/*Assignment 01 :- write a program to implement GCD and LCM Lab:DAA */

```
#include<iostream.h>
#include<conio.h>
void main()
{
clrscr();
int a,b;
cout<<"Enter Two Numbers For Finding GCD :\n";
cin>>a>>b;
int p=a*b;
while(a!=b)
{
if(a>b)
  a=a-b;
else
  b=b-a;
}
cout<<"GCD = "<<a;
cout<<"\nLCM = "<<p/a;
getch();
}
```

```
/* OUTPUT :-

Enter two number to find GCD & LCM  :
6
4
GCD=2
LCM=12
*/
```

/*Assignment 02 :- Program for finding Maximum using rules removal of recursion Lab:DAA*/

```
#include <iostream.h>
#include <conio.h>
#include<stdlib.h>
#include<time.h>
int a[200],n,k;
class maximum
{
private:
int i;
public:
void read();
int max(int);
void print();
};
void maximum::read()
{
 cout<<"...Program for finding Maximum using rules  removal of  recursion...";
 cout<<"\n\nEnter the number of element :- ";
 cin>>n;
 for(int p=1;p<=n;p++)
 {
  if(p%9==0)
  cout<<endl;
  a[p]=rand();
  cout<<a[p]<<"\t";
 }
}
int maximum::max(int i)
{
 int add;
 int top=0,st[400],j;
 l1:
 if(i<n)
 {
 top=top+1;
 st[top]=i;
 top=top++;
 st[top]=2;
 i=i+1;
 goto l1;
 l2:
 j=st[top];
 top=top-1;
 if(a[i]>a[j])
 k=i;
 else
 k=j;
 }
 else
 k=n;
 if(top==0)
 return k;
```

```
 else
 add=st[top];
 top=top-1;
 i=st[top];
 top=top-1;
 top=top+1;
 st[top]=k;
 if(add==2)
 goto l2;
}
void maximum::print()
{
  cout<<"\n Max Position :- "<<k<<endl;
  cout<<"\n Max Element :- "<<a[k];
}
 main()
{
  clrscr();
  maximum m;
  m.read();
  m.max(1);
  m.print();
  getch();
  return 0;
}

/* OUTPUT :-
  ...Program for finding Maximum using rules  removal of  recursion...

Enter the number of element :- 10
346     130     10982   1090    11656   7117    17595   6415
22948   31126
 Max Position :- 10

 Max Element :- 31126
 */
```

/* Assignment 03:- Program for Searching An element Using Rules Of Removal Of Recursion. Lab:DAA*/

```
#include<iostream.h>
#include<conio.h>
#include<stdlib.h>
#include<time.h>
class array
{
  private:
  int *a,size,*stack,top,p,item;
 public:
array()
{
 cout<<"\n Enter the number of elements :- ";
 cin>>size;
 a=new int[size];
stack=new int[size*2];
top=-1;
}
void getdata();
void display();
int search(int);
};
void array::getdata()
{
for(int i=0;i<size;i++)
 {
   if(i%8==0)
   cout<<"\n";
   a[i]=random(100);
  cout<<a[i]<<" ";
}
cout<<"\n Enter the initial Position :-";
cin>>p;
cout<<"\n Enter the item to be search :-";
cin>>item;
}
int array::search(int b)
{
 int pos,addr,i;
while(b<size)
{
 top++;
stack[top]=b;
top++;
stack[top]=2;
b++;
}
pos=-1;
do
{
addr=stack[top];
top--;
```

```
i=stack[top];
top--;
if(addr==2 && a[i]==item)
{
 if(pos== -1)
 cout<<"\n Element is found at Position :";
 else
 cout<<" , ";
 pos=i+1;
 cout<<pos;
}
}while(top>0);
return pos;
}
void array::display()
{
 if(search(p-1) == -1)
 cout<<"\n Item Is not Found In the array...";
}

void main()
 {
  clrscr();
  clock_t e,s;
  array obj;
  obj.getdata();
  s=clock();
  obj.display();
  e=clock();
  cout<<"\n The Time Complexity is :-"<<((e-s)/CLK_TCK);
  getch();
}

/*  OUTPUT :-
 Program for Searching An element Using Rules Of Removal Of Recurssion.
 Enter the number of elements :- 10
1 0 33 3 35 21 53 19
70 94 27 44 10 69 56 4
16 81 68 76 82 95 21 42
95 83 92 81 45 60 66 59
54 72 11 40 12 67 47 49
56 34 86 26 17 42 69 16
53 64 62 0 78 26 46 38
37 58 60 27 17 80 29 33
40 24 41 5 49 98 0 40
6 7 25 97 35 40 19 19
21 24 75 88 90 73 46 53
3 13 45 48 59 25 11 70
64 88 87 4
 Enter the initial Position :-1
 Enter the item to be search :-94
 Element is found at Position :10
      The Time Complexity is :-0   */
```

/*Assignment 04:- Program For Binomial Co-efficient .. Lab:DAA */

```
#include<iostream.h>
#include<conio.h>
int binomial(int a,int b)
{
 if((a==b) || (b==0))
  return 1;
 else
  return(binomial(a-1,b-1)+binomial(a-1,b));
}
void main()
{
 clrscr();
 int n;
 cout<<"Program For Binomial Co-efficient ..\n\n";
 cout<<"Enter level :-";
 cin>>n;
 for(int i=0;i<=n;i++)
 cout<<binomial(n,i)<<"\t";
 getch();
}
/*     OUTPUT
Program For Binomial Co-efficient ..Enter level :-2
1       2       1
*/
```

/*Assignment 05 :- Implement program for Binomial Coefficent by Using rules of removal of recursion Lab:DAA */

```
#include<iostream.h>
#include<conio.h>
int b=0;
class bin
{
int n,m,top;
public:
 void read();
 int binomial(int n,int m);
 int topcheck();
};
void bin::read()
{
 cout<<"\nprogram for Binomial Coefficent by Using rules of removal of recursion\n\n";
 cout<<"\n Enter the Value n:-";
 cin>>n;
 cout<<"\n Enter the Value m :-";
 cin>>m;
 binomial(n,m);
 cout<<endl<<"Binomial Coefficint Is :-"<<b;
}
int bin::topcheck()
{
 if(top==0)
 return(1);
 return(0);
}
int bin::binomial(int n,int m)
{
 int st[100];
 top=0;
 L1:
  if((n==m)||(m==0))
  {
   b=b+1;
   if(topcheck())
   {
    return(b);
   }
   else
   goto L2;
   }
   else
   {
    top=top+1;
    st[top]=n;
    top=top+1;
    st[top]=m;
    n=n-1;
```

```
   m=m-1;
   goto L1;
   }
   L2:
    m=st[top];
    top--;
    n=st[top];
    top--;
    n--;
    goto L1;
}
void main()
{
 bin b1;
 clrscr();
 b1.read();
 getch();
}


/* OUTPUT:-
program for Binomial Coefficent by Using rules of removal of recursion
Enter the Value n:-3
Enter the Value m :-2
Binomial Coefficint Is :-3
 */
```

/* Assignment 06 :- Write a program for Max heap insert.. Lab :DAA */

```
#include<iostream.h>
#include<conio.h>
#include<stdlib.h>
#include<time.h>
class heap
{
 private:
          int a[1000],n;
 public:
          void getdata();
          void insert();
          void disp();
};
void heap::getdata()
{
  cout<<"\n...Program for Max Heap Insert...\n\n";
  cout<<"Enter the Size := " ;
  cin>>n;
  for(int i=1;i<=n;i++)
  {
    a[i]=random(20000);
  }
}
void heap::insert()
{
  for(int j=1;j<=n;j++)
  {
   int i=j;
   int item=a[i];
   while((i>1) && (a[i/2]<item))
   {
          a[i]=a[i/2];
          i=i/2;
   }
  a[i]=item;
  }
}
void heap::disp()
{
 for(int i=1;i<=n;i++)
 {
           if(i%8==0)
           cout<<"\n";
           cout<<a[i]<<"\t";
 }
}
void main()
{
 clrscr();
 clock_t e,s;
 heap h;
```

```
 h.getdata();
 cout<<"\n\n Befor Insert :: ";
 h.disp();
 s=clock();
 h.insert();
 e=clock();
 cout<<"\n\n After Insert ::\n";
 h.disp();
 cout<<"\n\nThe Time Complexity is ::"<<((e-s)/CLK_TCK);
 getch();
}

/* OUTPUT :-

...Program for Max Heap Insert...

Enter the Size := 10
 Befor Insert :: 211    79     6702    665     7114    4343    10739
3915    14006   18997
 After Insert ::
18997   14006   7114    6702    10739   211     4343
79      3915    665

The Time Complexity is ::0
*/
```

/* Assignment 07 :- Write a program for Min Heap Insert.. Lab:DAA */

```
#include<iostream.h>
#include<conio.h>
#include<stdlib.h>
#include<time.h>
class heap
{
 private:
        int a[1000],n;
 public:
        void getdata();
        void insert();
        void disp();
};
void heap::getdata()
{
  cout<<"\n...Program for Min Heap Insert...\n\n";
  cout<<"Enter the Size := " ;
  cin>>n;
  for(int i=1;i<=n;i++)
  {
    a[i]=random(20000);
  }
}
void heap::insert()
{
  for(int j=1;j<=n;j++)
  {
   int i=j;
   int item=a[i];
   while((i>1) && (a[i/2] > item))
   {
        a[i]=a[i/2];
        i=i/2;
   }
  a[i]=item;
  }
}
void heap::disp()
{
 for(int i=1;i<=n;i++)
 {
         if(i%8==0)
         cout<<"\n";
         cout<<a[i]<<"\t";
 }
}
void main()
{
 clrscr();
 clock_t e,s;
 heap h;
 h.getdata();
```

```
 cout<<"\n\n Befor Insert :: ";
 h.disp();
 s=clock();
 h.insert();
 e=clock();
 cout<<"\n\n After Insert ::\n";
 h.disp();
 cout<<"\n\nThe Time Complexity is ::"<<((e-s)/CLK_TCK);
 getch();
}

/* OUTPUT :-

...Program for Min Heap Insert...
Enter the Size := 10
 Befor Insert :: 211    79     6702    665     7114    4343    10739
3915    14006   18997
 After Insert ::
79      211     4343    665     7114    6702    10739
3915    14006   18997

The Time Complexity is ::0
*/
```

