PROGRAM

```
#include<stdio.h>
#include<conio.h>
void main()
{
int a[50],i,n,x,flag=0;
clrscr();
printf("Enter the value of n:");
scanf("%d",&n);
printf("Enter the numbers:");
for(i=1;i<=n;i++)
scanf("%d",&a[i]);
printf("Enter the number to be searched:");
scanf("%d",&x);
for(i=1;i<=n;i++)
{
if(x==a[i])
{
printf("The number %d is found in %d position",x,i);
flag=1;
break;
}
}
if(flag==0)
printf("The number is not found");
getch();
}
```

OUTPUT:

```
Enter the value of n : 6
Enter the number :
20
15
28
31
56
18
Enter the number to be searched : 15
The number is found in 2 position
Enter the number to be searched: 12
The number is not found
```

PROGRAM

```
#include<stdio.h>
#include<conio.h>
void main()
{
int a[100],m,n,i,j,temp;
void binary(int a[100],int m,int n);
clrscr();
printf("Enter the size of array");
scanf("%d",&m);
printf("Enter array element");
for(i=1;i<=m;i++)
{
scanf("%d",&a[i]);
}
for(i=1;i<=m;i++)
{
for(j=i+1;j<=m;j++)
{
if(a[i]>a[j])
{
temp=a[i];
a[i]=a[j];
a[j]=temp;
}
}
}
printf("Sorted array");
for(i=1;i<=m;i++)
{
printf("\n%d",a[i]);
}
printf("\nEnter the search element");
scanf("%d",&n);
binary(a,m,n);
getch();
}
void binary(int a[100],int m,int n)
{
int low=1,high=m,mid,flag=0;
while(low<=high&&flag==0)
{
mid=(low+high)/2;
if(n==a[mid])
```

```
{
flag=1;
printf("The number %d is found in:%d",n,mid);
}
else if(n>a[mid])
{
low=mid+1;
}
else
{
high=mid-1;
}
}
if(flag==0)
{
printf("The number is not found");
}
}
```

OUTPUT:

```
Enter the size of array : 5
Enter array element :
31
15
20
18
10
Sorted array :
10
15
18
20
31
Enter the search element: 18
The number 18 is found in 3

Enter the search element: 9
The number is not found
```

PROGRAM

```
#include<stdio.h>
#include<conio.h>
int n;
void main()
{
int i,a[10];
void insertion(int a[10]);

clrscr();
printf("Enter the number of terms:");
scanf("%d",&n);
printf("Enter the number:");
for(i=0;i<n;i++)
scanf("%d",&a[i]);
insertion(a);
getch();
}
void insertion(int a[10])
{
int i,j,temp;
for(i=0;i<=n-1;i++)
{
temp=a[i];
j=i-1;
while ((j>=0)&&(a[j]>temp))
{
a[j+1]=a[j];
j=j-1;
}
a[j+1]=temp;
}
printf("The sorted elements are:");
for(i=0;i<n;i++)
printf("\n%d",a[i]);
}
```

OUTPUT

```
Enter the number of elements : 4
Enter the elements :
23
45
56
32
The sorted elements are :
23
32
45
56
```

PROGRAM:

```
#include<stdio.h>
#include<conio.h>
void main()
{
int a[20],min,k=0,n,i,j;
clrscr();
printf("\nEnter the number of elements:");
scanf("%d",&n);
printf("Enter the number:");
for(i=1;i<=n;i++)
scanf("%d",&a[i]);
for(i=1;i<=n-1;i++)
{
min=a[i];
k=i;
for(j=i+1;j<=n;j++)
{
if(a[j]<min)
{
min=a[j];
k=j;
}
a[k]=a[i];
}
a[i]=min;
}
printf("The sorted elements are:");
for(i=1;i<=n;i++)
printf("\n%d",a[i]);
getch();
}
```

OUTPUT:

```
Enter the number of elements : 4
Enter the elements :
23
45
56
32
The sorted elements are :
23
32
45
56
```

PROGRAM

```
#include<stdio.h>
#include<conio.h>
int n;
void main()
{
int i,a[10];
void bubble(int a[10]);
clrscr();
printf("Enter the number of elements:");
scanf("%d",&n);
printf("Enter the elements:");
for(i=0;i<n;i++)
scanf("%d",&a[i]);
bubble(a);
getch();
}
void bubble(int a[10])
{
int i,j,temp;
for(i=0;i<=n-2;i++)
{
for(j=0;j<=n-2-i;j++)
{
if(a[j]>a[j+1])
{
temp=a[j];
a[j]=a[j+1];
a[j+1]=temp;
}
}
}
printf("The sorted array:");
for(i=0;i<n;i++)
printf("\n%d",a[i]);
}
```

OUTPUT:

```
Enter the number of elements : 4
Enter the elements :
23
45
56
32
The sorted elements are :
23
32
45
56
```

PROGRAM

```
#include<stdio.h>
#include<conio.h>
int n;
void shell(int a[10],int n);
void main()
{
int i,a[10];
clrscr();
printf("Enter the number of elements:");
scanf("%d",&n);
printf("Enter the elements:");
for(i=0;i<n;i++)
{
scanf("%d",&a[i]);
}
shell(a,n);
printf("\n The sorted elements are:");
for(i=0;i<n;i++)
printf("\n%d",a[i]);
getch();
}
void shell(int a[10],int n)
{
int k,i,j,temp;
for(k=n/2;k>0;k=k/2)
for(i=k;i<n;i++)
{
temp=a[i];
for(j=i;(j>=k)&&(a[j-k]>temp);j=j-k)
{
a[j]=a[j-k];
}
a[j]=temp;
}
}
```

OUTPUT

```
Enter the number of elements : 4
Enter the elements :
23
45
56
32
The sorted elements are :
23
32
45
56
```

PROGRAM

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
int n;
void quick(int a[10], int low,int high);
int partition(int a[10],int low, int high);
void swap(int a[10],int *low, int *high);
void main()
{
int i,a[10];
clrscr();
printf("Enter the number of elements:");
scanf("%d",&n);
printf("Enter the elements:");
for(i=0;i<n;i++)
{
scanf("%d",&a[i]);
}
quick(a,0,n-1);
printf("\n The sorted elements are:");
for(i=0;i<n;i++)
printf("\n%d",a[i]);
getch();

}
void quick(int a[10],int low,int high)
{
int m,i;
if(low<high)
{
m=partition(a,low,high);
quick(a,low,m-1);
quick(a,m+1,high);
}
}
int partition(int a[10],int low,int high)
{
int pivot=a[low],i=low,j=high;
while(i<=j)
```

```
{
while(a[i]<=pivot)
i++;
while(a[j]>pivot)
{
j--;
}
if(i<=j)
swap(a,&i,&j);
}
swap(a,&low,&j);
return j;
}
void swap(int a[10],int*i,int*j)
{
int temp;
temp=a[*i];
a[*i]=a[*j];
a[*j]=temp;
}
```

OUTPUT

```
Enter the number of elements : 4
Enter the elements :
23
45
56
32
The sorted elements are :
23
32
45
56
```

PROGRAM

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
int n;
void merge(int a[30],int low,int high);
int combine(int a[30],int low,int mid,int high);
void main()
{
int i,a[30];
clrscr();
printf("Enter the number of elements:");
scanf("%d",&n);
printf("Enter the elements:");
for(i=0;i<n;i++)
{
scanf("%d",&a[i]);
}
merge(a,0,n-1);
printf("The sorted elements are:");
for(i=0;i<n;i++)
printf("\n%d",a[i]);
getch();
}
void merge(int a[30],int low,int high)
{
int mid;
if(low<high)
{
mid=(low+high)/2;
merge(a,low,mid);
merge(a,mid+1,high);
combine(a,low,mid,high);
}
}
int combine(int a[30],int low,int mid,int high)
{
int k=low,i=low,j=mid+1;
int temp[30];
while((i<=mid) && (j<=high))
{
if(a[i]<=a[j])
{
```

```
temp[k]=a[i];
i++;
k++;
}
else
 {
temp[k]=a[j];
j++;
k++;
}
}
while(i<=mid)
{
temp[k]=a[i];
i++;
k++;
}
while(j<=high)
{
temp[k]=a[j];
j++;
k++;
}
for(i=low;i<=high;i++)
a[i]=temp[i];
}
```

OUTPUT

```
Enter the number of elements : 4
Enter the elements :
23
45
56
32
The sorted elements are :
23
32
45
56
```

PROGRAM

```
#include<stdio.h>
#include<conio.h>
int min = 0, count = 0, array[100] = {0}, array1[100] = {0};
 void main()
{
   int k, i, j, temp, t, n;
   printf("Enter the number of elements :");
   scanf("%d", &count);
   printf("Enter the elements :");
   for (i = 0; i < count; i++)
   {
      scanf("%d", &array[i]);
      array1[i] = array[i];
   }
   for (k = 0; k < 3; k++)
   {
      for (i = 0; i < count; i++)
      {
         min = array[i] % 10;       /* To find minimum lsd */
         t = i;
         for (j = i + 1; j < count; j++)
         {
            if (min > (array[j] % 10))
            {
               min = array[j] % 10;
               t = j;
            }
     }
         temp = array1[t];
         array1[t] = array1[i];
         array1[i] = temp;
         temp = array[t];
         array[t] = array[i];
         array[i] = temp;

      }
      for (j = 0; j < count; j++)       /*to find MSB */
         array[j] = array[j] / 10;
   }
```

```
   printf("The sorted elements are: ");
   for (i = 0; i < count; i++)
      printf("%d ", array1[i]);
getch();
}
```

OUTPUT

```
Enter the number of elements : 4
Enter the elements :
23
45
56
32
The sorted elements are :
23
32
45
56
```

