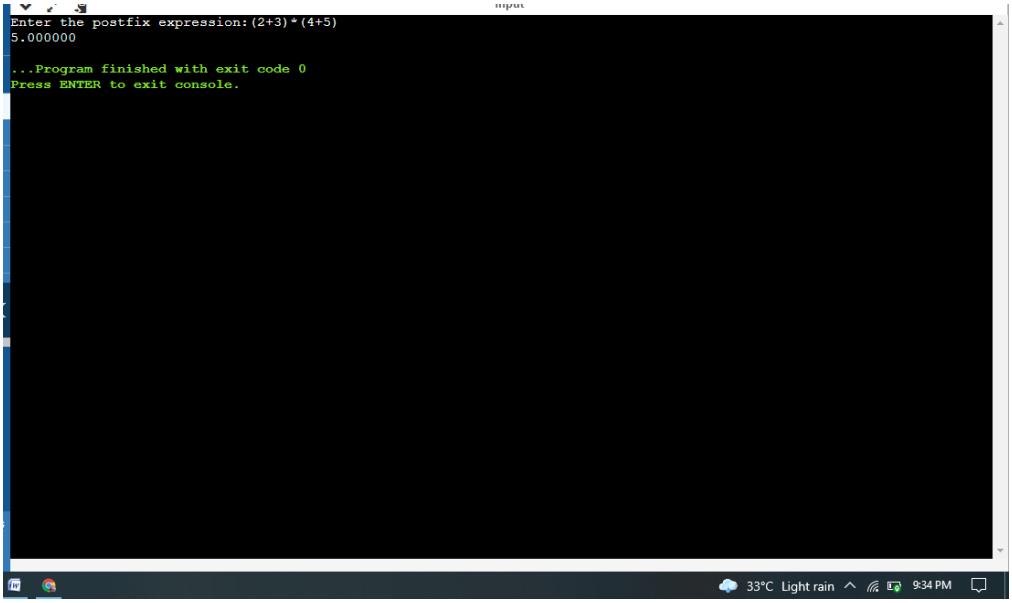
To write a program for evaluating a postfix expression.

CODE:

```
#include<stdio.h>
#include<string.h>
#include<math.h>
#include<ctype.h>
float stack[20];
int top=-1;
void push(float a)
{
   top++;
   stack[top]=a;
}
float op(char c)
{
   float res;
   switch(c)
   {
      case '+':
         res=stack[top-1]+stack[top];
         return res;
         break;
      case '-':
         res=stack[top-1]-stack[top];
         return res;
         break;
      case '*':
         res=stack[top-1]*stack[top];
         return res;
         break;
      case '/':
         res=stack[top-1]/stack[top];
         return res;
         break;
      case '^':
         res=pow(stack[top-1],stack[top]);
         return res;
         break;
   }
}
int main()
{
   char arr[20];
```

```
    int size,i,m;
    float result;
    printf("Enter the postfix expression:");
    scanf("%s",arr);
    size=strlen(arr);
    for(i=0;i<size;i++)
    {
        if(isalnum(arr[i]))
        {
            m=arr[i]-'0';
            push(m);
        }
        else
        {
            result=op(arr[i]);
            top=top-2;
            push(result);
        }
    }
    printf("%f",stack[top]);
    top--;
}
```

OUTPUT:



RESULT:

Thus a program for postfix evaluation is done successfully.

EX.NO:07
7/10/21

IMPLEMTATION OF REVERSING A QUEUE USING STACK

AIM:

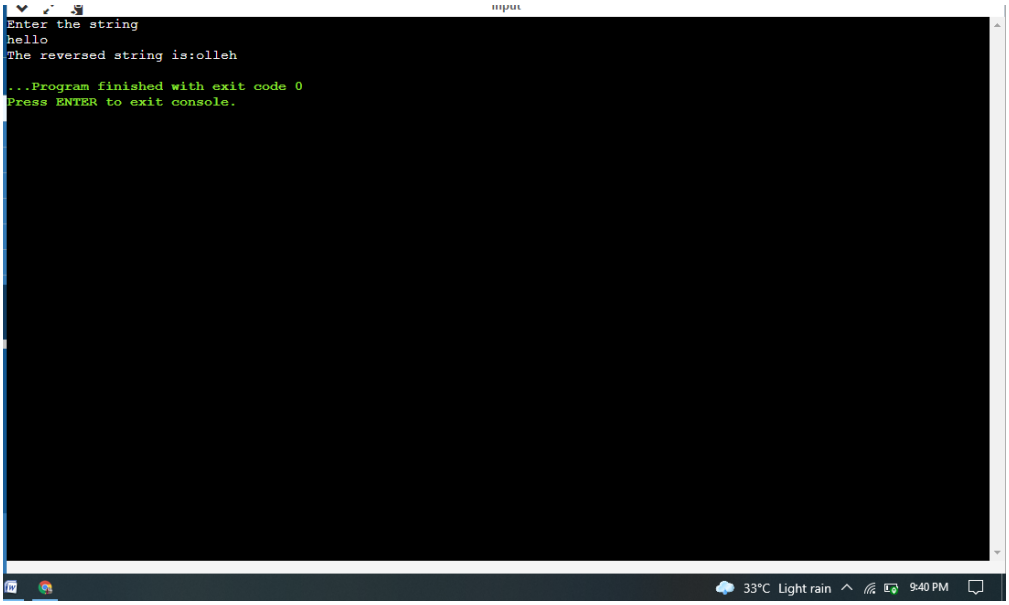
To write a program for reversing a queue using stack.

CODE:

```
#include<stdio.h>
#include<string.h>
int top=-1;
int front=-1,rear=-1;
char queue[20];
char stack[20];
void enqueue(char c)
{

   if(front==-1&&rear==-1)
   {
      front=rear=0;
      queue[front]=c;
   }
   else
   {
      rear++;
      queue[rear]=c;
   }
}
void dequeue()
{
   if(top==-1)
   {
      top=0;
      stack[top]=queue[front];
      front++;
   }
   else
   {
      top++;
      stack[top]=queue[front];
      front++;
   }
   if(front==rear)
   {
      top++;
      stack[top]=queue[front];
      front=rear=-1;
   }
}
void pop()
```

```
{
   if(front==-1&&rear==-1)
   {
      front=rear=0;
      queue[front]=stack[top];
      top--;
   }
   else
   {
      rear++;
      queue[rear]=stack[top];
      top--;
   }
}
int main()
{
   int i,size;
   char arr[20];
   printf("Enter the string\n");
   scanf("%s",arr);
   size = strlen(arr);
   for(i=0;i<size;i++)
   {
      enqueue(arr[i]);
   }
   for(i=0;i<size;i++)
   {
      dequeue();
   }
   while(top>=0)
   {
      pop();
   }
   printf("The reversed string is:");
   for(i=front;i<=rear;i++)
   {
      printf("%c",queue[i]);
   }
   return 0;
}
```

OUTPUT:



RESULT:

Thus a program for reversing a queue using stack is done successfully.

EX.NO:08
7/10/21

IMPLEMENTATION OF BINARY SEARCH TRAVERSALS

AIM:

To write a program for implementing binary search traversals.