/* Assignment 08:-Program for MAX heap using Heapify/Adjust Lab:DAA */

```
#include<iostream.h>
#include<conio.h>
#include<stdlib.h>
#include<time.h>
int a[1000],n;
class heap
{
        int i,j,item;
        public:
         void get();
         void show();
         void adjust(int [],int i,int j);
         void heapify(int [],int);

};
void heap::get()
{
        cout<<"Program for Max Heap using Heapify/Adjust..." ;
        cout<<"\n\nEnter the size of array :";
        cin>>n;
        for(i=1;i<=n;i++)
        a[i]=random(1000);
}
void heap::show()
{
         cout<<"\n The element is:= \n";
         for(i=1;i<=n;i++)
         cout<<a[i]<<"\t";
}
void heap::adjust(int a[],int i,int n)
{
        j=2*i;
        item=a[i];
        while(j<=n)
        {
         if((j<n) && (a[j]<a[j+1]))
         j++;
         if(item>=a[j])
         break;
         a[j/2]=a[j];
         j=2*j;
        }
        a[j/2]=item;
}
void heap::heapify(int a[],int n)
{
        for(i=n/2;i>=1;i--)
        adjust(a,i,n);
}
void main()
{
        clrscr();
```

```
        clock_t e,s;
        heap h;
        h.get();
        h.show();
        s=clock();
        h.heapify(a,n);
        e=clock();
        h.show();
        cout<<"\n The Time complexityis => "<<(e-s)/CLK_TCK;
        getch();
}

 /* OUTPUT :-
 Program for Max Heap using Heapify/Adjust...
Enter the size of array :7
 The element is:=
10      3       335     33      355     217     536
 The element is:=
536     355     335     33      3       217     10
 The Time complexityis => 0
 */
```

/* Assignment 09:-Program for MIN heap using Heapify/Adjust Lab:DAA */

```
#include<iostream.h>
#include<conio.h>
#include<stdlib.h>
#include<time.h>
int a[1000],n;
class heap
{
        int i,j,item;
        public:
         void get();
         void show();
         void adjust(int [],int i,int j);
         void heapify(int [],int);

};
void heap::get()
{
        cout<<"Program for Min Heap using Heapify/Adjust..." ;
        cout<<"\n\nEnter the size of array :";
        cin>>n;
        for(i=1;i<=n;i++)
        a[i]=random(1000);
}
void heap::show()
{
         cout<<"\n The element is:= \n";
         for(i=1;i<=n;i++)
         cout<<a[i]<<"\t";
}
void heap::adjust(int a[],int i,int n)
{
        j=2*i;
        item=a[i];
        while(j<=n)
        {
         if((j<n) && (a[j]>a[j+1]))
         j++;
         if(item<=a[j])
         break;
         a[j/2]=a[j];
         j=2*j;
        }
        a[j/2]=item;
}
void heap::heapify(int a[],int n)
{
        for(i=n/2;i>=1;i--)
        adjust(a,i,n);
}
void main()
{
        clrscr();
```

```
        clock_t e,s;
        heap h;
        h.get();
        h.show();
        s=clock();
        h.heapify(a,n);
        e=clock();
        h.show();
        cout<<"\n The Time complexityis => "<<(e-s)/CLK_TCK;
        getch();
}

//OUTPUT :-
/*
Program for Min Heap using Heapify/Adjust...

Enter the size of array :7

 The element is:=
10      3       335     33      355     217     536
 The element is:=
3       10      217     33      355     335     536
 The Time complexityis => 0
*/
```

/* Assignment 10:- Write a program for HeapSort Ascending Using Insert/Delete Lab:DAA */

```
#include<iostream.h>
#include<conio.h>
#include<time.h>
#include<stdlib.h>
class heap
{
        int item,i,b[1000];
        private:
        int a[1000],n;
        public:
        void getdata();
        int delheap();
        void insert(int);
        void adjust(int[],int,int);
        void heapsort();
        void disp();
        void disp1();
};
void heap::getdata()
{
        cout<<"Program for HeapSort for Descending using Insert/Delete...\n";
        cout<<"\nEnter Size :=";
        cin>>n;
        for(int i=1;i<=n;i++)
        {
          a[i]=random(20000);
        }
}
void heap::insert(int i)
{
        int item=a[i];
        while((i>1) && (a[i/2]>item))
        {
         a[i]=a[i/2];
         i=i/2;
        }
        a[i]=item;
}
void heap::adjust(int a[],int i,int n)
{
        int j=2*i;
        item=a[i];
        while(j<=n)
        {
          if((j<n) && (a[j] > a[j+1]))
          j=j+1;
          if(item<=a[j])
          break;
          a[j/2]=a[j];
          j=2*j;
        }
        a[j/2]=item;
```

```
}
int heap::delheap()
{
        if(n==0)
        {
          cout<<"heap is empty :";
        }
        int x=a[1];
        a[1]=a[i];
        adjust(a,1,i-1);
        return x;
}
void heap::heapsort()
{
        for(i=1;i<=n;i++)
        insert(i);
        disp();
        for(i=n;i>=1;i--)
        b[i]=delheap();
}
void heap::disp()
{
        for(i=1;i<=n;i++)
        {
          if(i%8==0)
          cout<<"\n";
          cout<<a[i]<<"\t";
        }
}
void heap::disp1()
{
        for(i=1;i<=n;i++)
        {
          if(i%8==0)
          cout<<"\n";
          cout<<b[i]<<"\t";
        }
}
void main()
{
        clrscr();
        clock_t e,s;
        heap h;
        h.getdata();
        cout<<"\n\n Before Sort  :\n";
        s=clock();
        h.heapsort();
        e=clock();
        cout<<"\n\n After Sort :\n";
        h.disp1();
        cout<<"\n\n Time Complexity ::"<<((e-s)/(CLK_TCK));
        getch();
}
```

```
/*OUTPUT :
Program for HeapSort for Descending using Insert/Delete...

Enter Size :=50


 Before Sort  :
79      211     2179    665     3596    3273    9020
830     2428    6889    4388    4343    6702    10739   11286
3915    2279    8123    9497    11284   9821    5288    7114
8535    12909   16790   18464   16217   13964   12094   13233
11994   10983   14400   3303    16309   13710   14006   13424
18997   15287   16552   17374   19190   5495    8461    13895
8885    10765   19051

 After Sort :
19190   19051   18997   18464   17374   16790   16552
16309   16217   15287   14400   14006   13964   13895   13710
13424   13233   12909   12094   11994   11286   11284   10983
10765   10739   9821    9497    9020    8885    8535    8461
8123    7114    6889    6702    5495    5288    4388    4343
3915    3596    3303    3273    2428    2279    2179    830
665     211     79

 Time Complexity ::0
*/
```

/*Assignment 11:- Write a program for HeapSort Descending Using Insert/Delete Lab:DAA */

```
#include<iostream.h>
#include<conio.h>
#include<time.h>
#include<stdlib.h>
class heap
{
        int item,i,b[1000];
       private:
        int a[1000],n;
       public:
       void getdata();
       int delheap();
       void insert(int);
       void adjust(int[],int,int);
       void heapsort();
       void disp();
       void disp1();
};
void heap::getdata()
{
       cout<<"Program for HeapSort for Ascending using Insert/Delete...\n";
       cout<<"\nEnter Size :=";
       cin>>n;
       for(int i=1;i<=n;i++)
       {
         a[i]=random(20000);
       }
}
void heap::insert(int i)
{
       int item=a[i];
       while((i>1) && (a[i/2]<item))
       {
        a[i]=a[i/2];
        i=i/2;
       }
       a[i]=item;
}
void heap::adjust(int a[],int i,int n)
{
       int j=2*i;
       item=a[i];
       while(j<=n)
       {
         if((j<n) && (a[j]<a[j+1]))
         j=j+1;
         if(item>=a[j])
         break;
         a[j/2]=a[j];
         j=2*j;
       }
       a[j/2]=item;
```

```
}
int heap::delheap()
{
        if(n==0)
        {
          cout<<"heap is empty :";
        }
        int x=a[1];
        a[1]=a[i];
        adjust(a,1,i-1);
        return x;
}
void heap::heapsort()
{
        for(i=1;i<=n;i++)
        insert(i);
        disp();
        for(i=n;i>=1;i--)
        b[i]=delheap();
}
void heap::disp()
{
        for(i=1;i<=n;i++)
        {
          if(i%8==0)
          cout<<"\n";
          cout<<a[i]<<"\t";
        }
}
void heap::disp1()
{
        for(i=1;i<=n;i++)
        {
          if(i%8==0)
          cout<<"\n";
          cout<<b[i]<<"\t";
        }
}
void main()
{
        clrscr();
        clock_t e,s;
        heap h;
        h.getdata();
        cout<<"\n\n Before Sort  :\n";
        s=clock();
        h.heapsort();
        e=clock();
        cout<<"\n\n After Sort :\n";
        h.disp1();
        cout<<"\n\n Time Complexity ::"<<((e-s)/(CLK_TCK));
        getch();
}
```

```
//OUTPUT
 /*
 Program for HeapSort for Ascending using Insert/Delete...

Enter Size :=50


 Before Sort  :
19190   18997   19051   14400   17374   18464   16217
14006   13710   16552   16309   12909   16790   11286   13233
10983   11994   8123    13424   11284   15287   5495    13895
8535    10765   2179    13964   4343    9020    8885    12094
79      3303    830     2279    3915    2428    6702    9497
665     9821    6889    10739   5288    3596    4388    8461
211     3273    7114

 After Sort :
79      211     665     830     2179    2279    2428
3273    3303    3596    3915    4343    4388    5288    5495
6702    6889    7114    8123    8461    8535    8885    9020
9497    9821    10739   10765   10983   11284   11286   11994
12094   12909   13233   13424   13710   13895   13964   14006
14400   15287   16217   16309   16552   16790   17374   18464
18997   19051   19190

 Time Complexity ::0
*/
```