PROGRAM

```
#include<stdio.h>
#include<stdlib.h>
#include<conio.h>
void create_list(int);
void addatbeg(int);
void addafter(int,int);
void del(int);
void display();
void rev();
void search(int);
void count();
struct node
{
        int info;
        struct node *link;
}*start;

void main()
{
        int choice=0,n,m,position,i;
        start=NULL;
        clrscr();
        while(1)
        {
                printf("\n1.INSERTION");
                printf("\n2.ADD AT BEGINNING");
                printf("\n3.ADD AFTER");
                printf("\n4.DELETION");
                printf("\n5.DISPLAY");
                printf("\n6.COUNT");
                printf("\n7.REVERE");
                printf("\n8.SEARCH");
                printf("\n9.QUIT");
                printf("\nENTER THE CHOICE...");
                scanf("%d",&choice);
                switch(choice)
                {
```

```
case 1:
        printf("\nNODES YOU NEED...");
        scanf("%d",&n);
        for(i=0;i<n;i++)
        {
                printf("\nENTER THE ELEMENT...");
                scanf("%d",&m);
                create_list(m);
        }
        break;
case 2:

        printf("\nENTER THE ELEMENT...");
        scanf("%d",&m);
        addatbeg(m);
        break;
case 3:
        printf("\nENTER THE ELEMENT...");
        scanf("%d",&m);
        printf("\nENTER POSTION TO INSERT...");
        scanf("%d",&position);
        addafter(m,position);
        break;
case 4:
        if(start==NULL)
        {
                printf("\nLIST IS EMPTY");
                continue;
        }
        printf("\nENTER THE ELEMENT FOR DELETION...");
        scanf("%d",&m);
        del(m);
        break;
case 5:
        display();
        break;
case 6:
        count();
        break;
case 7:
        rev();
        break;
case 8:
        printf("\nENTER THE ELEMENT TO BE SEARCHED...");
        scanf("%d",&m);
        search(m);
```

```
                        break;
                case 9:
                        exit(0);
                default:
                        printf("\nTHE INVALID CHOICE");
                }
        }
}
void create_list(int data)
{
        struct node *q,*temp;
        temp=malloc(sizeof(struct node));
        temp->info=data;
        temp->link=NULL;
        if(start==NULL)
        {
                start=temp;
        }
        else
        {
                q=start;
                while(q->link!=NULL)
                {
                        q=q->link;
                }
                q->link=temp;
        }
}
void addatbeg(int data)
{
        struct node *q,*temp;
        temp=malloc(sizeof(struct node));
        temp->info=data;
        temp->link=start;
        start=temp;
}
void addafter(int data,int pos)
{
        struct node *q,*temp;
        int i;
        q=start;
        for(i=0;i<(pos-1);i++)
        {
                q=q->link;
                if(q==NULL)
```

```
                {
                        printf("\nTHERE ARE LESS THAN %d ELEMENT.",pos);
                        return;
                }
        }
        temp=malloc(sizeof(struct node));
        temp->link=q->link;
        temp->info=data;
        q->link=temp;
}
void del(int data)
{
        struct node *temp,*q;
        if(start->info==data)
        {
                temp=start;
                start=start->link;
                free(temp);
                return;
        }
        q=start;
        while(q->link->link!='\0')
        {
                if(q->link->info==data)
                {
                        temp=q->link;
                        q->link=temp->link;
                        free(temp);
                        return;
                }
                q=q->link;
        }
        if(q->link->info==data)
        {
                temp=q->link;
                free(temp);
                q->link='\0';
                return;
        }
        printf("\nELEMENT %d NOT FOUND.",data);
}
void display()
{
        struct node *q;
        if(start=='\0')
```

```
	{
		printf("\nTHE LIST IS EMPTY");
		return;
	}
	q=start;
	while(q!='\0')
	{
		printf("\nTHE ELEMENT IS...%d",q->info);
		q=q->link;
	}
	printf("\n");
}
void count()
{
	struct node *q=start;
	int cnt=0;
	while(q!='\0')
	{
		q=q->link;
		cnt++;
	}
	printf("\nNUMBER OF ELEMENT ARE %d",cnt);
}
void rev()
{
	struct node *p1,*p2,*p3;
	if(start->link=='\0')
	{
		return;
	}
	p1=start;
	p2=p1->link;
	p3=p2->link;
	p1->link='\0';
	p2->link=p1;
	while(p3!='\0')
	{
		p1=p2;
		p2=p3;
		p3=p3->link;
		p2->link=p1;
	}
	start=p2;
	printf("\nTO CHECK REVERSED LIST PLEASE CHOOSE OPTION 5");
}
```

```
void search(int data)
{
	struct node *ptr=start;
	int pos=1;
	while(ptr!='\0')
	{
		if(ptr->info==data)
		{
			printf("\nITEM %d FOUND AT POSITON...%d",data,pos);
			return;
		}
		ptr=ptr->link;
		pos++;
	}
	if(ptr=='\0')
	{
		printf("\nITEM %d NOT FOUND IN LIST",data);
	}
}
```

OUTPUT:

1. INSERTION
2. ADD AT BEGINNING
3. ADD AFTER
4. DELETION
5. DISPLAY
6. COUNT
7. REVERSE
8. SEARCH
9. QUIT

ENTER THE CHOICE….1
NODES YOU NEED…3
ENTER THE ELEMET…32
ENTER THE ELEMET…5
ENTER THE ELEMET…7

1. INSERTION
2. ADD AT BEGINNING
3. ADD AFTER
4. DELETION
5. DISPLAY
6. COUNT
7. REVERSE
8. SEARCH
9. QUIT

ENTER THE CHOICE….2
ENTER THE ELEMET…3

1. INSERTION
2. ADD AT BEGINNING
3. ADD AFTER
4. DELETION
5. DISPLAY
6. COUNT
7. REVERSE
8. SEARCH

```
9.QUIT
ENTER THE CHOICE….3
ENTER THE ELEMET…78
ENTER POSITION TO INSERT…2

1.INSERTION
2.ADD AT BEGINNING
3.ADD AFTER
4.DELETION
5.DISPLAY
6.COUNT
7.REVERSE
8.SEARCH
9.QUIT
ENTER THE CHOICE….5
ELEMENT IS…3
ELEMENT IS…32
ELEMENT IS…78
ELEMENT IS…5
ELEMENT IS…7

1.INSERTION
2.ADD AT BEGINNING
3.ADD AFTER
4.DELETION
5.DISPLAY
6.COUNT
7.REVERSE
8.SEARCH
9.QUIT
ENTER THE CHOICE….4
ENTER THE ELEMENT FOR DELETION…32

1.INSERTION
2.ADD AT BEGINNING
3.ADD AFTER
4.DELETION
5.DISPLAY
6.COUNT
7.REVERSE
8.SEARCH
9.QUIT
ENTER THE CHOICE….6
NUMBER OF ELEMENTS ARE…4
```

1. INSERTION
2. ADD AT BEGINNING
3. ADD AFTER
4. DELETION
5. DISPLAY
6. COUNT
7. REVERSE
8. SEARCH
9. QUIT

ENTER THE CHOICE….7
ELEMENT IS…7
ELEMENT IS…5
ELEMENT IS…78
ELEMENT IS…3

1. INSERTION
2. ADD AT BEGINNING
3. ADD AFTER
4. DELETION
5. DISPLAY
6. COUNT
7. REVERSE
8. SEARCH
9. QUIT

ENTER THE CHOICE….8
ENTER ELEMENT TO BE SEARCHED…78
ITEM 78 FOUND AT POSITION 3

PROGRAM

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
void create_list(int);
void addatbeg(int);
void addafter(int,int);
void del(int);
void display();
void rev();
void count();
struct node
{
        struct node *prev;
        int info;
        struct node *next;
}*start;

void main()
{
        int choice,n,m,po,i;
        clrscr();
        start=NULL;
        while(1)
        {
                printf("\n 1.CREATE A LIST");
                printf("\n 2.ADD AT BEGINNING");
                printf("\n 3.ADD AFTER");
                printf("\n 4.DELETE");
                printf("\n 5.DISPLAY");
                printf("\n 6.COUNT");
                printf("\n 7.REVERSE");
                printf("\n 8.QUIT");
                printf("\nENTER YOUR CHOICE...");
                scanf("%d",&choice);
                switch(choice)
                {
                        case 1:
                                printf("\nENTER NODES YOU NEED...");
                                scanf("%d",&n);
                                for(i=0;i<n;i++)
                                {
```

```
                printf("\nENTER THE ELEMENT...");
                scanf("%d",&m);
                create_list(m);
        }
break;
case 2:

        printf("\nENTER THE ELEMENT...");
        scanf("%d",&m);
        addatbeg(m);
 break;
case 3:


        printf("\nENTER THE ELEMENT...");
        scanf("%d",&m);
        printf(" ENTER THE POSTION...");
        scanf("%d",&po);
        addafter(m,po);
 break;
case 4:
        if(start==NULL)
        {
                printf("\nTHE LIST IS EMPTY");
                continue;
        }
        printf("\nENTER THE DELETION ELEMENT...");
        scanf("%d",&m);
        del(m);
 break;
case 5:
        if(start==NULL)
        {
                printf("\n List is empty!");
                continue;
        }
        display();
 break;
case 6:
        count();
 break;
case 7:
        if(start==NULL)
        {

                printf("\nLIST IS EMPTY!");
                continue;
```

}

```
                        rev();
                    break;
                    case 8:
                        exit(0);
                    default:
                        printf("\nWRONG CHOICE");
                }
                getch();
        }
}
void create_list(int num)
{
        struct node *q,*tmp;
        tmp=malloc(sizeof(struct node));
        tmp->info=num;
        tmp->next=NULL;
        if(start==NULL)
        {
                tmp->prev=NULL;
                start->prev=tmp;
                start=tmp;
        }
        else
        {
                q=start;
                while(q->next!=NULL)
                {
                        q=q->next;
                }
                q->next=tmp;
                tmp->prev=q;
        }
}
void addatbeg(int num)
{
        struct node *tmp;
        tmp=malloc(sizeof(struct node));
        tmp->prev=NULL;
        tmp->info=num;
        tmp->next=start;
        start->prev=tmp;
        start=tmp;
}
void addafter(int num,int c)
{
```

```
        struct node *tmp,*q;
        int i;
        q=start;
        for(i=0;i<(c-1);i++)
        {
                q=q->next;
                if(q==NULL)
                {
                        printf("\nTHERE IS LESS THAN %d ELEMENT",c);
                        return;
                }
        }
        tmp=malloc(sizeof(struct node));
        tmp->info=num;
        q->next->prev=tmp;
        tmp->next=q->next;
        tmp->prev=q;
        q->next=tmp;
}
void del(int num)
{
        struct node *tmp,*q;
        if(start->info==num)
        {
                tmp=start;
                start=start->next;
                start->prev=NULL;
                free(tmp);
                return;
        }
        q=start;
        while(q->next->info!=num)
        {
                q=q->next;
        }
        if((q->next->info==num)&&(q->next->next==NULL))
        {
                tmp=q->next;
                free(tmp);
                q->next=NULL;
                return;
        }
        if(q->next->info==num)
        {
                tmp=q->next;
```

```
                q->next=tmp->next;
                tmp->next->prev=q;
                free(tmp);
                return;
        }
        printf("\nTHE ELEMENT %d NOT FOUND",num);
}
void display()
{
        struct node *q;
        if(start==NULL)
        {
                printf("\nLIST IS EMPTY");
                return;
        }
        q=start;
        printf("\nLIST IS");
        while(q!=NULL)
        {
                printf(" %d ",q->info);
                q=q->next;
        }
        printf("\n");
}
void count()
{
        struct node *q=start;
        int cnt=0;
        while(q!=NULL)
        {
                q=q->next;
                cnt++;
        }
        printf("\nNUMBER OF ELEMENT IS %d",cnt);
}
void rev()
{
        struct node *p1,*p2;
        p1=start;
        p2=p1->next;
        p1->next=NULL;
        p1->prev=p2;
        while(p2!=NULL)
        {
                p2->prev=p2->next;
```

```
            p2->next=p1;
            p1=p2;
            p2=p2->prev;
        }
        start=p1;
}
```

OUTPUT:

```
1.CREATE  A LIST
2.ADD AT BEGINNING
3.ADD AFTER
4.DELETION
5.DISPLAY
6.COUNT
7.REVERSE
8.SEARCH
9.QUIT
ENTER THE CHOICE….1
ENTER NODES YOU NEED… 2
ENTER THE ELEMENT…12
ENTER THE ELEMENT 43