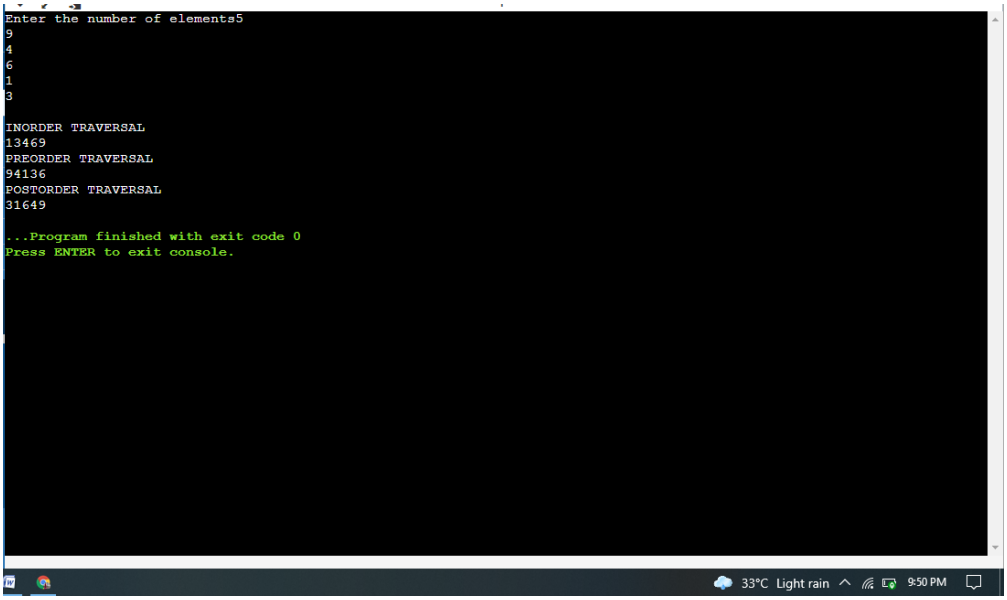
CODE:

```
#include<stdio.h>
#include<stdlib.h>
struct node
{
   struct node *left;
   int data;
   struct node *right;
};
struct node *root,*temp;
struct node* tree(struct node *temp,int ele)//insertion in binary search tree
{
   struct node *new;
   new=malloc(sizeof(struct node*));
   new->left=NULL;
   new->data=ele;
   new->right=NULL;
   if(temp==NULL)
   {
      return new;
   }
   else
   {
      if(new->data>temp->data)
      {
         temp->right=tree(temp->right,ele);
      }
      else if(new->data<temp->data)
      {
         temp->left=tree(temp->left,ele);
      }
   }
   return temp;
}
void inorder(struct node *temp1)
{
   if(temp1!=NULL)
   {
```

```
        inorder(temp1->left);
        printf("%d",temp1->data);
        inorder(temp1->right);
    }

}
void preorder(struct node *temp1)
{
     if(temp1!=NULL)
    {
        printf("%d",temp1->data);
        preorder(temp1->left);
        preorder(temp1->right);
    }

}
void postorder(struct node *temp1)
{
     if(temp1!=NULL)
    {
        postorder(temp1->left);
        postorder(temp1->right);
        printf("%d",temp1->data);
    }

}
void main()
{
    int n,i,ref;
    printf("Enter the number of elements");
    scanf("%d",&n);
    int arr[n];
    for(i=0;i<n;i++)
    {
        scanf("%d",&arr[i]);
    }
    root=NULL;
    for(i=0;i<n;i++)
    {
        ref=arr[i];
        root=tree(root,ref);
    }
    printf("\nINORDER TRAVERSAL\n");
   inorder(root);
     printf("\nPREORDER TRAVERSAL\n");
    preorder(root);
    printf("\nPOSTORDER TRAVERSAL\n");
    postorder(root);
```

}

OUTPUT:



```
Enter the number of elements5
9
4
6
1
3

INORDER TRAVERSAL
13469
PREORDER TRAVERSAL
94136
POSTORDER TRAVERSAL
31649

...Program finished with exit code 0
Press ENTER to exit console.
```

RESULT:

Thus a program for binary search traversal is done successfully.

EX.NO:09

BUBBLE SORTING

AIM:

To write a program for implementing bubble sorting.

CODE:

```
#include<stdio.h>
void main()
{
   int n,i,j,t;
   printf("Enter the number of elements:");
   scanf("%d",&n);
   int arr[n];
   printf("Enter the elements");
   for(i=0;i<n;i++)
   {
      scanf("%d",&arr[i]);
   }
   for(i=0;i<n;i++)
   {
      for(j=0;j<n-1-i;j++)
      {
         if(arr[j]>arr[j+1])
         {
            t=arr[j];
            arr[j]=arr[j+1];
            arr[j+1]=t;
         }
      }
   }
   for(i=0;i<n;i++)
   {
      printf("%d",arr[i]);
   }
}
```

OUTPUT:

```
Enter the number of elements:6
Enter the elements 9
4
1
3
6
2
123469

...Program finished with exit code 0
Press ENTER to exit console.
```

RESULT:

Thus a program for implementing bubble sort is done successfully.

EX.NO:10 (A)

AIM:

To write program to implement graph traversals.