/* Assignment 12 :- Program for Heapsort Ascending using Adjust/Heapify... Lab:DAA */

```
#include<iostream.h>
#include<conio.h>
#include<stdlib.h>
#include<time.h>
int b[5000];
class heap
{
        public:
          void heapsort(int a[],int n);
          void heapify(int a[],int n);
          void adjust(int a[],int i, int n);
};
void heap::heapsort(int a[],int n)
{
        heapify(a,n);
        for(int i=n;i>=2;i--)
        {
         int t=a[i];
         a[i]=a[1];
         a[1]=t;
         adjust(a,1,i-1);
        }
}
void heap::heapify(int a[],int n)
{
        int i;
        for(i=n/2;i>=1;i--)
        {
         adjust(a,i,n);
        }
}
void heap::adjust(int a[],int i,int n)
{
        int j=2*i;
        int item=a[i];
        while(j<=n)
        {
         if((j<n) && (a[j]<a[j+1]))
         j=j+1;
         if(item>=a[j])
          break;
         else
         {
          a[j/2]=a[j];
          j=2*j;
         }
        }
        a[j/2]=item;
}
void main()
{
```

```
        clrscr();
        clock_t e,s;
        int n,i;
        heap h;
        cout<<"..Program For HeapSort Ascending using Adjust/Heapify..\n\n";
        cout<<"\n Enter Size Of The Array :->";
        cin>>n;
        for(i=1;i<=n;i++)
        {
         if(i%8==0)
         cout<<"\n";
         b[i]=random(n);
         cout<<"\t"<<b[i];
        }
        s=clock();
        h.heapsort(b,n);
        e=clock();
        cout<<"\n After Heap Sorting : \n";
        for(i=1;i<=n;i++)
        {
         if(i%8==0)
         cout<<"\n";
         cout<<"\t"<<b[i];
        }
        cout<<"\n\n The Time Complexity Is :="<<((e-s)/CLK_TCK);
        getch();    }
/*OUTPUT :-
..Program For HeapSort Ascending using Adjust/Heapify..


 Enter Size Of The Array :->50

        0     0     16    1     17    10    26    9
        35    47    13    22    5     34    28    2
        8     40    34    38    41    47    10    21
        47    41    46    40    22    30    33    29
        27    36    5     20    6     33    23    24
        28    17    43    13    8     21    34    8
        26    32
 After Heap Sorting :

        0     0     1     2     2     5     8     8
        9     10    13    16    17    20    20    21
        21    22    22    23    24    26    26    26
        26    27    27    28    28    29    30    32
        33    33    34    34    34    34    36    38
        40    40    40    41    41    41    43    46
        47    47

 The Time Complexity Is :=0
 */
```

/* Assignment 13 :- Program for Heapsort Descending using Adjust/Heapify... Lab:DAA */

```
#include<iostream.h>
#include<conio.h>
#include<stdlib.h>
#include<time.h>
int b[5000];
class heap
{
        public:
          void heapsort(int a[],int n);
          void heapify(int a[],int n);
          void adjust(int a[],int i, int n);
};
void heap::heapsort(int a[],int n)
{
        heapify(a,n);
        for(int i=n;i>=2;i--)
        {
         int t=a[i];
         a[i]=a[1];
         a[1]=t;
         adjust(a,1,i-1);
        }
}
void heap::heapify(int a[],int n)
{
        int i;
        for(i=n/2;i>=1;i--)
        {
         adjust(a,i,n);
        }
}
void heap::adjust(int a[],int i,int n)
{
        int j=2*i;
        int item=a[i];
        while(j<=n)
        {
         if((j<n) && (a[j]>a[j+1]))
         j=j+1;
         if(item<=a[j])
          break;
         else
         {
          a[j/2]=a[j];
          j=2*j;
         }
        }
        a[j/2]=item;
}
void main()
{       clrscr();
        clock_t e,s;
```

```
        int n,i;
        heap h;
        cout<<"..Program For HeapSort Descending using Adjust/Heapify..\n\n";
        cout<<"\n Enter Size Of The Array :->";
        cin>>n;
        for(i=1;i<=n;i++)
        {
         if(i%8==0)
         cout<<"\n";
         b[i]=random(n);
         cout<<"\t"<<b[i];
        }
        s=clock();
        h.heapsort(b,n);
        e=clock();
        cout<<"\n After Heap Sorting : \n";
        for(i=1;i<=n;i++)
        {
         if(i%8==0)
         cout<<"\n";
         cout<<"\t"<<b[i];
        }
        cout<<"\n\n The Time Complexity Is :="<<((e-s)/CLK_TCK);
        getch();
}
/* OUTPUT :-
..Program For HeapSort Descending using Adjust/Heapify..


 Enter Size Of The Array :->50
      0     0     16    1     17    10    26
      9     35    47    13    22    5     34    28
      2     8     40    34    38    41    47    10
      21    47    41    46    40    22    30    33
      29    27    36    5     20    6     33    23
      24    28    17    43    13    8     21    34
      8     26    32
 After Heap Sorting :
      47    47    47    46    43    41    41
      40    40    38    36    35    34    34    34
      33    33    32    30    29    28    28    27
      26    26    24    23    22    22    21    21
      20    17    17    16    13    13    10    10
      9     8     8     8     6     5     5     2
      1     0     0

 The Time Complexity Is :=0
 */
```

/* Assignment 14 :- Program for Union And Find Lab:DAA */

```
#include<iostream.h>
#include<conio.h>
#include<stdlib.h>
int p[100];
class heap
{
        int n,a,b,c,no;
        public:
        void menu();
        void read_e();
        void s_union(int,int);
        int find(int);
        void print();
};
void heap::menu()
{
        int ch;
        cout<<"..... Program for Union And Find .....\n\n";
        cout<<"1:Read_ele 2:Simple union 3:Find 4:Print 5:Exit"<<endl;
        while(ch!=5)
        {
         switch(ch)
         {
          case 1:
              read_e();
              break;
          case 2:
              cout<<"\n Enter root of two set node: ";
              cin>>a>>b;
              s_union(a,b);
              break;
          case 3:
              cout<<"\n Find the node :";
              cin>>no;
              c=find(no);
              cout<<"root node is :";
              cout<<c;
              break;
          case 4:
              print();
              break;
          case 5:
          exit(0);
         }
         cout<<"\n Enter the choice :";
         cin>>ch;
        }
}
void heap::read_e()
{
        cout<<"\n Enter the Number of elements :";
        cin>>n;
```

```
        cout<<"\n Enter Elements :";
        for(int i=1;i<=n;i++)
        cin>>p[i];
}
void heap::s_union(int i,int j)
{
        p[i]=j;
}
int heap::find(int i)
{
        while(p[i]>=0)
        {
         i=p[i];
        }
        return i;
}
void heap::print()
{
        cout<<"\n Union of Two set :";
        for(int i=1;i<=n;i++)
        {
         cout<<p[i]<<" ";
        }
}

void main()
{
        clrscr();
        heap h;
        h.menu();
        getch();
}

/* OUTPUT :-
..... Program for Union And Find .....

1:Read_ele 2:Simple union 3:Find 4:Print 5:Exit

 Enter the choice :1
 Enter the Number of elements :6
 Enter Elements :-1 1 1 -1 4 4
 Enter the choice :2
 Enter root of two set node: 1 4
 Enter the choice :4
 Union of Two set :4 1 1 -1 4 4
 Enter the choice :3
 Find the node :6
root node is :4
 Enter the choice :5
  */
```

/* Assignment 15:- Program for Weighted Union & Collapsing Find. Lab:DAA */

```
#include<conio.h>
#include<iostream.h>
#include<process.h>
class TreeOp
{
        int a[100],n,i,j,k,l,r;
        public:

void get()
{
        cout<<".... Program for Weighted Union & Collapsing Find.... \n\n ";
        cout<<"Enter The size :";
        cin>>n;
        cout<<"Enter the Elements :";
        for(i=1;i<=n;i++)
        {
          cin>>a[i];
        }
        cout<<"\n";
        for(i=1;i<=n;i++)
        {
         cout<<a[i]<<" ";
        }
}
void wunion()
{
        cout<<"\n Enter the two roots for union :\n";
        cin>>j>>k;
        int temp;
        temp=a[j]+a[k];
        if(a[j]>a[k])
        {
          a[j]=k;
          a[k]=temp;
        }
        else
        {
          a[k]=j;
          a[j]=temp;
        }
        cout<<"\n";
         for(i=1;i<=n;i++)
         {
           cout<<a[i]<<" ";
         }
}
void wfind()
{
        cout<<"\n Enter the element to find root :";
        cin>>l;
        r=l;
        while(a[r]>0)
```

```
            {
              r=a[r];
            }
            cout<<"Root is : "<<r;
            while(l!=r)
            {
              int s=a[l];
              a[l]=r;
              l=s;
            }
        }
};
void main()
{
            TreeOp t;
            clrscr();
            t.get();
            int ch;
            while(ch!=3)
            {
              cout<<"\n 1: Weighted Union 2: Collapsing Find 3: Exit ";
              cout<<"\nEnter your choice :";
              cin>>ch;
              switch(ch)
              {
               case 1:t.wunion();
                        break;
               case 2:t.wfind();
                        break;
               case 3:exit(0);
                        break;
              }
            }
              getch();
}

 /*OUTPUT :-
 .... Program for Weighted Union & Collapsing Find....

 Enter The size :10
Enter the Elements : -4 1 -4 1 3 2 3 5 -2 9
-4 1 -4 1 3 2 3 5 -2 9
 1: Weighted Union 2: Collapsing Find 3: Exit
Enter your choice :2
 Enter the element to find root :8
Root is : 3
 1: Weighted Union 2: Collapsing Find 3: Exit
Enter your choice :1
 Enter the two roots for union :
1 3
-8 1 1 1 3 2 3 3 -2 9
 1: Weighted Union 2: Collapsing Find 3: Exit
Enter your choice :3   */
```