1.CREATE  A LIST
2.ADD AT BEGINNING
3.ADD AFTER
4.DELETION
5.DISPLAY
6.COUNT
7.REVERSE
8.SEARCH
9.QUIT
ENTER THE CHOICE….2
ENTER THE ELEMENT…4

1.CREATE  A LIST
2.ADD AT BEGINNING
3.ADD AFTER
4.DELETION
5.DISPLAY
6.COUNT
7.REVERSE
8.SEARCH
9.QUIT
ENTER THE CHOICE….3
ENTER THE ELEMENT…23
ENTER THE POSITION…2

1.CREATE  A LIST
2.ADD AT BEGINNING
3.ADD AFTER
4.DELETION
5.DISPLAY
```

```
6.COUNT
7.REVERSE
8.SEARCH
9.QUIT
ENTER THE CHOICE….5
LIST IS….43  23  12  4

1.CREATE  A LIST
2.ADD AT BEGINNING
3.ADD AFTER
4.DELETION
5.DISPLAY
6.COUNT
7.REVERSE
8.SEARCH
9.QUIT
ENTER THE CHOICE….
LIST IS….4  12  23  43
```

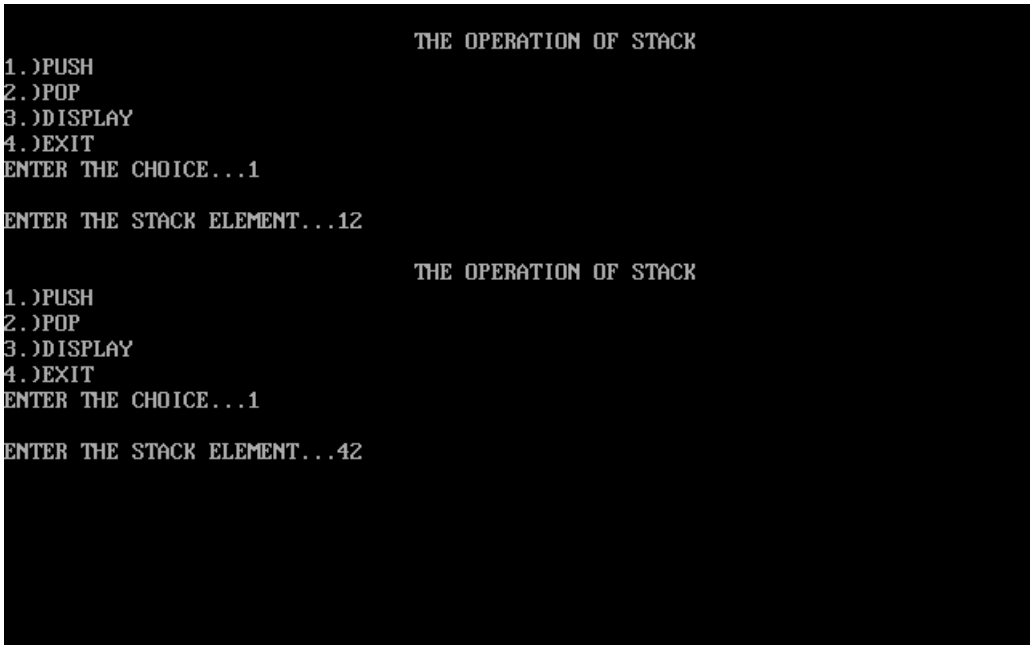
PROGRAM

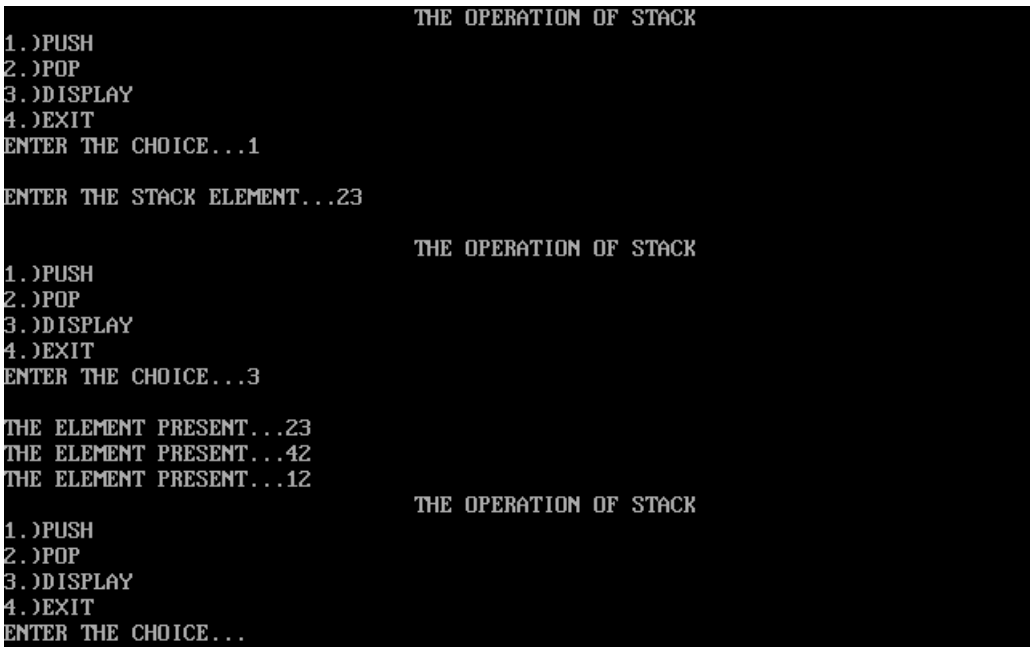
```
#include<stdio.h>
#include<conio.h>
#define SIZE 5
int front=-1,ch,stack[SIZE];
int push();
int  pop();
int display();
void main()
{
        clrscr();
        while(1)
        {
                printf("\n\t\t\t\tTHE OPERATION OF STACK");
                printf("\n1.)PUSH");
                printf("\n2.)POP");
                printf("\n3.)DISPLAY");
                printf("\n4.)EXIT");
                printf("\nENTER THE CHOICE...");
                scanf("%d",&ch);
                switch(ch)
                {
                        case 1:
```

```
                push();
            break;
            case 2:
                pop();
            break;
            case 3:
                display();
            break;
            case 4:
                exit(0);
            break;

            default:
                printf("\nTHE INVALID CHOICE");
        }
    }
    getch();
}
int push()
{
    int stack_item;
    if(front==(SIZE-1))
    {
        printf("\nTHE STACK IS OVERFLOW");
    }
    else
    {
        printf("\nENTER THE STACK ELEMENT...");
        scanf("%d",&stack_item);
        front++;
```

```
                stack[front]=stack_item;
        }
}
int pop()
{
        if(front==-1)
        {
                printf("\nTHE STACK IS UNDERFLOW");
        }
        else
        {
                printf("\nTHE STACK ELEMENT %d IS DELETED",stack[front]);
                front--;
        }
}

int display()
{
        int i;
        if(front==-1)
        {
                printf("\nTHE STACK IS UNDERFLOW");
        }
        else
        {
                for(i=front;i>=0;i--)
                {
                        printf("\nTHE ELEMENT PRESENT...%d",stack[front]);
                }
        } }
```

OUTPUT:


THE OPERATION OF STACK
1.)PUSH
2.)POP
3.)DISPLAY
4.)EXIT
ENTER THE CHOICE...1
ENTER THE STACK ELEMENT...12
THE OPERATION OF STACK
1.)PUSH
2.)POP
3.)DISPLAY
4.)EXIT
ENTER THE CHOICE...1
ENTER THE STACK ELEMENT...42


THE OPERATION OF STACK
1.)PUSH
2.)POP
3.)DISPLAY
4.)EXIT
ENTER THE CHOICE...1
ENTER THE STACK ELEMENT...23
THE OPERATION OF STACK
1.)PUSH
2.)POP
3.)DISPLAY
4.)EXIT
ENTER THE CHOICE...3
THE ELEMENT PRESENT...23
THE ELEMENT PRESENT...42
THE ELEMENT PRESENT...12
THE OPERATION OF STACK
1.)PUSH
2.)POP
3.)DISPLAY
4.)EXIT
ENTER THE CHOICE...

}

PROGRAM

```
#include<stdio.h>
#include<ctype.h>
#include<string.h>
static char str[20];
int top=-1;
main()
{
char in[20],post[20],ch;
int i,j,l;
clrscr();
printf("enter the string");
gets(in);
l=strlen(in);
for(i=0,j=0;i<l;i++)
if(isalpha(in[i]))
post[j++]=in[i];
else
{
if(in[i]=='(')
push(in[i]);
else if(in[i]==')')
while((ch=pop())!='(')
post[j++]=ch;
else
{
while(priority(in[i])<=priority(str[top]))
post[j++]=pop();
push(in[i]);
}
}
while(top!=-1)
post[j++]=pop();
post[j]='\0';
printf("\n equivalent infix to postfix is:%s",post);
getch();
}
priority (char c)
{
switch(c)
{
case'+':
case'-': return 1;
```

```
case'*':
case'/':
return 2;
case'$':            return 3;
}
return 0;
}
push(char c)
{
str[++top]=c;
}
pop()
{
return(str[top--]);
```

OUTPUT

enter the string(a+b)-(c-d)*e/f

equivalent infix to postfix is:ab+cd-e*f/-

enter the stringa+b/c*d

equivalent infix to postfix is:abc/d*+

PROGRAM

```
#include<stdio.h>
#include<conio.h>
struct node
{
        int data;
        struct node *right,*left;
}*root,*p,*q;
struct node *make(int y)
{
        struct node *newnode;
        newnode=(struct node *)malloc(sizeof(struct node));
        newnode->data=y;
        newnode->right=newnode->left=NULL;
        return(newnode);
}
void left(struct node *r,int x)
{
        if(r->left!=NULL)
        {
                printf("\nINVALID");
        }
        else
        {
                r->left=make(x);
        }
}

void right(struct node *r,int x)
{
        if(r->right!=NULL)
        {
                printf("\nINVALID");
        }
        else