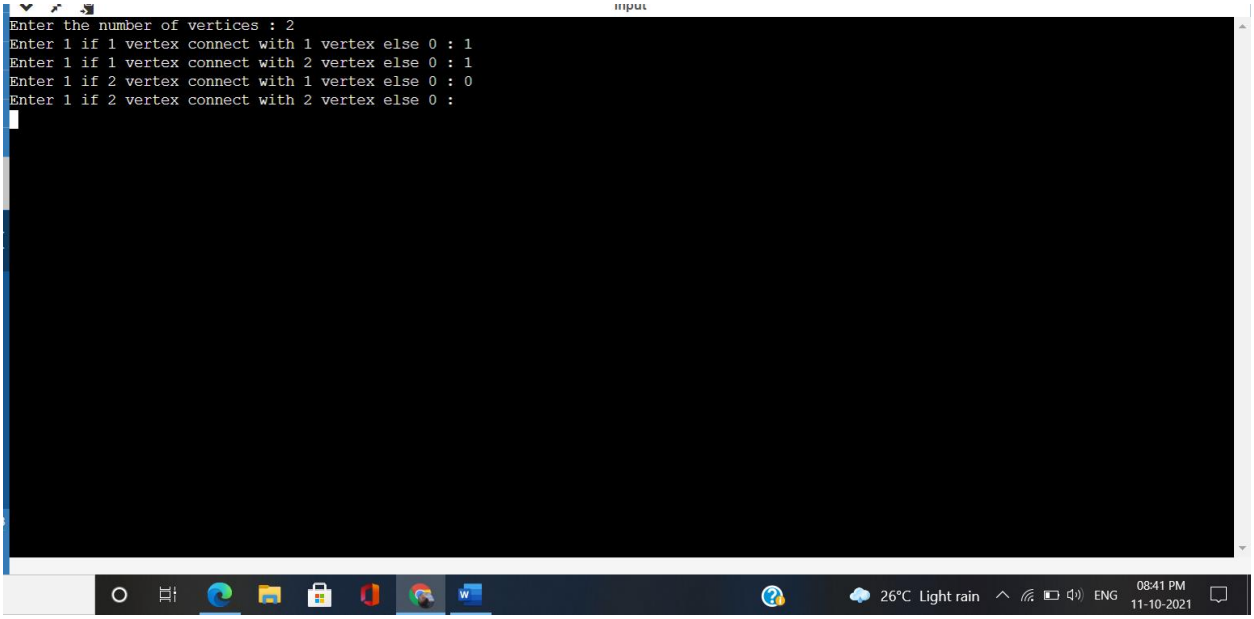
CODE:

```
#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,root,ch,j;
 char c,dummy;
 printf("Enter the number of vertices : ");
 scanf("%d",&n);
 for(i=1;i<=n;i++)
 {
 for(j=1;j<=n;j++)
 {
 printf("Enter 1 if %d vertex connect with %d vertex else 0 : ",i,j);
 scanf("%d",&a[i][j]);
 }
 }
 printf("Display of adjacency matrix :\n");
 for(i=1;i<=n;i++)
 {
 for(j=1;j<=n;j++)
 {
 printf("\t%d",a[i][j]);
 }
 printf("\n");
 }
 do
 {
 for(i=1;i<=n;i++)
 vis[i]=0;
 printf("\n1.Breadth First Search");
 printf("\n2.Depth First Search");
 printf("\nEnter your choice :");
 scanf("%d",&ch);
 printf("Enter the root to start visit :");
```

```
 scanf("%d",&root);
 switch(ch)
 {
 case 1:
 bfs(root,n);
 break;

 case 2:
 dfs(root,n);
 break;
 }
 printf("Enter \"y\" to continue and \"n\" to stop");
 scanf("%c",&dummy);
 scanf("%c",&c);
 }while((c=='y')||(c=='Y'));
}
//BFS
void bfs(int root,int n)
{
 int p,i;
 add(root);
 vis[root]=1;
 p=delete();
 if(p!=0)
 printf("\t%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 is 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
void dfs(int root,int n)
{
 int i,k;
 push(root);
 vis[root]=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);
 }
}
```



RESULT:

THUS THE PROGRAM HAS BEEN COMPILED AND EXECUTED SUCCESSFULY