/* Assignment 16 :- Program for searching element from given array using Binary Search. Lab:DAA*/

```
#include<iostream.h>
#include<conio.h>
#include<stdlib.h>
#include<time.h>
int a[1000];
class binary
{
int n,l,h,mid,x;
public:
 void get();
 void put();
 int bsearch(int x);
 void sort();
};
void binary::get()
{
cout<<"\n Enter the number of elements :" ;
cin>>n;
for(int i=1;i<=n;i++)
a[i]=random(20000);
}
void binary::put()
{
for(int i=1;i<=n;i++)
{
if(i%8==0)
cout<<endl;
cout<<a[i]<<'\t';
}
}
int binary::bsearch(int x)
{
l=1;
h=n;
while(l<=h)
{
mid=(l+h)/2;
if(x<a[mid])
h=mid-1;
else if(x>a[mid])
l=mid+1;
else
{
return mid;
}
}
return 0;
}
void binary::sort()
{
 for(int j=1;j<=n;j++)
```

```
 for(int i=j;i<=n;i++)
 {
 if(a[i]<a[j])
 {
 int temp=a[i];
 a[i]=a[j];
 a[j]=temp;
 }
}
}
void main()
{
 clrscr();
 cout<<"\n*** Program for Binary Search ***\n";
 int x,y,i,yes,no;
 clock_t e,s;
 binary b;
 b.get();
 b.sort();
 cout<<"\n Sorted elements are :"<<endl;
 b.put();
 cout<<"\n Enter elements You want to find :"<<endl;
 cin>>x;
 s=clock();
 y=b.bsearch(x);
 cout<<"find at Position : "<<y<<endl;
 b.put();
 e=clock();
 cout<<"\n Time Complexity is :- "<<((e-s)/CLK_TCK);
 getch();
}

/* Output :-
 *** Program for Binary search ***
 Enter the number of elements :10

 Sorted elements are :
79      211     665     3915    4343    6702    7114
10739   14006   18997
 Enter elements You want to find :
79
find at Position : 1
79      211     665     3915    4343    6702    7114
10739   14006   18997
 Time Complexity is :- 0
 */
```

/* Assignment 17 :- Program for to find maximum and minimum from a given array using maxmin.
Lab:DAA */

```
#include<iostream.h>
#include<conio.h>
#include<stdlib.h>
#include<time.h>
int max,min,n;
class sai
{
        int a[100];
        public:
        int j,i;
        void get();
        void put();
        void maxmin(int,int,int &,int &);
        void show();
};
void sai::get()
{
        cout<<"\n Enter the Size of array :- ";
        cin>>n;
        for(i=1;i<=n;i++)
        {
          a[i]=random(1000);
        }
}
void sai::put()
{
        cout<<"\n show The Element of array :- \n";
        for(i=1;i<=n;i++)
        {
                if(i%8==0)
                cout<<"\n";
                cout<<a[i]<<"\t";
        }
}


void sai::maxmin(int i,int j,int &max,int &min)
{
        int mid,max1,min1,max2,min2;
        if(i==j)
        {
        max=min=a[i];
        }
        else if(i==j-1)
        {
        // cout<<a[i]<<" "<<a[j];
          if(a[i]<a[j])
          {
           max=a[j];
           min=a[i];
          }
```

```
            else
            {
            max=a[i];
            min=a[j];
            }

          }
          else
          {
           mid=(i+j)/2;
           maxmin(i,mid,max1,min1);
           maxmin(mid+1,j,max2,min2);
           max=max1>max2?max1:max2;
           min=min1<min2?min1:min2;
          }
}

void sai::show()
{
        cout<<"\n The maximum element is :- "<<max;
        cout<<"\n The Minimum element is :- "<<min;
}



void main()
{
        clrscr();
        cout<<"\n\t*** Program for to find maximum and minimum from a given array using maxmin ***\n\t";
        sai s;
        clock_t e,l;
        s.get();
        s.put();
        e=clock();
        s.maxmin(1,n,max,min);
        l=clock();
        s.show();
        cout<<"\n The Time Complexity is :- "<<(l-e)/CLK_TCK;
        getch();
}

/* OUTPUT :-

      *** Program for to find maximum and minimum from a given array using max
min ***

 Enter the Size of array :- 5

 show The Element of array :-
10     3      335    33     355
 The maximum element is :- 355
 The Minimum element is :- 3
 The Time Complexity is :- 0        */
```

/* Assignment 18 :- Program for Ascending Merge Sort ... Lab :DAA */

```
#include<iostream.h>
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#include<time.h>
int n;
class Merge
{
int a[1000],i,j;
public:
void read();
void merge_sort(int l,int h);
void merge1(int l,int m,int h);
void disp();
};
void Merge::read()
{
for(i=0;i<n;i++)
{
a[i]=random(20000);
}
}
void Merge::merge_sort(int l,int h)
{
int m1;
if(l<h)
{
m1=int((l+h)/2);
merge_sort(l,m1);
merge_sort(m1+1,h);
merge1(l,m1,h);
}
}

void Merge::merge1(int l,int m,int h)
{
int h1=l,b[1800];
int i=l;
j=m+1;
while((h1<=m)&&(j<=h))
{
if(a[h1]<=a[j])
{
b[i]=a[h1];
i++;
h1++;
}
else
{
b[i]=a[j];
i++;
j++;
```

```
}
}
if(h1<=m)
{
while(h1<=m)
{
b[i]=a[h1];
i++;
h1++;
}
}
else
{
while(j<=h)
{
b[i]=a[j];
i++;
j++;
}
}
for(int k=l;k<=h;k++)
a[k]=b[k];
}
void Merge::disp()
{
for(i=0;i<n;i++)
{
cout<<a[i]<<"\t";
if((i+1)%9==0)
cout<<endl;
}
}
void main()
{
clrscr();
randomize();
clock_t s,e;
int l,h;
Merge m;
cout<<"Program for Ascending Merge Sort ...\n\n";
cout<<"Enter the Element:";
cin>>n;
h=n-1;
l=0;
m.read();
cout<<"\n\nDisplay the ArrayElement=\n\n";
m.disp();
s=clock();
m.merge_sort(l,h);
e=clock();
cout<<"\nAfter Sorting=\n";
m.disp();
cout<<"\nTime Complexity is:= "<<((e-s)/CLK_TCK);
```

```
getch();
}
/* OUTPUT :-
Program for Ascending Merge Sort ...

Enter the Element:20


Display the ArrayElement=

1217    2426    16497   2396    1635    1326    17865   18869   9981
18029   3826    1749    9149    5263    17811   5421    3775    19800
6145    17186
After Sorting=
1217    1326    1635    1749    2396    2426    3775    3826    5263
5421    6145    9149    9981    16497   17186   17811   17865   18029
18869   19800
Time Complexity is:= 0
 */
```

/* Assignment 19 :- Program for Descending Merge Sort ... Lab:DAA */

```
#include<iostream.h>
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#include<time.h>
int n;
class Merge
{
int a[1000],i,j;
public:
void read();
void merge_sort(int l,int h);
void merge1(int l,int m,int h);
void disp();
};
void Merge::read()
{
for(i=0;i<n;i++)
{
a[i]=random(20000);
}
}
void Merge::merge_sort(int l,int h)
{
int m1;
if(l<h)
{
m1=int((l+h)/2);
merge_sort(l,m1);
merge_sort(m1+1,h);
merge1(l,m1,h);
}
}
void Merge::merge1(int l,int m,int h)
{
int h1=l,b[1800];
int i=l;
j=m+1;
while((h1<=m)&&(j<=h))
{
if(a[h1]>=a[j])
{
b[i]=a[h1];
i++;
h1++;
}
else
{
b[i]=a[j];
i++;
j++;
}
```

```
}
if(h1<=m)
{
while(h1<=m)
{
b[i]=a[h1];
i++;
h1++;
}
}
else
{
while(j<=h)
{
b[i]=a[j];
i++;
j++;
}
}
for(int k=l;k<=h;k++)
a[k]=b[k];
}
void Merge::disp()
{
for(i=0;i<n;i++)
{
cout<<a[i]<<"\t";
if((i+1)%9==0)
cout<<endl;
}
}
void main()
{
clrscr();
randomize();
clock_t s,e;
int l,h;
Merge m;
cout<<"Program For Descending Merge Sort ....\n\n";
cout<<"Enter the Element:";
cin>>n;
h=n-1;
l=0;
m.read();
cout<<"\n\nDisplay the ArrayElement=\n\n";
m.disp();
s=clock();
m.merge_sort(l,h);
e=clock();
cout<<"\nAfter Sorting=\n";
m.disp();
cout<<"\nTime Com.= "<<((e-s)/CLK_TCK);
getch();
```

```
}

/* OUTPUT :-
Program For Descending Merge Sort ....

Enter the Element:20


Display the ArrayElement=

10358   1643    9472    3864    102     5807    4162    15154   16666
9315    15129   6801    8252    4668    8776    12038   2719    966
5570    6927
After Sorting=
16666   15154   15129   12038   10358   9472    9315    8776    8252
6927    6801    5807    5570    4668    4162    3864    2719    1643
966     102
Time Com.= 0
*/
```

/* Assignment 20:- Program for Ascending Quick Sort. Lab:DAA */

```
#include<iostream.h>
#include<conio.h>
#include<stdlib.h>
#include<time.h>
int n,a[1000];
class q_sort
{
        public:
        void get();
        void put();
        void quick_sort(int,int);
        int partition(int,int);
};
void q_sort::get()
{
        cout<<"\n Enter the Size Of Array := ";
        cin>>n;
          for(int i=1;i<=n;i++)
          {
            a[i]=random(2000);
          }
}
void q_sort::put()
{
        for(int i=1;i<=n;i++)
        {
          if(i%8==0)
          cout<<"\n";
          cout<<a[i]<<"\t";
        }
}
void q_sort::quick_sort(int p,int q)
{
        int j;
         if(p<q)
         {
           j=partition(p,q);
           quick_sort(p,j-1);
           quick_sort(j+1,q);
         }
}
int q_sort::partition(int m,int p)
{
        int v=a[m];
        int i=m+1;
        int j=p;
        while(i<=j)
        {
          while(a[i]<=v)
          i=i+1;

          while(a[j]>=v)
```

```
            j=j-1;
             if(i<j)
             {
              p=a[i];
              a[i]=a[j];
              a[j]=p;
             }
            }
            a[m]=a[j];
            a[j]=v;
            return j;
}
void main()
{
        clrscr();
        clock_t e,s;
        q_sort q;
        cout<<"\n.... Program For Ascending Quick Sort ......\n\n";
        q.get();
        cout<<"\n Display the element of array before sort :=\n\n";
        q.put();
        s=clock();
        q.quick_sort(1,n);
        e=clock();
        cout<<"\n Display the element of array before sort :=\n";
        q.put();
        cout<<"\n The time Complexity  :="<<(e-s)/CLK_TCK;
        getch();
}


 /*OUTPUT :-

.... Program For Ascending Quick Sort ......


 Enter the Size Of Array := 10

 Display the element of array before sort :=

21     7      670     66     711     434     1073
391     1400    1899
 Display the element of array before sort :=
7      21      66      391     434     670     1073
711     1400    1899
 The time Complexity  :=0
*/
```