                r->right=make(x);
}

void inorder(struct node *r)
{
        if(r!=NULL)
        {
                inorder(r->left);
                printf("\t %d",r->data);
```

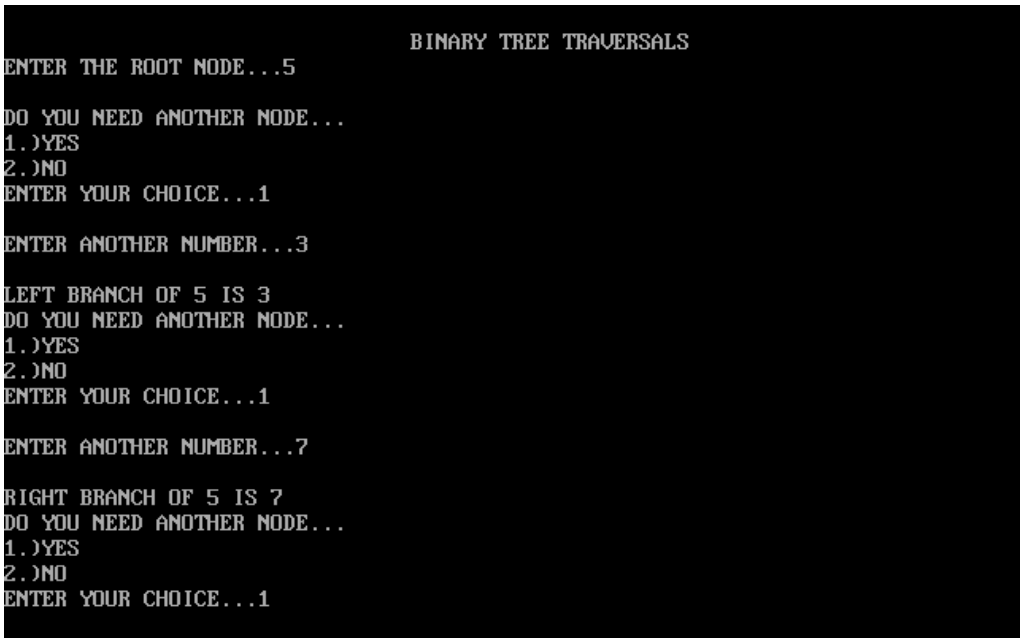
```
                inorder(r->right);
        }
}

void preorder(struct node *r)
{
        if(r!=NULL)
        {
                printf("\t %d",r->data);
                preorder(r->left);
                preorder(r->right);
        }
}
void postorder(struct node *r)
{
        if(r!=NULL)
        {
                postorder(r->left);
                postorder(r->right);
                printf("\t %d",r->data);
        }
}
void main()
{
        int no;
        int choice;
        clrscr();
        printf("\n\t\t\t\tBINARY TREE TRAVERSALS");
        printf("\nENTER THE ROOT NODE...");
        scanf("%d",&no);
        root=make(no);
        p=root;
        while(1)
        {
                printf("\nDO YOU NEED ANOTHER NODE...");
                printf("\n1.)YES");
                printf("\n2.)NO");
                printf("\nENTER YOUR CHOICE...");
                scanf("%d",&choice);
                if(choice==1)
                {
                        printf("\nENTER ANOTHER NUMBER...");
                        scanf("%d",&no);
                        while(1)
                        {
```

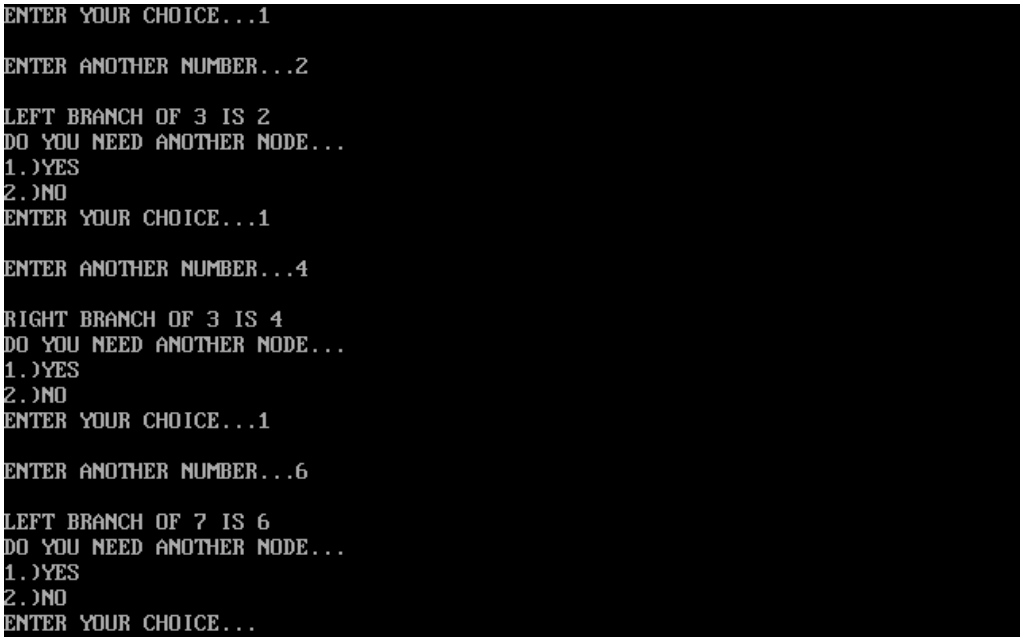
```
                        p=root;
                        q=root;
                        while((no!=p->data)&&(q!=NULL))
                        {
                                p=q;
                                if(no<p->data)
                                {
                                        q=p->left;
                                }
                                else
                                {
                                        q=p->right;
                                }
                        }

                        if(no<p->data)
                        {
                                printf("\nLEFT BRANCH OF %d IS %d",p->data,no);
                                left(p,no);
                                break;
                        }
                        else
                        {
                                right(p,no);
                                printf("\nRIGHT BRANCH OF %d IS %d",p->data,no);
                                break;
                        }
                }
        }

        else if(choice==2)
        {
                break;
        }
        else
        {
                printf("\nINVALID OPTION");
                continue;
        }
}

while(1)
{
        printf("\n1.)INORDER TRAVERSALS");
        printf("\n2.)PREORDER TRAVERSALS");
        printf("\n3.)POST TRAVERSALS");
        printf("\n4.)EXIT");
```

```
printf("\nENTER YOUR CHOICE...");
```

```
        scanf("\%d",&choice);
        switch(choice)
        {
                case 1:
                        inorder(root);
                 break;
                 case 2:
                        preorder(root);
                 break;
                 case 3:
                        postorder(root);
                 break;
                 case 4:
                        exit(0);
                default:
                        printf("\nINVALID OPTION");
                break;
        }
        getch();
}
```

OUTPUT:


BINARY TREE TRAVERSALS
ENTER THE ROOT NODE...5
DO YOU NEED ANOTHER NODE...
1.)YES
2.)NO
ENTER YOUR CHOICE...1
ENTER ANOTHER NUMBER...3
LEFT BRANCH OF 5 IS 3
DO YOU NEED ANOTHER NODE...
1.)YES
2.)NO
ENTER YOUR CHOICE...1
ENTER ANOTHER NUMBER...7
RIGHT BRANCH OF 5 IS 7
DO YOU NEED ANOTHER NODE...
1.)YES
2.)NO
ENTER YOUR CHOICE...1


ENTER YOUR CHOICE...1
ENTER ANOTHER NUMBER...2
LEFT BRANCH OF 3 IS 2
DO YOU NEED ANOTHER NODE...
1.)YES
2.)NO
ENTER YOUR CHOICE...1
ENTER ANOTHER NUMBER...4
RIGHT BRANCH OF 3 IS 4
DO YOU NEED ANOTHER NODE...
1.)YES
2.)NO
ENTER YOUR CHOICE...1
ENTER ANOTHER NUMBER...6
LEFT BRANCH OF 7 IS 6
DO YOU NEED ANOTHER NODE...
1.)YES
2.)NO
ENTER YOUR CHOICE...

```
LEFT BRANCH OF 7 IS 6
DO YOU NEED ANOTHER NODE...
1.)YES
2.)NO
ENTER YOUR CHOICE...1

ENTER ANOTHER NUMBER...9

RIGHT BRANCH OF 7 IS 9
DO YOU NEED ANOTHER NODE...
1.)YES
2.)NO
ENTER YOUR CHOICE...2

1.)INORDER TRAVERSALS
2.)PREORDER TRAVERSALS
3.)POST TRAVERSALS
4.)EXIT
ENTER YOUR CHOICE...1
        2       3       4       5       6       7       9
1.)INORDER TRAVERSALS
2.)PREORDER TRAVERSALS
3.)POST TRAVERSALS
4.)EXIT
ENTER YOUR CHOICE...2
```

```
RIGHT BRANCH OF 7 IS 9
DO YOU NEED ANOTHER NODE...
1.)YES
2.)NO
ENTER YOUR CHOICE...2

1.)INORDER TRAVERSALS
2.)PREORDER TRAVERSALS
3.)POST TRAVERSALS
4.)EXIT
ENTER YOUR CHOICE...1
        2       3       4       5       6       7       9
1.)INORDER TRAVERSALS
2.)PREORDER TRAVERSALS
3.)POST TRAVERSALS
4.)EXIT
ENTER YOUR CHOICE...2
        5       3       2       4       7       6       9
1.)INORDER TRAVERSALS
2.)PREORDER TRAVERSALS
3.)POST TRAVERSALS
4.)EXIT
ENTER YOUR CHOICE...3
        2       4       3       6       9       7       5
```

PROGRAM

```
#include<stdio.h>
#include<conio.h>
#include<process.h>
#include<stdlib.h>
# define MAX 10
# define TRUE 1
# define FALSE 0
typedef struct node
{
int vertex;
struct node *next;
}node;
node  *head[10];
int visited[MAX];
int Queue[MAX];
int front,rear;
int n;
void add(int vertex1, int vertex2)
{
node *New, *first;
New=(node *)malloc(sizeof(node));
if(New==NULL)
printf("\n Insuffucient memory");
New->vertex=vertex2;
New->next=NULL;
first=head[vertex1];
if(first==NULL)
head[vertex1]=New;
else
{
while(first->next!=NULL)
first=first->next;
first->next=New;
}
New=(node *)malloc(sizeof(node));
if(New==NULL)
printf("\n Insufficient memory");
New->vertex=vertex1;
New->next=NULL;
first=head[vertex2];
if(first==NULL)
head[vertex2]=New;
else
```

```
{
while(first->next!=NULL)
first=first->next;
first->next=New;
}
}
void create()
{
int V1,V2;
char ans='y';
for(V1=0;V1<MAX;V1++)
head[V1]=NULL;
printf("\n enter the vertices no. beginning witn 0");
do
{
printf("\n Enter the edge of the graph");
scanf("%d %d",&V1,&V2);
if(V1>=MAX || V2>=MAX)
printf("\n Invalid Vertex value");
else
add(V1,V2);
printf("\n Want to add more edges(y/n)");
ans=getche();
}while(ans=='y');
}
void bfs(int V1)
{
int i;
node *first;
front=-1;
rear=-1;
Queue[++rear]=V1;
while(front!=rear)
{
i=Queue[++front];
if(visited[i]==FALSE)
{
printf("\n %d",i);
visited[i]=TRUE;
}
first=head[i];
while(first!=NULL)
{
if(visited[first->vertex]==FALSE)
Queue[++rear]=first->vertex;
```

```
first=first->next;
}
}
}
void main()
{
int V1,V2;
char  ans;
clrscr();
create();
do
{
for(V1=0;V1<n;V1++)
visited[V1]=FALSE;
clrscr();
printf("\n Enter the vertex from which u want to traverse");
scanf("%d",&V1);
if(V1>=MAX)
printf("\n Invalid vertex");
else
{
printf("\n The breadth first search of the Graph is\n");
front=rear=-1;
bfs(V1);
getch();
}
printf("\n Do u want to continue");
ans=getche();
}
while(ans=='y');
exit(0);
}
```

OUTPUT