/* Assignment 20:- Program for Descending Quick Sort. Lab:DAA */

```
#include<iostream.h>
#include<conio.h>
#include<stdlib.h>
#include<time.h>
int n,a[1000];
class q_sort
{
        public:
        void get();
        void put();
        void quick_sort(int,int);
        int partition(int,int);
};
void q_sort::get()
{
        cout<<"\n Enter the Size Of Array := ";
        cin>>n;
          for(int i=1;i<=n;i++)
          {
            a[i]=random(2000);
          }
}
void q_sort::put()
{
        for(int i=1;i<=n;i++)
        {
          if(i%8==0)
          cout<<"\n";
          cout<<a[i]<<"\t";
        }
}
void q_sort::quick_sort(int p,int q)
{
        int j;
         if(p<q)
         {
           j=partition(p,q);
           quick_sort(p,j-1);
           quick_sort(j+1,q);
         }
}
int q_sort::partition(int m,int p)
{
        int v=a[m];
        int i=m+1;
        int j=p;
        while(i<=j)
        {
          while(a[i]>=v)
          i=i+1;

          while(a[j]<=v)
```

```
            j=j-1;
             if(i<j)
             {
              p=a[i];
              a[i]=a[j];
              a[j]=p;
             }
            }
            a[m]=a[j];
            a[j]=v;
            return j;
}
void main()
{
        clrscr();
        clock_t e,s;
        q_sort q;
        cout<<"\n.... Program For Descending Quick Sort ......\n\n";
        q.get();
        cout<<"\n Display the element of array before sort :=\n\n";
        q.put();
        s=clock();
        q.quick_sort(1,n);
        e=clock();
        cout<<"\n Display the element of array before sort :=\n";
        q.put();
        cout<<"\n The time Complexity  :="<<(e-s)/CLK_TCK;
        getch();
}


/*OUTPUT :-

.... Program For Descending Quick Sort ......


 Enter the Size Of Array := 10

 Display the element of array before sort :=

21     7      670     66     711     434     1073
391     1400    1899
 Display the element of array before sort :=
1899    1400    1073    670     711     434     391
66      21      7
 The time Complexity  :=0
 */
```

/* Assignment 22 :- Program for Strasans matrix multiplication. Lab:DAA */

```
#include<iostream.h>
#include<conio.h>
#include<time.h>
class mmu1
{
 int a[3][3],b[3][3],c[3][3],p,q,r,s,t,u,v,i,j;
 public:
 void get();
 void put();
 void formula();
};
void mmu1::get()
{
 cout<<"\nEnter the matrix 1 :- ";
 for(i=1;i<=2;i++)
 for(j=1;j<=2;j++)
 cin>>a[i][j];

 cout<<"\nEnter the matrix 2 :- ";
 for(i=1;i<=2;i++)
 for(j=1;j<=2;j++)
 cin>>b[i][j];
}
void mmu1::formula()
{
 p=((a[1][1]+a[2][2])*(b[1][1]+b[2][2]));
 q=((a[2][1]+a[2][2])*(b[1][1]));
 r=((a[1][1])*(b[1][2]-b[2][2]));
 s=((a[2][2])*(b[2][1]-b[1][1]));
 t=((a[1][1]+a[1][2])*(b[2][2]));
 u=((a[2][1]-a[1][1])*(b[1][1]+b[1][2]));
 v=((a[1][2]-a[2][2])*(b[2][1]+b[2][2]));
 c[1][1]=p+s - t+v;
 c[1][2]=r+t;
 c[2][1]=q+s;
 c[2][2]=p+r - q+u;
}
void mmu1::put()
{
 for(int i=1;i<=2;i++)
 {
 for(int j=1;j<=2;j++)
 cout<<c[i][j]<<" ";
 cout<<"\n";
}
}
void main()
{
 clrscr();
 cout<<"\n\t*** Program for Stranssen's Matrix Multiplication ***\n\t";
 mmu1 m;
 clock_t e,s;
```

```
 m.get();
 s=clock();
 m.formula();
 cout<<"\nOutput :- "<<endl;
 m.put();
 e=clock();
 cout<<"\n Time complexity :- "<<((e-s)/CLK_TCK);
 getch();
}
/*OUTPUT :-

      *** Program for Stranssen's Matrix Multiplication ***

Enter the matrix 1 :-
1 1
1 1

Enter the matrix 2 :-
2 2
2 2

Output :-
4 4
4 4
 Time complexity :- 0            */
```

/* Assignment 23 :- Program for Knapsack solutions. Lab:DAA */

```
#include<iostream.h>
#include<conio.h>
#include<time.h>
int m,n;
class knapsack
{
 float p[20],w[20],x[20],i,j,sum;
 public:
 void get();
 void order();
 void knap(int,int);
 void show();
};
void knapsack::get()
{
 cout<<"\nEnter the element size & sack size :- \n";
 cin>>n>>m;

 cout<<"\nEnter the profit :- \n";
 for(i=1;i<=n;i++)
 cin>>p[i];

 cout<<"\nEnter the weight :- \n";
 for(i=1;i<=n;i++)
 cin>>w[i];
}
void knapsack::order()
{
 for(i=1;i<=n;i++)
 for(j=1;j<n;j++)
 {
  if((p[j]/w[j])<=(p[j+1]/w[j+1]))
  {
   int temp=p[j];
   p[j]=p[j+1];
   p[j+1]=temp;

   temp=w[j];
   w[j]=w[j+1];
   w[j+1]=temp;
  }
 }
}
void knapsack::knap(int m,int n)
{
 int u;
 sum=0.0;
 for(i=1;i<=n;i++)
 x[i]=0.0;
 u=m;
 for(i=1;i<=n;i++)
 {
```

```
  if(w[i]>u)
  break;
  x[i]=1.0;
  u=u-w[i];
 }
 if(i<=n)
 x[i]=u/w[i];
 for(i=1;i<=n;i++)
 sum=sum+(p[i]*x[i]);
}
void knapsack::show()
{
 for(i=1;i<=n;i++)
 cout<<x[i]<<" ";
 cout<<"\n----------------------\n";
 cout<<"Profit :- "<<sum<<"\n";
 cout<<"\n----------------------\n";
}
void main()
{
 clrscr();
 cout<<"\n\t*** Program for Knapsack Solution ***\n\t";
 knapsack k;
 k.get();
 k.order();
 k.knap(m,n);
 k.show();
 getch();
}
/* OUTPUT :-

       *** Program for Knapsack Solution ***

Enter the element size & sack size :-
7 15

Enter the profit :-
10 5 15 7 6 8 3

Enter the weight :-
2 3 5 7 1 4 1
1 1 1 1 1 0.666667 0
----------------------
Profit :- 45.333332

----------------------
 */
```

/* Assignment 24:-Minmum cost spanning tree using prims algorithm Lab:DAA */

```
#include<iostream.h>
#include<conio.h>
int g[100][100],tree[100],n;
class spanning
{
        public:
         void get()
         {
                cout<<"Enter the number of nodes :";
                cin>>n;
                cout<<"\n Enter the graph \n";
                for(int i=1;i<=n;i++)
                {
                 for(int j=1;j<=n;j++)
                 {
                   cin>>g[i][j];
                 }
         }
}
void dis()
{
        cout<<"\n The Graph Is :\n";
        for(int i=1;i<=n;i++)
        {
                cout<<"\n";
                for(int j=1;j<=n;j++)
                {
                  cout<<g[i][j]<<" ";
                }

        }
}
void prims()
{
        int total=0,v1,v2;
        for(int l=1;l<=n;l++)
                {
                  tree[l]=0;
                }
                tree[1]=1;
                cout<<"\n V1 V2 min_dist : ";
                for(int k=2;k<=n;k++)
                {
                        int min_dist=30000;
                        for(int i=1;i<=n;i++)
                        {
                         for(int j=1;j<=n;j++)
                         {
                          if(g[i][j] && ((tree[i] && !tree[j])||(!tree[i]&&tree[j])))
                          {
                                if(g[i][j]<min_dist)
                                {
```

```
                                        min_dist=g[i][j];
                                        v1=i;
                                        v2=j;
                                    }
                                }
                               }
                              }
                              cout<<"\n"<<v1<<" "<<v2<<" "<<min_dist;
                              tree[v1]=1;
                              tree[v2]=1;
                              total=total+min_dist;
                        }
                        cout<<"\n Cost of spanning tree is : "<<total;
                }
};
void main()
{
        spanning s;
        clrscr();
        cout<<"\n\t. Program For Minimum cost spanning tree Usin Prims Algorithm .\t\n\n";
        s.get();
        s.dis();
        s.prims();
        getch();
}
 /* OUTPUT :-
 Enter the number of nodes :6

 Enter the graph
0 3 0 0 2 0
3 0 2 4 0 0
0 2 0 0 5 2
0 4 0 0 1 0
2 0 5 1 0 6
0 0 2 0 6 0

 The Graph Is :

0 3 0 0 2 0
3 0 2 4 0 0
0 2 0 0 5 2
0 4 0 0 1 0
2 0 5 1 0 6
0 0 2 0 6 0
 V1 V2 min_dist :
1 5 2
4 5 1
1 2 3
2 3 2
3 6 2
 Cost of spanning tree is : 10
*/
```

/* Assignment 25 :- Program for demonstrate Kruskal Algorithm. Lab:DAA */

```
#include<iostream.h>
#include<conio.h>
#define INFINITY 999
typedef struct Graph
{
int v1;
int v2;
int cost;
}GR;
GR G[20];
int tot_edges,tot_nodes;
void create();
void spanning_tree();
int Minimum(int);
void main()
{
clrscr();
cout<<"\n\t Graph Creation by adjacency matrix ";
create();
spanning_tree();
getch();
}
void create()
{
int k;
cout<<"\n Enter Total number of nodes: ";
cin>>tot_nodes;
cout<<"\n Enter Total number of edges: ";
cin>>tot_edges;
for(k=1;k<=tot_edges;k++)
{
cout<<"\n Enter Edge in (V1 V2)form ";
cin>>G[k].v1>>G[k].v2;
cout<<"\n Enter Corresponding Cost ";
cin>>G[k].cost;
}
}
void spanning_tree()
{
int count,k,v1,v2,i,j,tree[10][10],pos,parent[10];
int sum;
int Find(int v2,int parent[]);
void Union(int i,int j,int parent[]);
count=0;
k=1;
sum=0;
for(i=1;i<=tot_nodes;i++)
parent[i]=i;
while(count!=tot_nodes-1)
{
pos=Minimum(tot_edges);//finding the minimum cost edge
if(pos==-1)//Perhaps no node in the graph
```

```
break;
v1=G[pos].v1;
v2=G[pos].v2;
i=Find(v1,parent);
j=Find(v2,parent);
if(i!=j)
{
tree[k][1]=v1;//storing the minimum edge in array tree[]
tree[k][2]=v2;
k++;
count++;
sum+=G[pos].cost;//accumulating the total cost of MST
Union(i,j,parent);
}
G[pos].cost=INFINITY;
}
if(count==tot_nodes-1)
{
cout<<"\n Spanning tree is...";
cout<<"\n-------------------------\n";
for(i=1;i<=tot_nodes-1;i++)
{
cout<<tree[i][1];
cout<<" - ";
cout<<tree[i][2];
cout<<"]";
}
cout<<"\n--------------------------";
cout<<"\nCost of Spanning Tree is = "<<sum;
}
else
{
cout<<"There is no Spanning Tree";
}
}
int Minimum(int n)
{
int i,small,pos;
small=INFINITY;
pos=-1;
for(i=1;i<=n;i++)
{
if(G[i].cost<small)
{
small=G[i].cost;
pos=i;
}
}
return pos;
}
int Find(int v2,int parent[])
{
while(parent[v2]!=v2)
```

```
{
v2=parent[v2];
}
return v2;
}
void Union(int i,int j,int parent[])
{
if(i<j)
parent[j]=i;
else
parent[i]=j;
}


/* Output :-
 Enter Edge in (V1 V2)form 1 4

 Enter Corresponding Cost 4
 Enter Edge in (V1 V2)form 1 3
 Enter Corresponding Cost 2
 Enter Edge in (V1 V2)form 2 4
 Enter Corresponding Cost 3
 Enter Edge in (V1 V2)form 2 3
 Enter Corresponding Cost 1
 Enter Edge in (V1 V2)form 3 4
 Enter Corresponding Cost 4




 Spanning tree is...
--------------------------
2 - 3]1 - 3]2 - 4]
--------------------------
        Cost of Spanning Tree is = 6

*/
```

/* Assignment 26 :- Program for single source shortest path. Lab:DAA */

```
#include<iostream.h>
#include<conio.h>
int n;
class single
{
int v,cost[10][10],i,j,s[10];
float dist[10];
public:
void get();
void sisource();
void display();
};
void single::get()
{
cout<<"\nEnter the no. of vertices :- \n";
cin>>n;
cout<<"\nEnter the adjency matrix :- \n";
for(i=1;i<=n;i++)
for(j=1;j<=n;j++)
{
cin>>cost [i][j];
if(cost [i][j]==-1)
cost [i][j]=9999;
}
}
void single::sisource()
{
v=1;
for(i=1;i<=n;i++)
{
s[i]=0;
dist[i] = cost [v][i];
}
s[v]=1;
dist[v]=0.0;
int minu,u;
for(int num=2;num<=n;num++)
{
for(i=1;i<=n;i++)
if(s[i]==0)
minu=dist[i];
for(i=1;i<=n;i++)
{
if(s[i]==0 && dist[i]<minu)
{
minu=dist[i];
u=i;
}
}
s[u]=1;
for(i=1;i<=n;i++)
{
```

```
if(cost [u][i]>0 && cost [u][i] < 9999 && s[i]==0) {
if(dist[i] > (dist [u] + cost[u][i]))
{
dist [i]= dist [u] + cost [u][i];
}
}
}
}
}
void single::display()
{
cout<<endl;
for(i=1;i<=n;i++)
{
cout<<"distance from 1---------->"<<i<<"\t";
cout<<dist[i]<<" ";
cout<<endl;
}
}
void main()
{
clrscr();
single g;
g.get();
g.sisource();
g.display();
getch();
}

/* Output :-

Enter the no. of vertices :- 6

Enter the adjency matrix :-
0 50 45 10 -1 -1
-1 0 10 15 -1 -1
-1 -1 0 -1 30 -1
20 -1 -1 0 15 -1
-1 20 35 -1 0 -1
-1 -1 -1 -1 3 0

distance from 1---------->1   0
distance from 1---------->2   45
distance from 1---------->3   45
distance from 1---------->4   10
distance from 1---------->5   25
distance from 1---------->6   9999

*/
```

/* Assignment 27 :- Implementation of 0/1 knapsack. Lab:DAA */

```
#include<iostream.h>
#include<conio.h>
#include<stdlib.h>
int table[10][10];
int w[100],v[100];
int W,n;
class knap
{
public:
knap()
{
cout<<"\n\t\t 0/1 Knapsack Problem using Dynamic Programming"; /*initialization of table*/
for(int i=0;i<=n;i++)
{
for(int j=0;j<=W;j++)
{
table[i][j]=0;
}
}
}
void get()
{
cout<<"\n Enter how many products you want to declare :- \n"; cin>>n;
cout<<"\n Enter Knapsack size :\n";
cin>>W;
cout<<"\n Enter "<<n<<" products weight and profit :- \n"; for(int i=1;i<=n;i++)
{ cout<<"\n Enter Product "<<i<<" weight :- \t";
cin>>w[i];
cout<<"\n Enter Product "<<i<<" profit :- \t";
cin>>v[i];
}
}
int max(int a,int b)
{
if(a>b)
return a;
else
return b;
}
void find_item(int i,int k,int w[5])
{
cout<<"\nFor the Knapsack...\n";
while(i>0 && k>0)
{
if(table[i][k]!=table[i-1][k])
{
cout<<"\nItem "<<i<<" is selected\n";
k=k-w[i];
i=i-1;
}
else
i=i-1;
```

```
}
}
void DKP(int n,int W,int w[5],int v[5])
{
int i,j;
int val1,val2;
for(i=0;i<=n;i++)
{
for(j=0;j<=W;j++)
{
table[i][0]=0;
table[0][j]=0;
}
}
for(i=1;i<=n;i++)
{
for(j=1;j<=W;j++)
{
if(w[i]<=j)
{
val1=table[i-1][j];
val2=v[i]+table[i-1][j-w[i]];
table[i][j]=max(val1,val2);
}
else
table[i][j]=table[i-1][j];
}
}
cout<<"\n Table constructed using dynamic programming is...\n";
for(i=0;i<=n;i++)
{
for(j=0;j<=W;j++)
cout<<table[i][j]<<"\t";
cout<<"\n";
}
find_item(n,W,w);
}
};
void main()
{
knap k;
clrscr();
k.get();
k.DKP(n,W,w,v);
getch();
}

/*  Output :-
 Enter how many products you want to declare :- 4

 Enter knapsack size :- 5

 Enter 4 product weight and profit :-
```

Enter product 1 weight :- 2

Enter product 1 weight :- 3

Enter Product 2 weight :- 3

Enter Product 2 profit :- 4

Enter Product 3 weight :- 4

Enter Product 3 profit :- 5

Enter Product 4 weight :- 5

Enter Product 4 profit :- 6
Table constructed using dynamic programming is...

0	0	0	0	0	0
0	0	3	3	3	3
0	0	3	4	4	7
0	0	3	4	5	7
0	0	3	4	5	7

For the Knapsack...
Item 2 is selected
Item 1 is selected

*/

/* Assignment 28 :- Program for longest common subsequence. Lab:DAA */

```
#include<iostream.h>
#include<conio.h>
#include<string.h>
#include<stdio.h>
void print_lcs(char b[][20],char x[],int i,int j)
{
if(i==0 || j==0)
return;
if(b[i][j]=='c')
{
print_lcs(b,x,i-1,j-1);
cout<<x[i-1]<<"\t";
}
else
if(b[i][j]=='u')
print_lcs(b,x,i-1,j);
else
print_lcs(b,x,i,j-1);
}
void lcs_length(char x[],char y[])
{
int m,n,i,j,c[20][20];
char b[20][20];
m=strlen(x);
n=strlen(y);
for(i=0;i<=m;i++)
c[i][0]=0;
for(i=0;i<=n;i++)
c[0][i]=0;
for(i=1;i<=m;i++)
for(j=1;j<=n;j++)
{
if(x[i-1]==y[j-1])
{
c[i][j]=c[i-1][j-1]+1;
b[i][j]='c'; //c stands for left up cross
}
else
if(c[i-1][j]>=c[i][j-1])
{
c[i][j]=c[i-1][j];
b[i][j]='u'; //u stands for upright or above
}
else
{
c[i][j]=c[i][j-1];
b[i][j]='l'; //l stands for left
}
}
print_lcs(b,x,m,n);
}
void lcs()
```

```
{
int i,j;
char x[20],y[20];
cout<<"1st sequence :- ";
gets(x);
cout<<"2nd sequence :- ";
gets(y);
cout<<"\nLCS are :- ";
lcs_length(x,y);
cout<< "\n";
// lcs_length(y,x);
}
void main()
{
clrscr();
char ch;
do
{
lcs();
cout<<"\nContinue(y/n) :- ";
cin>>ch;
}
while(ch=='y'||ch=='Y');
getch();
}

/*Output :-
1st sequence :- BAAADC
2nd sequence :- ABAADC

LCS are :- B    A       A       D       C

Continue(y/n) :- n

*/
```