```
enter the vertices no. beginning with 0
Enter the edge of the graph 0 1

Want to add more edges(y/n)y
Enter the edge of the graph
0 2

Want to add more edges(y/n)y
Enter the edge of the graph
0 3

Want to add more edges(y/n)y
Enter the edge of the graph 0 4

Want to add more edges(y/n)n
Enter the vertex from which u want to traverse0

The breadth first search of the Graph is
0
1
2
3
4
Do u want to continue n
```

PROGRAM

```
#include<stdio.h>
#include<conio.h>
#include<process.h>
#include<stdlib.h>
# define MAX 10
# define TRUE 1
# define FALSE 0
typedef struct node
{
int vertex;
struct node *next;
}node;
node  *head[10];
int  visited[MAX];
int Queue[MAX];
int front,rear;
int n;
void add(int vertex1, int vertex2)
{
node *New, *first;
New=(node *)malloc(sizeof(node));
if(New==NULL)
printf("\n Insuffucient memory");
New->vertex=vertex2;
New->next=NULL;
first=head[vertex1];
if(first==NULL)
head[vertex1]=New;
else
{
while(first->next!=NULL)
first=first->next;
first->next=New;
}
New=(node *)malloc(sizeof(node));
if(New==NULL)
printf("\n Insufficient memory");
New->vertex=vertex1;
New->next=NULL;
first=head[vertex2];
if(first==NULL)
```

```
head[vertex2]=New;
else
{
while(first->next!=NULL)
first=first->next;
first->next=New;
}
}
void create()
{
int V1,V2;
char ans='y';
for(V1=0;V1<MAX;V1++)
head[V1]=NULL;
printf("\n enter the vertices no. beginning witn 0");
do
{
printf("\n Enter the edge of the graph");
scanf("%d %d",&V1,&V2);
if(V1>=MAX || V2>=MAX)
printf("\n Invalid Vertex value");
else
add(V1,V2);
printf("\n Want to add more edges(y/n)");
ans=getche();
}while(ans=='y');
}
void dfs(int V1)
{
int V2;
node *first;

printf("\n %d",V1);
visited[V1]=TRUE;

first=head[V1];
while(first!=NULL)

if(visited[first->vertex]==FALSE)
dfs(first->vertex);
else
first=first->next;
}
void main()
{
```

```
int V1,V2;
char ans='y';
clrscr();
create();
do
{
for(V1=0;V1<n;V1++)
visited[V1]=FALSE;
clrscr();
printf("\n Enter the vertex from which u want to traverse");
scanf("%d",&V1);
if(V1>=MAX)
printf("\n Invalid vertex");
else
{
printf("\n The depth first search of the Graph is\n");
dfs(V1);
getch();
}
printf("\n Do u want to continue");
ans=getche();
}
while(ans=='y');
exit(0);
}
```

OUTPUT

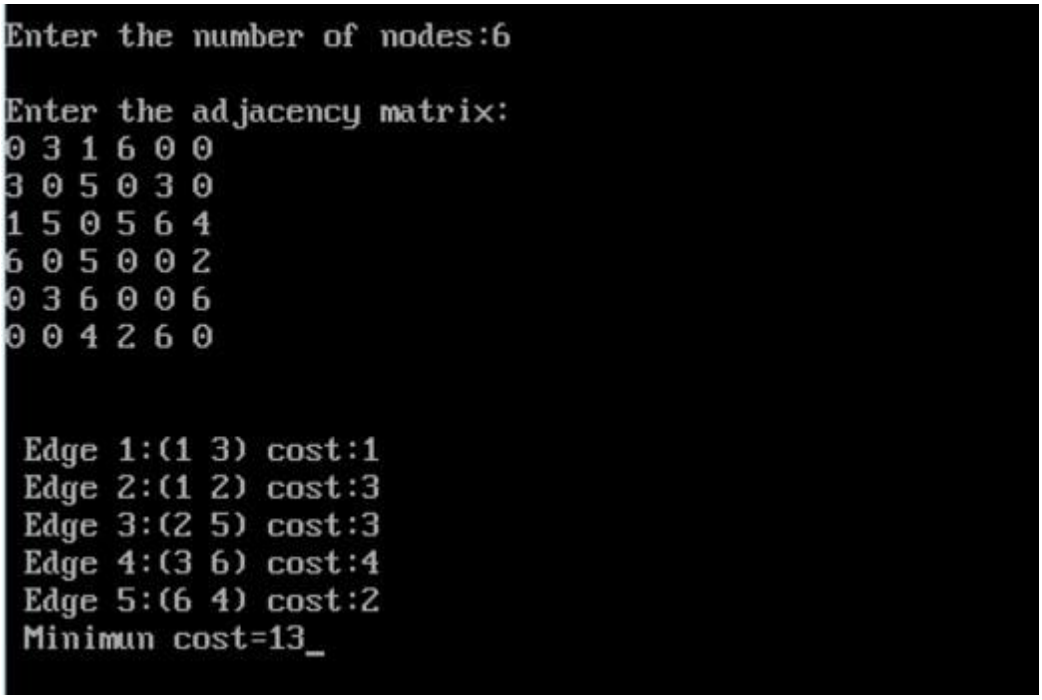
```
enter the vertices no. beginning witn 0
Enter the edge of the graph0 1

Want to add more edges(y/n)y
Enter the edge of the graph0 2

Want to add more edges(y/n)y
Enter the edge of the graph
1 3

Want to add more edges(y/n)y
Enter the edge of the graph
2 3

Want to add more edges(y/n)n
Enter the vertex from which u want to traverse0

The depth first search of the Graph is

0
1
3
2
Do u want to continue
```

PROGRAM

```
#include<stdio.h>
 #include<conio.h>
 int a,b,u,v,n,i,j,ne=1;
 int visited[10]={0},min,mincost=0,cost[10][10];
 void main()
 {
 clrscr();
 printf("\nEnter the number of nodes:");
 scanf("%d",&n);
 printf("\nEnter the adjacency matrix:\n");
 for(i=1;i<=n;i++)
for(j=1;j<=n;j++)
{
scanf("%d",&cost[i][j]);
if(cost[i][j]==0)
cost[i][j]=999;
}
 visited[1]=1;
 printf("\n");
 while(ne < n)
 {
 for(i=1,min=999;i<=n;i++)
 for(j=1;j<=n;j++)
 if(cost[i][j]< min)
 if(visited[i]!=0)
 {
```

```
min=cost[i][j];
a=u=i;
b=v=j;
}
if(visited[u]==0 || visited[v]==0)
{
printf("\n Edge %d:(%d %d) cost:%d",ne++,a,b,min);
mincost+=min;
visited[b]=1;
}
cost[a][b]=cost[b][a]=999;
}
printf("\n Minimun cost=%d",mincost);
getch();
}
```

OUTPUT


Enter the number of nodes:6
Enter the adjacency matrix:
0 3 1 6 0 0
3 0 5 0 3 0
1 5 0 5 6 4
6 0 5 0 0 2
0 3 6 0 0 6
0 0 4 2 6 0
Edge 1:(1 3) cost:1
Edge 2:(1 2) cost:3
Edge 3:(2 5) cost:3
Edge 4:(3 6) cost:4
Edge 5:(6 4) cost:2
Minimun cost=13_

PROGRAM:

```
#include<stdio.h>
#include<conio.h>
#define INFINITY 1000
int n,cost[10][10],dist[10],visit[10],front,rear,v;
void Initial(int n)
{
	int i,j;
	for(i=1;i<=n;i++)
	{
		for(j=1;j<=n;j++)
		{
			cost[i][j] = INFINITY;
		}
	}
	for(i=1;i<=n;i++)
	{
		dist[i] = INFINITY;
		visit[i] = 0;
	}
}
int nearest()
{
	int i,node, weight;
	weight = INFINITY;
	for(i=1;i<=n;i++)
	{
		if(visit[i] == 0 &&dist[i] < weight)
		{
			weight = dist[i];
			node=i;
		}
	}
	return node;
}
void Dijkstra(int v)
{
	int i,u,q,alt;
	dist[v]=0;
	for(i=1;i<=n;i++)
	{
		u=nearest();
```

```
                for(v=1;v<=n;v++)
                {
                        if(cost[u][v] == INFINITY)
                        {
                                continue;
                        }
                        alt = dist[u] + cost[u][v];
                        if(alt <dist[v])
                        {
                                dist[v] = alt;
                        }
                }
                visit[u] = 1;
        }
 }
void display()
{
        int i,j;
        printf("\nCOST MATRIX\n");
        for(i=1;i<=n;i++)
        {
                for(j=1;j<=n;j++)
                {
                        printf("%d\t",cost[i][j]);
                }
                printf("\n");
        }
}
void main()
{
        int i,j,k;
        clrscr();
        printf("ENTER THE NUMBER OF VERTCIES...");
        scanf("%d",&n);
        Initial(n);
        for(i=1;i<=n;i++)
        {
                for(j=i+1;j<=n;j++)
                {
                        printf("ENTER THE SHORTEST PATH BETWEEN %d AND %d",i,j);
                        printf("\nIF EDGE NOT EXIST ENTER 1000...");
                        scanf("%d",&cost[i][j]);
                        cost[j][i] = cost[i][j];
                }
        }
```