/* Assignment 29 :- Implentation of matrix chain multiplication. Lab:DAA */

```
#include<iostream.h>
#include<conio.h>
#define INF 30000
int p[100],m[100][100],s[100][100],k,n,j,q,i,l;
class matrix_chain
{
public:
void get()
{
cout<<"\n Enter total matrix count :\n";
cin>>n;
cout<<"\n Enter Matrix Orders :\n";
for(int i=0;i<=n;i++)
{
cout<<"\nP["<<i<<"]=";
cin>>p[i];
}
matrix_chain1(p,n);
}
void matrix_chain1(int p[],int n)
{
for (i=1;i<=n;i++)
m[i][i]=0;
for (l=2;l<=n;l++)
{
for (i=1;i<=n-l+1;i++)
{
j=i+l-1;
m[i][j]=INF;
for (k=i;k<=j-1;k++)
{
q=m[i][k]+m[k+1][j]+p[i-1]*p[k]*p[j];
if(q<m[i][j])
{
m[i][j]=q;
s[i][j]=k;
}
}
}
}
cout<<"\n Total Optimal Scalar Multiplication needed are :- "<<m[1][n];
cout<<"\n Array M[i][j] is \n";
for(i=1;i<=n;i++)
{
cout<<"\n";
for(j=1;j<=n;j++)
cout<<m[i][j]<<"\t";
}
cout<<"\n Array S[i][j] is :- \n";
for(i=1;i<=n;i++)
{
cout<<"\n";
```

```
for(j=1;j<=n;j++)
cout<<s[i][j]<<"\t";
}
}
void print_optimal(int i,int j)
{
if (i==j)
cout<<" A "<<i;
else
{
cout<<" ( ";
print_optimal(i,s[i][j]);
print_optimal(s[i][j]+1,j);
cout<<" ) ";
}
}
};
void main()
{
clrscr();
matrix_chain m1;
m1.get();
m1.print_optimal(1,n);
getch();
}

/* Output :-

 Enter total matrix count :- 3
 Enter Matrix Orders :
P[0]=10

P[1]=100

P[2]=5

P[3]=50
 Total Optimal Scalar Multiplication needed are :- 7500
 Array M[i][j] is

0       5000    7500
0       0       25000
0       0       0

 Array S[i][j] is :-

0       1       2
0       0       2
0       0       0
optimal sequence is :-
( ( A 1 A 2 ) A 3 )

*/
```

/* Assignment 30 :- Program for All pair shortest path. Lab:DAA */

```
#include<iostream.h>
#include<conio.h>
class all
{
int s[10][10],a[10][10],i,j,k,n,m;
public:
void get();
int min(int,int);
void find();
void display();
};
int all::min(int m,int n)
{
return(m<n ?m:n);
}
void all::get()
{
cout<<"\n Enter the size of element :- \n";
cin>>n;
cout<<"\nEnter the element in array :- \n";
for(i=1;i<=n;i++)
{
for(j=1;j<=n;j++)
{
cin>>a[i][j];
if(a[i][j]==-1)
{
s[i][j]=9999;
}
else
{
s[i][j]=a[i][j];
}
}
}
}
void all::find()
{
for(i=1;i<=n;i++)
for(j=1;j<=n;j++)
for(k=1;k<=n;k++)
{
if(i==j)
{
s[i][j]=0;
}
else
s[i][j]=min(s[i][j],s[i][k]+s[k][j]);
if(s[i][j]>=9999)
s[i][j]=0;
}
```

```
}
void all::display()
{
cout<<"\n display the element after perform operation find :- \n";
for(i=1;i<=n;i++)
{
for(j=1;j<=n;j++)
{
cout<<s[i][j]<<"\t";
}
cout<<endl;
}
}
void main()
{
clrscr();
all a;
a.get();
a.find();
a.display();
getch();
}

/*Output :-

Enter the size of element :- 3

Enter the element in array :-
0 4 11
6 0 2
3 -1 0

 display the element after perform operation find :-
0       4       6
5       0       2
3       7       0

*/
```

/* Assignment 31 :- Program to demonstrate breath first traversal. Lab:DAA */

```
#include<iostream.h>
#include<conio.h>
class bfstree
{
int reach[20],a[20][20],q[20],n,i,j,f,r,index;
public:
bfstree()
{
f=r=0;
index=1;
}
void get();
void bfs(int);
};
void bfstree::get()
{
cout<<"\nEnter number of vertices :- ";
cin>>n;
cout<<"\nEnter Adjacency matrix :- ";
for(i=1;i<=n;i++)
for(j=1;j<=n;j++)
{
reach[i]=0;
cin>>a[i][j];
}
for(i=1;i<=n;i++)
if(reach[i]==0)
bfs(i);
}
void bfstree::bfs(int index)
{ f=r=0;
reach[index]=1;
f++;
r++;
q[r]=index;
while(f<=r)
{
index=q[f];
f++;
cout<<index<<"\t";
for(j=1;j<=n;j++)
{
if(a[index][j]==1 && reach[j]!=1)
{
reach[j]=1;
r++;
q[r]=j;
}
}
}
}
void main()
```

```
{
clrscr();
bfstree b;
b.get();
getch();
}

/* Output :-

Enter number of vertices :- 6

Enter Adjacency matrix :-
0 1 1 0 0 0
1 0 0 1 0 0
1 0 0 0 0 1
0 1 0 0 1 1
0 0 0 1 0 0
0 0 1 1 0 0

1      2      3      4      6      5

*/
```

/* Assignment 32 :- Program to demonstrate depth fisrt traversal. Lab:DAA */

```
#include<iostream.h>
#include<conio.h>
class dfstree
{
int a[20][20], visited[20],n,i,j;
public:
void dfs(int);
void get();
};
void dfstree::get()
{
cout<<"\nEnter the number of node :- ";
cin>>n;
for(i=1;i<=n;i++)
visited[i]=0;
cout<<"\nEnter the adjancy matrix :- ";
for(i=1;i<=n;i++)
{ for(j=1;j<=n;j++)
cin>>a[i][j];
}
for(i=1;i<=n;i++)
if(visited[i]==0)
dfs(i);
}
void dfstree::dfs(int v)
{
int k;
visited[v]=1;
cout<<"\t"<<v;
for(k=1;k<=n;k++)
if(a[v][k]==1)
if(visited[k]==0)
dfs(k);
}
void main()
{ clrscr();
dfstree d;
d.get();
getch();
}
/*Output :-
Enter the number of node :- 5
Enter the adjancy matrix :-
0 1 1 0 0
1 0 0 1 1
1 0 0 1 0
0 1 1 0 1
0 1 0 1 0

        1       2       4       3       5
*/
```

/* Assignment 33 :- Program for demonstrate Topological sort. Lab:DAA */

```
#include<iostream.h>
#include<conio.h>
class top
{
public :
int
cost[10][10],n1,n,indeg[10],q[10],visit[10],i,j;
int f,r,count;
top()
{
f=r=0;
}
void get ()
{
cout<<"\nEnter number of vertices :- ";
cin>>n;
cout<<"\nEnter matrix :- \n";
for(i=1;i<=n;i++)
for(j=1;j<=n;j++)
cin>>cost[i][j];
for(i=1;i<=n;i++)
{
indeg[i]=0;
visit[i]=0;
}
for(i=1;i<=n;i++)
for(j=1;j<=n;j++)
{ if(cost[i][j]==1)
indeg[j]=indeg[j]+1;
}
cout<<"\n Indegree :- \n";
cout<<"\n";
for(int k=1;k<=n;k++)
{
cout<<"Indgeree of NODE"<<k<<"Is"<<indeg[k]<<"\t"<<"\n";
}
}
void topo()
{
for(i=1;i<=n;i++)
{
if(indeg[i]==0 && visit[i]!=1)
{
if(f==0 && r==0)
{
f++;
r++;
}
else
r++;
q[r]=i;
```

```
visit[i]=1;
}
}
while(f<=r)
{
n1=q[f];
f++;
cout<<" "<<n1;
for(j=1;j<=n;j++)
{ if(cost[n1][j]==1 && visit[j]!=1) {
indeg[j]=indeg[j]-1;
if(indeg[j]==0)
{
r++;
q[r]=j;
visit[j]=1;
}
}
}
}
}
};
void main()
{
clrscr();
top p;
p.get();
p.topo();
getch();
}
/* Output :-

Enter number of vertices :- 7

Enter matrix
0 1 1 0 0 0 0
0 0 0 0 1 0 1
0 0 0 0 0 1 0
1 0 0 0 0 1 0
0 0 0 0 0 0 0
0 0 0 0 0 0 0
0 0 0 0 1 1 0
Indegree :
Indgeree of NODE 1 Is :- 1
Indgeree of NODE 2 Is :- 1
Indgeree of NODE 3 Is :- 1
Indgeree of NODE 4 Is :- 0
Indgeree of NODE 5 Is :- 2
Indgeree of NODE 6 Is :- 3
Indgeree of NODE 7 Is :- 1

 4 1 2 3 7 5 6
*/
```

```
/* Assignment 34 :- program for n Queen for all solution
   Lab:DAA
*/
#include<iostream.h>
#include<conio.h>
#include<math.h>
int x[100],n;
class nqueen
{
int z;
public:
void get();
void show();
void queen(int,int);
int place(int,int);
};
void nqueen::get()
{
cout<<"Enter the no of queens :- \n";
cin>>n;
for(int i=1;i<=n;i++)
x[i]=0;
z=0;
queen(1,n);
}
void nqueen::queen(int k,int n)
{
for(int i=1;i<=n;i++)
{
if(place(k,i))
{
x[k]=i;
if(k==n)
{
cout<<endl;z++; cout<<z<<":->";
for( i=1;i<=n;i++)
cout<<x[i]<<"\t";
}
else
queen(k+1,n);
}
}
}
int nqueen::place(int k,int i)
{
for(int j=1;j<=k-1;j++)
{
if((x[j]==i) || abs(x[j]-i)==(abs(j-k)))
return 0;
}
return 1;
}
void main()
```

```
{
clrscr();
nqueen n;
n.get();
getch();
}
/* OutPut  :-
Enter the no of queens :-
4

1:->2   4       1       3
2:->3   1       4       2

*/
```

/* Assignment 35 :- program for n Queen for in equivalent solution Lab:DAA */

```
#include<iostream.h>
#include<conio.h>
#include<math.h>
class nqueen
{
int n,x[200],cnt;
public :
nqueen(int);
void putdata();
int place(int);
void NQueen();
};
nqueen :: nqueen(int no)
{
n = no;
cnt = 0;
for(int i = 1;i <= n;i++)
x[i] = 0;
}
void nqueen :: putdata()
{
cout<<"\n";
for(int j = 1;j <= n;j++)
{
cout<<x[j]<<"\t";
}
}

void nqueen :: NQueen()
{
int k = 1;
x[k] = 0;
while(k > 0)
{
x[k] = x[k] + 1;
if( k == 1 && x[k] > (n/2) )
{
break;
}
while( x[k] <= n && place(k) == 0)
{
x[k] = x[k] + 1;
}
if(x[k] <= n)
{
if( k == n)
{
cnt++;
cout<<"\nSolution Number "<<cnt<<" : \n";
putdata();
}
else
```

```
{
k++;
x[k] = 0;
}
}
else
{
k--;
}
}
}
int nqueen :: place(int k)
{
for(int j = 1;j < k;j++)
{
if( x[j] == x[k] || abs(x[j] - x[k]) == abs(j - k) )
return(0);
}
return(1);
}
void main()
{
clrscr();
int no;
cout<<"\nEnter number of queen : ";
cin>>no;
if( no == 2 || no == 3)
{
cout<<"\nSolution is not possible.";
}
else
{
nqueen n(no);
n.NQueen();
}
getch();
}
/*  OUTPUT :-

Enter number of queen : 4

Solution Number 1 :

2       4       1       3
*/
```

/* Assignment 36 :- program for graph coloring Lab:DAA */

```
#include<iostream.h>
#include<conio.h>
#include<time.h>
int c[10][10],n,m;
class graph
{
int i,j,x[100];
public:
void get();
void color(int);
void show();
void nextvalue(int);
};
void graph::get()
{
cout<<"Enter the size of array\n";
cin>>n;
cout<<"Enter the color for graph\n";
cin>>m;
cout<<"Enter the adjancy matrix\n";
for(i=1;i<=n;i++)
{
for(j=1;j<=n;j++)
{
cin>>c[i][j];
}
}
for(i=1;i<=n;i++)
x[i]=0;
color(1);
}
void graph::nextvalue(int k)
{
do
{
x[k]=((x[k]+1)%(m+1));
if(x[k]==0)
return ;
for(j=1;j<=n;j++)
if((c[k][j]!=0)&&(x[k]==x[j]))
break;
if(j==n+1)
return;
}while(1);
}
void graph::color(int k)
{
do
{
nextvalue(k);
if(x[k]==0)
return;
```

```
if(k==n)
{
cout<<"\nColour of graph is :";
for(i=1;i<=n;i++)
cout<<x[i]<<"\t";
}
else
color(k+1);
}while(1);
}
void main()
{
clrscr();
graph g;
g.get();
getch();
}



/*OUTPUT :-
Enter the size of array
3
Enter the color for graph
2
Enter the adjancy matrix
0 1 0
1 0 1
0 1 0

Colour of graph is :1   2       1
Colour of graph is :2   1       2

*/
```

/*Assignment 37 :- Program for code 1 using postfix expression Lab:DAA */

```
#include<iostream.h>
#include<conio.h>
class node
{
public:
node *left,*right;
char data[30];
};
class code
{
private:
char expr[30];
node *n,*root;
int f;
public:
void get();
int Isoperand(char);
node *create_tree();
void print(char *);
void mycode(node *,int);
};
void code::get()
{
cout<<"\nEnter The Postfix Expresion:";
cin>>expr;
root = create_tree();
mycode(root,0);
}
int code::Isoperand(char c)
{
if((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z'))
return 1;
else
return 0;
}
void code::print(char *t)
{
switch(t[0])
{
case '+':cout<<"ADD ";break;
case '-':cout<<"SUB ";break;
case '*':cout<<"MPY ";break;
case '/':cout<<"DIV ";break;
default:cout<<t;
}
}
node* code::create_tree()
{
int i=0;
node *stack[10];
int top = -1;
while(expr[i] != '\0')
```

```
{
n = new node;
n->data[0] = expr[i];
n->data[1] = '\0';
n->left = NULL;
n->right = NULL;
if(Isoperand(expr[i]))
stack[++top] = n;
else
{
n->right = stack[top--];
n->left = stack[top--];
stack[++top] = n;
}
i++;
}
return stack[top];
}
void code::mycode(node *t,int i)
{
if(t->left == NULL && t->right == NULL)
{
cout<<"\nLOAD "<<t->data;
return;
}
f = 0;
if(t->right->left != NULL && t->right->right != NULL)
{
mycode(t->right,i);
i++;
cout<<"\nSTORE T"<<i;
t->right->data[0] = 'T';
t->right->data[1] = '1';
t->right->data[2] = '\0';
f = 1;
}
mycode(t->left,i);
if(f == 1)
{
cout<<"\n";
print(t->data);
cout<<"T"<<i;
i--;
}
else
{
cout<<"\n";
print(t->data);
cout<<" ";
print(t->right->data);
}
}
void main()
```

```
{
clrscr();
char ch;
do
{
code obj;
obj.get();
cout<<"\nAre You Want To Continue(Y/N):";
cin>>ch;
}while(ch == 'Y' || ch == 'y');
getch();
}
/* OUTPUT :-

Enter The Postfix Expresion:ab+

LOAD a
ADD  b
Are You Want To Continue(Y/N):n

*/
```

```
/* Assignment 38 :- Program for Code 2     Lab:DAA  */
#include<iostream.h>
#include<conio.h>
#include<stdio.h>
#include<ctype.h>
class vcode2
{
private:
int i,n,cnt,itop,istack[50],icnt;
char prefix[50],top,ch[100];
struct tree
{
char data;
int mr;
tree *left,*right,*parent;
};
public:
struct tree *ltemp,*rtemp,*temp,*root,*current,*stack[20];
vcode1()
{
top=-1;
itop=-1;
cnt=0;
cnt=1;
}
void spush(tree*);
void spop();
int ipop();
void ipush(int);
char *data(tree*);
void findmr(tree *);
void read();
void cal();
void print();
void inorder(tree*);
void preorder(tree*);
void code2(tree*,int);
};
void vcode2::read()
{
cout<<"\nENTER THE PREFIX EXPRESSION :";
cin>>prefix;
cout<<"\nENTER THE NUMBER OF REGISTERS :";
cin>>n;
}
void vcode2::spush(tree *ele)
{
stack[++top]=ele;
}
void vcode2::spop()
{
istack[top--];
}
```

```
void vcode2::ipush(int c)
{
istack[++top]=c;
}
int vcode2::ipop()
{
return(istack[itop--]);
}
void vcode2::cal()
{
root=NULL;
for(i=0;prefix[i]!='\0';i++)
{
if(root==NULL)
{
root=new tree;
root->data=prefix[i];
root->left=root->right=NULL;
spush(root);
if(isalpha(prefix[i+1]) && isalpha(prefix[i+2]))
{
ltemp=new tree;
ltemp->data=prefix[i+1];
ltemp->left=ltemp->right=NULL;
rtemp=new tree;
rtemp->data=prefix[i+2];
rtemp->left=rtemp->right=NULL;
root->left=ltemp;
root->right=rtemp;
spop();
i++;i++;
}
}
else
if(!(isalpha(prefix[i])))
{
temp= new tree;
temp->data=prefix[i];
temp->left=temp->right=NULL;
current=stack[top];
if(current->left!=NULL)
{
current->right=temp;
spop();
}
else
current->left=temp;
spush(temp);
if(isalpha(prefix[i+1]) && isalpha(prefix[i+2]))
{
ltemp=new tree;
ltemp->data=prefix[i+1];
ltemp->left=ltemp->right=NULL;
```

```
rtemp=new tree;
rtemp->data=prefix[i+2];
rtemp->left=rtemp->right=NULL;
current=stack[top];
current->left=ltemp;
current->right=rtemp;
spop();
i++;i++;
}
}
else
if(isalpha(prefix[i]))
{
temp= new tree;
temp->data=prefix[i];
temp->left=temp->right=NULL;
current=stack[top];
if(current->left!=NULL)
{
current->right=temp;
spop();
}
else
current->left=temp;
}
}
}
void vcode2::preorder(tree *r)
{
if(r!=NULL)
{
r->left->parent=r;
preorder(r->left);
findmr(r);
cout<<" "<<r->data;
r->right->parent=r;
preorder(r->right);
findmr(r);
}
}
void vcode2::inorder(tree *r)
{
if(r!=NULL)
{
inorder(r->left);
cout<<" "<<r->mr;
inorder(r->right);
}
}
void vcode2::findmr(tree *p)
{
int l1,l2;
if(p->left==NULL && p->right==NULL && p->parent->right==p)
```

```
p->mr=0;
else
if(p->left==NULL && p->right==NULL && p->parent->left==p)
p->mr=1;
else
if((l1=p->left->mr)!=(l2=p->right->mr))
p->mr=((l1>l2)?l1:l2);
else
if((l1=p->left->mr)==(l2=p->right->mr))
p->mr=l1+1;
}
void vcode2::print()
{
root->parent->data='=';
cout<<"INFIX :";
preorder(root);
cout<<endl;
cout<<"MR VALUES :";
inorder(root);
cout<<endl<<endl;
code2(root,1);
}
void vcode2::code2(tree *t,int icnt)
{
tree *lc,*rc;
if((t->left==NULL) && (t->right==NULL)&& t->parent->left==t)
{
cout<<"LOAD "<<t->data<<" R "<<icnt<<endl;
return;
}
lc=t->left;
rc=t->right;
if(rc->mr==0)
{
code2(lc,icnt);
cout<<data(t)<<" R "<<icnt<<","<<rc->data<<",R"<<icnt<<endl;
}
else
if(lc->mr >=n && rc->mr >=n)
{
code2(rc,icnt);
ipush(++cnt);
cout<<"STORE R"<<icnt<<",T"<<cnt<<endl;
code2(lc,icnt);
cout<<data(t)<<" R"<<icnt<<",T"<<cnt<<" ,R"<<icnt<<endl;
cnt=ipop();
}
else
if(lc->mr< rc->mr)
{
code2(rc,icnt);
code2(lc,icnt+1);
cout<<data(t)<<" R"<<icnt+1<<" ,R"<<icnt<<" ,R"<<icnt<<endl;
```

```
}
else
//if(lc->mr >=rc->mr &&rc->mr <n)
{
code2(lc,icnt);
code2(rc,icnt+1);
cout<<data(t)<<" R"<<icnt<<" ,R"<<icnt+1<<" ,R"<<icnt<<endl;
}
}
char* vcode2::data(tree *t1)
{
switch(t1->data)
{
case '+': return ("ADD");
case '-': return ("SUB");
case '*': return ("MPY");
case '/': return ("DIV");
}
return 0;
}
main()
{
vcode2 c;
clrscr();
c.read();
c.cal();
c.print();
getch();
return 0;
}
/* OUTPUT :-

ENTER THE PREFIX EXPRESSION :+ab

ENTER THE NUMBER OF REGISTERS :1
INFIX : a + b
MR VALUES : 1 1 0

LOAD a R 1
ADD R 1,b,R1
Null pointer assignment
*/
```