```
        display();
        printf("\nENTER THE SOURCE VERTEX...");
        scanf("%d",&v);
        Dijkstra(v);
        printf("\nRESULT\n");
        for(i=1;i<=n;i++)
        {
                printf("\nDISTANCE OF %d IS...%d",i,dist[i]);
        }
        getch();
}
```

OUTPUT:

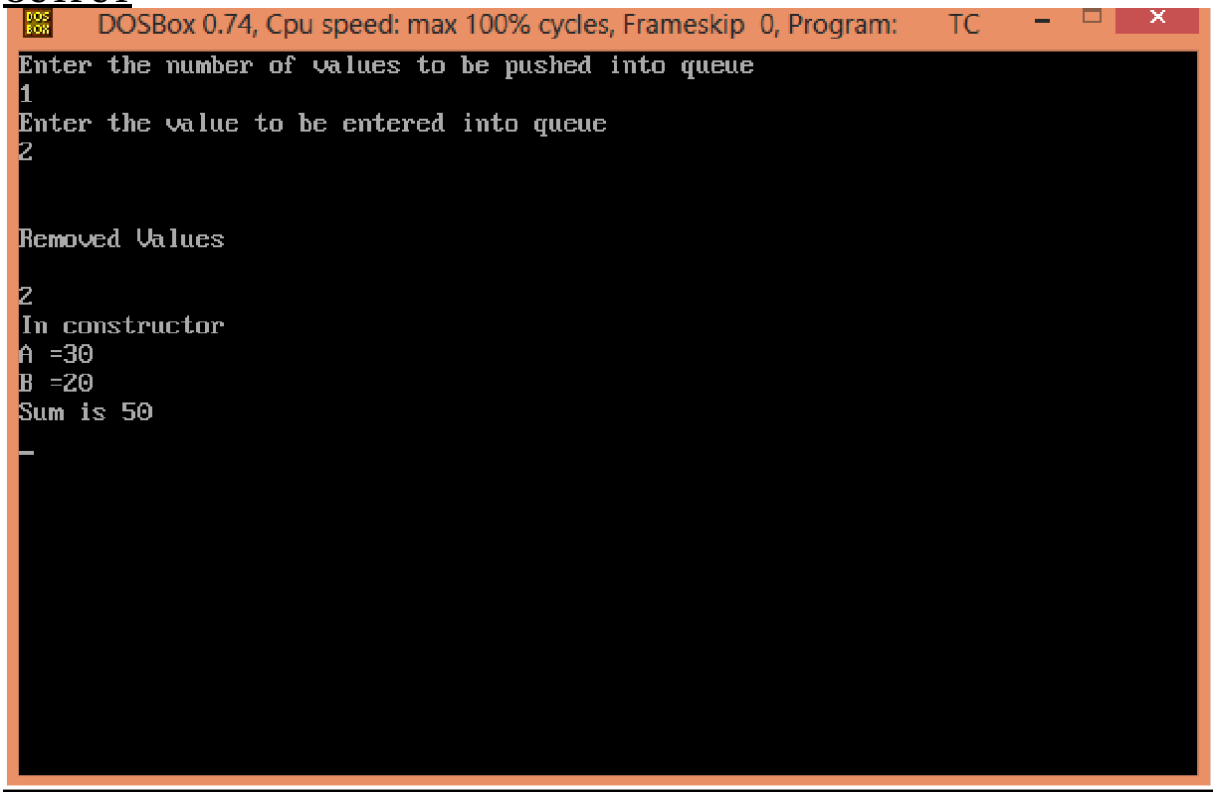
```
ENTER THE NUMBER OF VERTCIES...6
ENTER THE SHORTEST PATH BETWEEN 1 AND 2
IF EDGE NOT EXIST ENTER 1000...1
ENTER THE SHORTEST PATH BETWEEN 1 AND 3
IF EDGE NOT EXIST ENTER 1000...12
ENTER THE SHORTEST PATH BETWEEN 1 AND 4
IF EDGE NOT EXIST ENTER 1000...1000
ENTER THE SHORTEST PATH BETWEEN 1 AND 5
IF EDGE NOT EXIST ENTER 1000...1000
ENTER THE SHORTEST PATH BETWEEN 1 AND 6
IF EDGE NOT EXIST ENTER 1000...1000
ENTER THE SHORTEST PATH BETWEEN 2 AND 3
IF EDGE NOT EXIST ENTER 1000...9
ENTER THE SHORTEST PATH BETWEEN 2 AND 4
IF EDGE NOT EXIST ENTER 1000...3
ENTER THE SHORTEST PATH BETWEEN 2 AND 5
IF EDGE NOT EXIST ENTER 1000...1000
ENTER THE SHORTEST PATH BETWEEN 2 AND 6
IF EDGE NOT EXIST ENTER 1000...1000
ENTER THE SHORTEST PATH BETWEEN 3 AND 4
IF EDGE NOT EXIST ENTER 1000...4
ENTER THE SHORTEST PATH BETWEEN 3 AND 5
IF EDGE NOT EXIST ENTER 1000...5
ENTER THE SHORTEST PATH BETWEEN 3 AND 6
IF EDGE NOT EXIST ENTER 1000...
```

```
ENTER THE SHORTEST PATH BETWEEN 2 AND 6
 IF EDGE NOT EXIST ENTER 1000...1000
ENTER THE SHORTEST PATH BETWEEN 3 AND 4
 IF EDGE NOT EXIST ENTER 1000...4
ENTER THE SHORTEST PATH BETWEEN 3 AND 5
 IF EDGE NOT EXIST ENTER 1000...5
ENTER THE SHORTEST PATH BETWEEN 3 AND 6
 IF EDGE NOT EXIST ENTER 1000...
1000
ENTER THE SHORTEST PATH BETWEEN 4 AND 5
 IF EDGE NOT EXIST ENTER 1000...13
ENTER THE SHORTEST PATH BETWEEN 4 AND 6
 IF EDGE NOT EXIST ENTER 1000...15
ENTER THE SHORTEST PATH BETWEEN 5 AND 6
 IF EDGE NOT EXIST ENTER 1000...2

COST MATRIX
1000    1       12      1000    1000    1000
1       1000    9       3       1000    1000
12      9       1000    4       5       1000
1000    3       4       1000    13      15
1000    1000    5       13      1000    2
1000    1000    1000    15      2       1000

ENTER THE SOURCE VERTEX..._
ENTER THE SHORTEST PATH BETWEEN 4 AND 5
 IF EDGE NOT EXIST ENTER 1000...13
ENTER THE SHORTEST PATH BETWEEN 4 AND 6
 IF EDGE NOT EXIST ENTER 1000...15
ENTER THE SHORTEST PATH BETWEEN 5 AND 6
 IF EDGE NOT EXIST ENTER 1000...2

COST MATRIX
1000    1       12      1000    1000    1000
1       1000    9       3       1000    1000
12      9       1000    4       5       1000
1000    3       4       1000    13      15
1000    1000    5       13      1000    2
1000    1000    1000    15      2       1000

ENTER THE SOURCE VERTEX...1

RESULT

DISTANCE OF 1 IS...0
DISTANCE OF 2 IS...1
DISTANCE OF 3 IS...8
DISTANCE OF 4 IS...4
DISTANCE OF 5 IS...13
DISTANCE OF 6 IS...15
```

PROGRAM

```
#include<iostream.h>
#include <conio.h>
class myclass
{
int a,b;
public:
myclass();
~myclass();
void show();
};
myclass::myclass()
{
cout<<"In constructor\n";
a = 30;
b = 20;
}
myclass::~myclass()
{
cout<<"Destructing...\n";
}
void myclass::show()
{
cout<< "A =" << a <<endl<< "B =" << b <<endl;
cout<< "Sum is " <<a+b<<endl;
}
int main()
{
myclass ob;
ob.show();
getch();
return 0;
}
```

OUTPUT

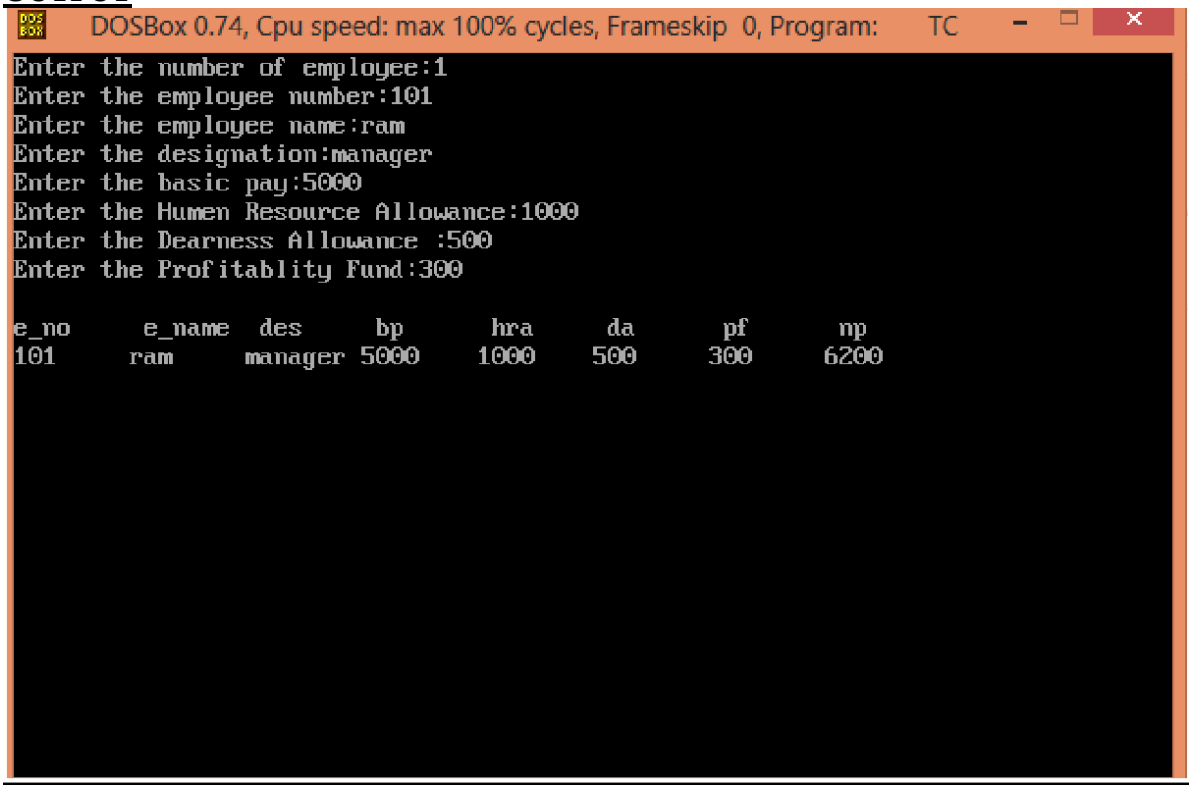

DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Program: TC
Enter the number of values to be pushed into queue
1
Enter the value to be entered into queue
2
Removed Values
2
In constructor
A =30
B =20
Sum is 50

PROGRAM

```
#include<iostream.h>
#include<conio.h>
class emp
{
  public:
    int eno;
    char name[20],des[20];
    void get()
    {
          cout<<"Enter the employee number:";
          cin>>eno;
          cout<<"Enter the employee name:";
          cin>>name;
          cout<<"Enter the designation:";
          cin>>des;
    }
};
class salary:public emp
{
    float bp,hra,da,pf,np;
  public:
    void get1()
    {
          cout<<"Enter the basic pay:";
          cin>>bp;
          cout<<"Enter the Humen Resource Allowance:";
          cin>>hra;
          cout<<"Enter the Dearness Allowance :";
          cin>>da;
          cout<<"Enter the Profitablity Fund:";
          cin>>pf;
    }
    void calculate()
    {
          np=bp+hra+da-pf;
    }
    void display()
    {
  cout<<eno<<"\t"<<name<<"\t"<<des<<"\t"<<bp<<"\t"<<hra<<"\t"<<da<<"\t"<<pf<<"\t"<<np<<"\n";
    }
};
void main()
{
   int i,n;
```

```
    char ch;
    salary s[10];
    clrscr();
    cout<<"Enter the number of employee:";
    cin>>n;
    for(i=0;i<n;i++)
    {
            s[i].get();
            s[i].get1();
            s[i].calculate();
    }
    cout<<"\ne_no \t e_name\t des \t bp \t hra \t da \t pf \t np \n";
    for(i=0;i<n;i++)
    {
            s[i].display();
    }
    getch();
}
```

OUTPUT


DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Program: TC
Enter the number of employee:1
Enter the employee number:101
Enter the employee name:ram
Enter the designation:manager
Enter the basic pay:5000
Enter the Humen Resource Allowance:1000
Enter the Dearness Allowance :500
Enter the Profitablity Fund:300
e_no e_name des bp hra da pf np
101 ram manager 5000 1000 500 300 6200

PROGRAM

```
#include<iostream.h>
#include<conio.h>
#include<string.h>
class student
{
protected:
int regno;
char name[20],dept[5];
public:
void getdata()
{
cout<<"\n enter the register number :";
cin>>regno;
cout<<"\n enter the name :";
cin>>name;
cout<<"\n enter the department :";
cin>>dept;
}
};
class marks: public student
{
protected:
int m1,m2,m3,m4,m5;
public:
void getmarks()
{
cout<<"\n enter the mark 1 :";
cin>>m1;
cout<<"\n enter the mark2 :";
cin>>m2;
cout<<"\n enter the mark3 :";
cin>>m3;
cout<<"\n enter the mark 4:";
cin>>m4;
cout<<"\n enter the mark 5:";
cin>>m5;
}};
class result: public marks
{
private:
int total;
float average;
```

```
char result[6];
public:
void calculate()
{
total=m1+m2+m3+m4+m5;
average=float(total)/5;
if(m1>49&& m2>49&& m3>49&& m4>49&& m5>49)
{
strcpy(result,"pass");
}
else
{
strcpy(result,"fail");
}
}
void display()
{
cout <<"\n\t\t\STUDENT DETAILS";
cout<<"\n register number :"<<regno;
cout<<"\nname:"<<name;
cout<<"\ndepartment :"<<dept ;
cout<<"\nmark1:"<<m1;
cout<<"\nmark2:"<<m2;
cout<<"\nmark3:"<<m3;
cout<<"\nmark4:"<<m4;
cout<<"\nmark5:"<<m5;
cout<<"\ntotal:"<<total;
cout<<"\naverage:"<<average;
cout<<"\nresult:"<<result;
}};
int main()
{
clrscr();
result r;
cout<<"\n\t\tSTUDENT INFORMATION ";
r.getdata();
r.getmarks();
r.calculate();
r.display();
getch();
return 0;
}
```

OUTPUT

```
 enter the department :EEE

 enter the mark 1 :89

 enter the mark2 :77

 enter the mark3 :67

 enter the mark 4:97

 enter the mark 5:65

                student details
 register number :101
name:ram'
department :EEE
mark1:89
mark2:77
mark3:67
mark4:97
mark5:65
total:395
average:79
result:pass_
```

PROGRAM

```
#include<iostream.h>
#include<conio.h>
#include<string.h>
class student
{
protected:
int regno;
char name[20],dept[5];
public:
void getdata()
{
cout<<"\n enter the register number :";
cin>>regno;
cout<<"\n enter the name :";
cin>>name;
cout<<"\n enter the department :";
cin>>dept;
}
};
class marks: public student
{
protected:
int m1,m2,m3,m4,m5;
public:
void getmarks()
{
cout<<"\n enter the mark 1 :";
cin>>m1;
cout<<"\n enter the mark2 :";
cin>>m2;
cout<<"\n enter the mark3 :";
cin>>m3;
cout<<"\n enter the mark 4:";
cin>>m4;
cout<<"\n enter the mark 5:";
cin>>m5;
}};
class result: public marks
{
private:
int total;
float  average;
char result[6];
public:
```

```
void calculate()
{
total=m1+m2+m3+m4+m5;
average=float(total)/5;
if(m1>49&& m2>49&& m3>49&& m4>49&& m5>49)
{
strcpy(result,"pass");
}
else
{
strcpy(result,"fail");
}
}
void display()
{
cout <<"\n\t\t\STUDENT DETAILS";
cout<<"\n register number :"<<regno;
cout<<"\nname:"<<name;
cout<<"\ndepartment :"<<dept ;
cout<<"\nmark1:"<<m1;
cout<<"\nmark2:"<<m2;
cout<<"\nmark3:"<<m3;
cout<<"\nmark4:"<<m4;
cout<<"\nmark5:"<<m5;
cout<<"\ntotal:"<<total;
cout<<"\naverage:"<<average;
cout<<"\nresult:"<<result;
}};
int main()
{
clrscr();
result r;
cout<<"\n\t\tSTUDENT INFORMATION ";
r.getdata();
r.getmarks();
r.calculate();
r.display();
getch();
return 0;
}
```

OUTPUT

```
enter the department :EEE

enter the mark 1 :98

enter the mark2 :78

enter the mark3 :67

enter the mark 4:56

enter the mark 5:99

                STUDENT DETAILS
 register number :101
name:ram
department :EEE
mark1:98
mark2:78
mark3:67
mark4:56
mark5:99
total:398
average:79.599998
result:pass
```

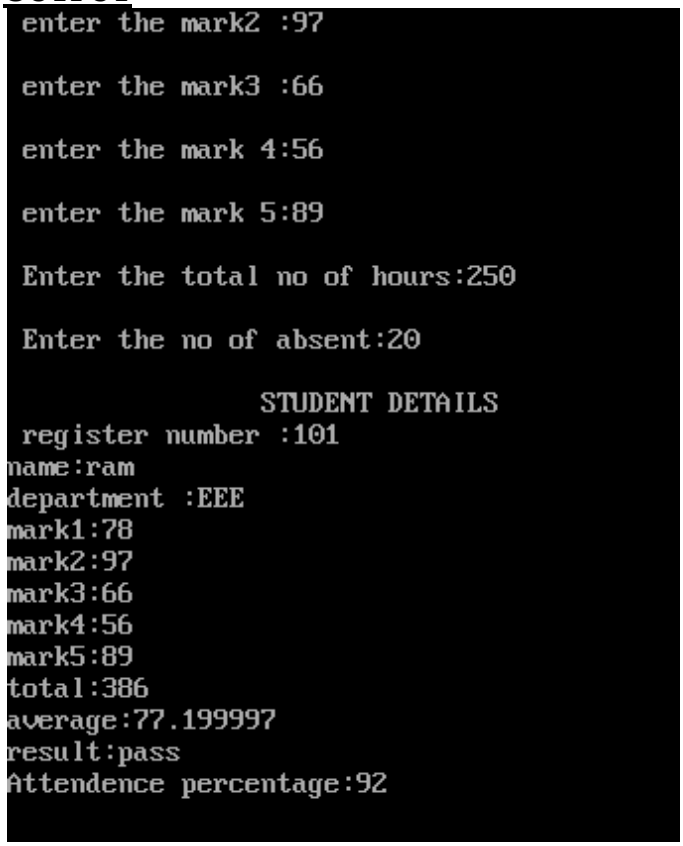
PROGRAM

```
#include<iostream.h>
#include<conio.h>
#include<string.h>
class student
{
protected:
int regno;
char name[20],dept[5];
public:
void getdata()
{
cout<<"\n enter the register number :";
cin>>regno;
cout<<"\n enter the name :";
cin>>name;
cout<<"\n enter the department :";
cin>>dept;
}
};
class marks: public student
{
protected:
int m1,m2,m3,m4,m5;
public:
void getmarks()
{
cout<<"\n enter the mark 1 :";
cin>>m1;
cout<<"\n enter the mark2 :";
cin>>m2;
cout<<"\n enter the mark3 :";
cin>>m3;
cout<<"\n enter the mark 4:";
cin>>m4;
cout<<"\n enter the mark 5:";
cin>>m5;
}
};
class attendance
{
protected:
int totalhours,absent ,present ;
```

```
float percentage;
public:
void getattendance()
{
cout<<"\n Enter the total no of hours:";
cin>>totalhours;
cout<<"\n Enter the no of absent:";
cin>>absent;
}
};
class result: public marks, public attendance
{
private:
int total;
float  average;
char result[6];
public:
void calculate()
{
total=m1+m2+m3+m4+m5;
average=float(total)/5;
if(m1>49&& m2>49&& m3>49&& m4>49&& m5>49)
{
strcpy(result,"pass");
}
else
{
strcpy(result,"fail");
}
present=totalhours-absent;
percentage=float(present)/totalhours;
percentage=percentage*100;
}
void display()
{
cout <<"\n\t\t\STUDENT DETAILS";
cout<<"\n register number :"<<regno;
cout<<"\nname:"<<name;
cout<<"\ndepartment :"<<dept ;
cout<<"\nmark1:"<<m1;
cout<<"\nmark2:"<<m2;
cout<<"\nmark3:"<<m3;
cout<<"\nmark4:"<<m4;
cout<<"\nmark5:"<<m5;
cout<<"\ntotal:"<<total;
```

```
cout<<"\naverage:"<<average;
cout<<"\nresult:"<<result;
cout<<"\nAttendence percentage:"<<percentage;
}
};
int main()
{
clrscr();
result r;
cout<<"\n\t\TSTUDENT INFORMATION ";
r.getdata();
r.getmarks();
r.getattendance();
r.calculate();
r.display();
getch();
return 0;
}
```

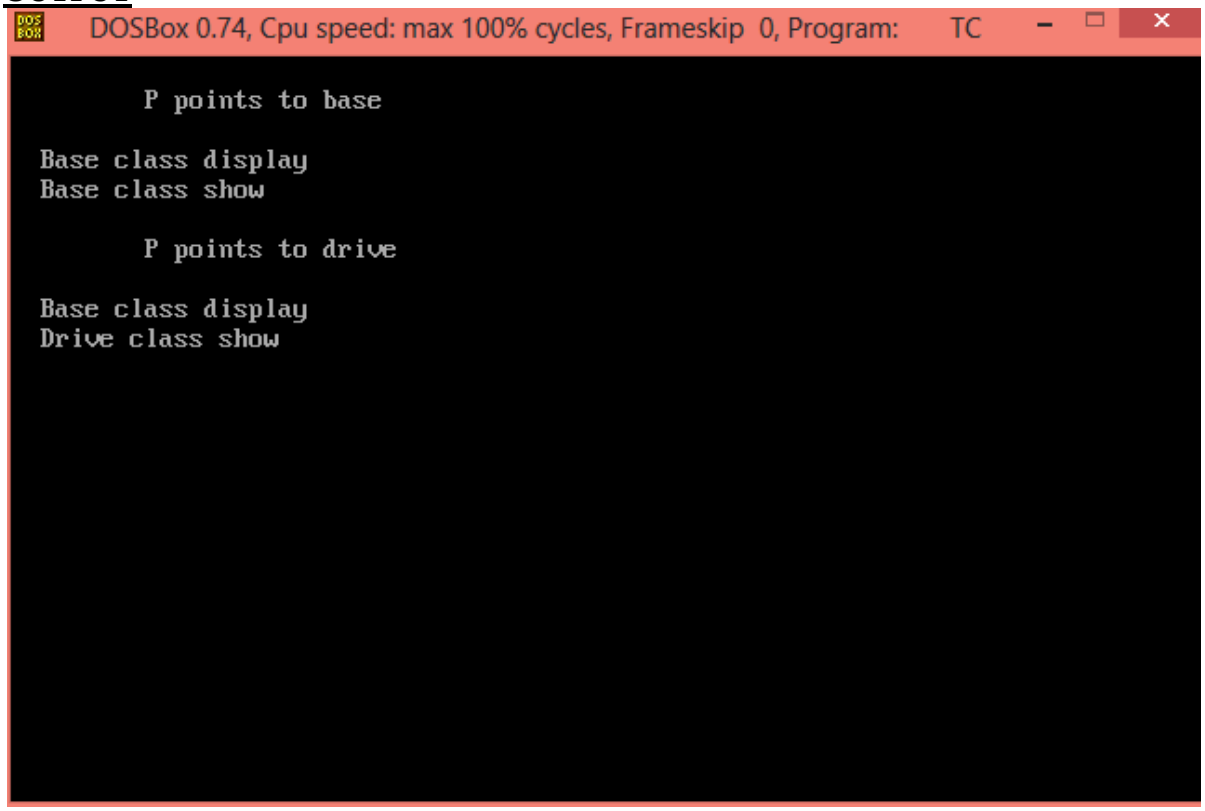
OUTPUT


enter the mark2 :97
enter the mark3 :66
enter the mark 4:56
enter the mark 5:89
Enter the total no of hours:250
Enter the no of absent:20
STUDENT DETAILS
register number :101
name:ram
department :EEE
mark1:78
mark2:97
mark3:66
mark4:56
mark5:89
total:386
average:77.199997
result:pass
Attendence percentage:92

PROGRAM

```
#include<iostream.h>
#include<conio.h>
class base
{
   public:
     virtual void show()
     {
                    cout<<"\n  Base class show";
     }
     void display()
     {
              cout<<"\n  Base class display" ;
     }};
 class drive:public base
{
   public:
     void display()
     {
              cout<<"\n  Drive class display";
     }
     void show()
     {
              cout<<"\n  Drive class show";
     }};

void main()
{
   clrscr();
   base obj1;
   base *p;
   cout<<"\n\t P points to base\n" ;
     p=&obj1;
   p->display();
   p->show();
     cout<<"\n\n\t P points to drive\n";
   drive obj2;
   p=&obj2;
   p->display();
   p->show();
   getch();
}
```

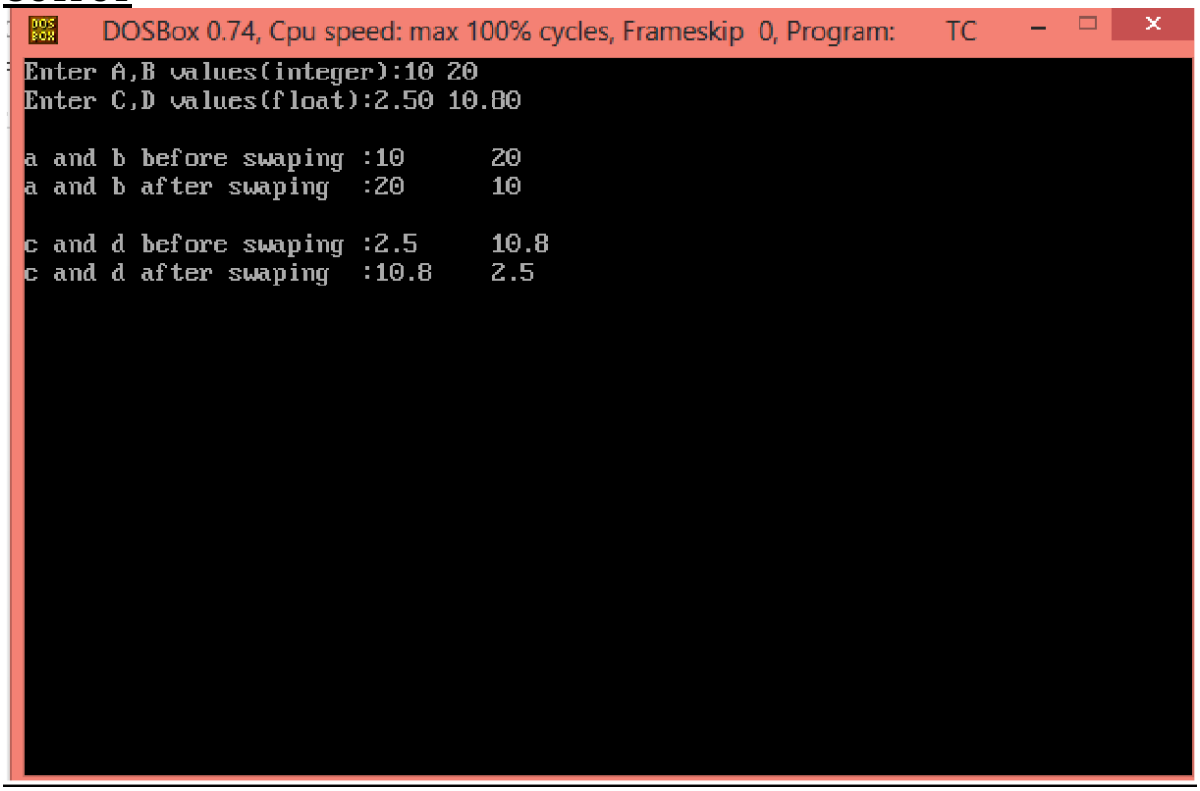
OUTPUT


DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Program: TC
P points to base
Base class display
Base class show
P points to drive
Base class display
Drive class show

PROGRAM

```
#include<iostream.h>
#include<conio.h>
template<class t>
void swap(t &x,t &y)
{
  t temp=x;
  x=y;
  y=temp;
}
 void fun(int a,int b,float c,float d)
{
  cout<<"\na and b before swaping :"<<a<<"\t"<<b;
  swap(a,b);
  cout<<"\na and b after swaping :"<<a<<"\t"<<b;
  cout<<"\n\nc and d before swaping :"<<c<<"\t"<<d;
  swap(c,d);
  cout<<"\nc and d after swaping  :"<<c<<"\t"<<d;
}
void main()
{
   int a,b;
   float c,d;
   clrscr();
   cout<<"Enter A,B values(integer):";
   cin>>a>>b;
   cout<<"Enter C,D values(float):";
   cin>>c>>d;
   fun(a,b,c,d);
   getch();
}
```

OUTPUT


DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Program: TC
Enter A,B values(integer):10 20
Enter C,D values(float):2.50 10.80
a and b before swaping :10 20
a and b after swaping :20 10
c and d before swaping :2.5 10.8
c and d after swaping :10.8 2.5

PROGRAM

```
#include<iostream.h>
#include<conio.h>
#include<exception.h>
void main()
{
int a,b,c;
float d;
clrscr();
cout<<"Enter the values of a:";
cin>>a;
cout<<"Enter the value of b:";
cin>>b;
cout<<"Enter the value of c:";
cin>>c;
try
{
if((a-b)!=0)
{
d=c/(a-b);
cout<<"result is:"<<d;
}
else
{
throw(a-b);
}
}
catch(int i)
{
cout<<"Answer is infinite because a-b is:"<<i;
}
getch();
}
```

OUTPUT

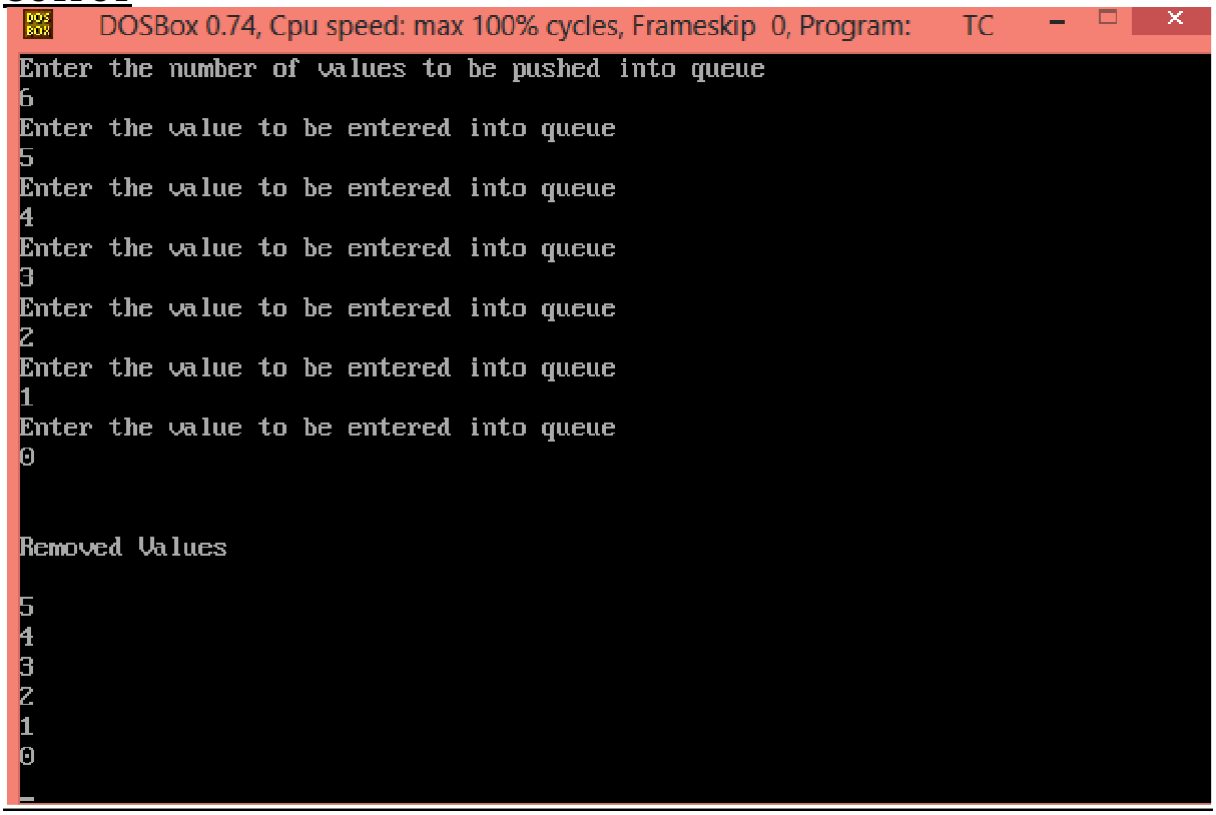
```
Enter the value for a: 20
Enter the value for b:20
Enter the value for c: 40

Answer is infinite because a-b is :0
```

PROGRAM

```
#include<iostream.h>
#include<conio.h>
struct node
{
int data;
node *next;
}*front = NULL,*rear = NULL,*p = NULL,*np = NULL;
void push(int x)
{
np = new node;
np->data = x;
np->next = NULL;
if(front == NULL)
{
front = rear = np;
rear->next = NULL;
}
else
{
rear->next = np;
rear = np;
rear->next = NULL;
}
}
int remove()
{
int x;
if(front == NULL)
{
cout<<"empty queue\n";
}
else
{
p = front;
x = p->data;
front = front->next;
delete(p);
return(x);
}
}
int main()
{
```

```
int n,c = 0,x;
clrscr();
cout<<"Enter the number of values to be pushed into queue\n";
cin>>n;
while (c < n)
{
cout<<"Enter the value to be entered into queue\n";
cin>>x;
push(x);
c++;
}
cout<<"\n\nRemoved Values\n\n";
while(1)
{
if (front != NULL)
cout<<remove()<<endl;
else
break;
}
getch();
}
```

OUTPUT


DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Program: TC
Enter the number of values to be pushed into queue
6
Enter the value to be entered into queue
5
Enter the value to be entered into queue
4
Enter the value to be entered into queue
3
Enter the value to be entered into queue
2
Enter the value to be entered into queue
1
Enter the value to be entered into queue
0
Removed Values
5
4
3
2
1
0