```
     -------------------------------------------------------------
-----Assignment Name: program for gcd & lcm
Class: MCA –II (Division B)                  Lab: CA Lab V (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();
}
```

```
Enter Two Numbers For Finding GCD :
6
4
GCD = 2
LCM = 12
```

```
-------------------------------------------------------------------
Assignment Name: Program for union & find
Class: MCA -II (Division B)                  Lab: CA Lab V (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<<"1:read_ele 2:simpleunion 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 ele";
  cin>>n;
  cout<<"\n ent element:";
  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;
    }
    return 0;
    }
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 :=
1:read_ele 2:simpleunion 3:find 4:print 5:exit

 enter the choice1

 enter the number  of ele6

 ent element:-1 1 1 -1 4 4

 enter the choice2

 enter root of two set node 1 4

 enter the choice4

 union of two set4 1 1 -1 4 4
 enter the choice3

 find the node6
root node is:4
 enter the choice5  */
```

```
--------------------------------------------------------------------
Assignment Name: Program for Max Heap using Insert
Class: MCA -II (Division B)                  Lab: CA Lab V (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 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 BEFORE INSERT:==>";
  h.disp();
  s=clock();
  h.insert();
  e=clock();
  cout<<"\n AFTER INSERT:==>\n";
  h.disp();
  cout<<"\n THE TIME COMPLEXITY IS:==>"<<((e-s) / CLK_TCK);
  getch();
}

//Output
 ENTER THE SIZE:==>10

 BEFORE 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 Name: Program for Min Heap using Insert
Class: MCA -II (Division B)                  Lab: CA Lab V (DAA)
-------------------------------------------------------------------
#include<iostream.h>
#include<conio.h>
#include<stdlib.h>
#include<time.h>
class heap
{
  private:
        int a[2000],n;
  public:
       void getdata();
       void insert();
       void disp();
};
void heap::getdata()
{
  cout<<"\n ENTER THE SIZE:==>";
  cin>>n;
  for(int i=1;i<=n;i++)
  {
    a[i]=random(1000);
  }
}
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%9==0)
   cout<<"\n";
     cout<<a[i]<<"\t";
 }
}
void main()
{
  clrscr();
  clock_t e,s;
  heap h;
  h.getdata();
  cout<<"\n BEFORE INSERT:==>";
  h.disp();
  s=clock();
  h.insert();
  e=clock();
  cout<<"\n AFTER INSERT:==>\n";
  h.disp();
  cout<<"\n THE TIME COMPLEXITY IS:==>"<<((e-s) / CLK_TCK);
  getch();
}

//Output

 ENTER THE SIZE:==>10

 BEFORE INSERT:==>10    3       335     33      355     217     536
195

700     949
 AFTER INSERT:==>
3       10      217     33      355     335     536     195
700     949
 THE TIME COMPLEXITY IS:==>0
```

```
--------------------------------------------------------------------
Assignment Name: Prog. For HeapSort Ascending using Insert / Delete
Class: MCA -II (Division B)                  Lab: CA Lab V (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<<"Enter 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;
return;
}
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 emtpy";
}
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 Befor Sort"<<endl;
s=clock();
 h.heapsort();
e=clock();
 cout<<"\n\n After Sort"<<endl;
 h.disp1();
 cout<<"\n\n Time Complexity"<<((e-s)/CLK_TCK);
 getch();
}

/*
  OutPut
Enter Size: 50


 Befor 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 Complexity0
*/
```

```
------------------------------------------------------------------
Assignment Name: Prog. For HeapSort descending using Insert/Delete
Class: MCA -II (Division B)                   Lab: CA Lab V (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<<"Enter 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;
return;
}
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 emtpy";
}
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 Befor Sort"<<endl;
s=clock();
 h.heapsort();
e=clock();
 cout<<"\n\n After Sort"<<endl;
 h.disp1();
 cout<<"\n\n Time Complexity"<<((e-s)/CLK_TCK);
 getch();
}

/*
  OutPut

 Enter Size: 50

 Befor 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 Complexity0
 */
```

```
-------------------------------------------------------------------
Assignment Name: Program for max heap using Heapify/Adjust
Class: MCA -II (Division B)                 Lab: CA Lab V (DAA)
-------------------------------------------------------------------
#include<iostream.h>
#include<conio.h>
#include<time.h>
#include<stdlib.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<<"enter the size of array";
 cin>>n;
 for(i=1;i<=n;i++)
 a[i]=random(1000);
}
void heap::show()
{
  cout<<"\nthe 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<<"\nthe TimeComplexity is=>"<<(e-s)/CLK_TCK;
 getch();
}
/*
output==>
enter the size of array
7
element are=>
10      3       335     33      355     217     536
ele after max heap=>
536     355     335     33      3       217     10

Time Complexity is=>0
*/
```

```
-------------------------------------------------------------------
Assignment Name: Program for Min Heap using Heapify / Adjust
Class: MCA -II (Division B)                  Lab: CA Lab V (DAA)
-------------------------------------------------------------------
#include<iostream.h>
#include<conio.h>
#include<time.h>
#include<stdlib.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<<"enter the size of array";
 cin>>n;
 for(i=1;i<=n;i++)
 a[i]=random(1000);
}
void heap::show()
{
  cout<<"\nthe 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<<"\nthe TimeComplexity is=>"<<(e-s)/CLK_TCK;
 getch();
}
/*
output==>
enter the size of array
7
element are=>
10      3       335     33      355     217     536

element are=>
3       10      217     33      355     335     536

Time Complexity is=>0
*/
```

```
--------------------------------------------------------------------
Assignment Name:Program for Heapsort Ascending using Adjust/Heapify
Class: MCA -II (Division B)                  Lab: CA Lab V (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])
       return;
      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<<"\nENTER SIZE OF THE ARRAY:=>";
   cin>>n;
   for(i=0;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<<"\nAFTER HEAP SORTING\n";
   for(i=0;i<n;i++)
   {
      if(i%8==0)
     cout<<"\n";
      cout<<"\t"<<b[i];
   }
    cout<<"\nTHE TIME COMPLEXITY IS :=>"<<((e-s) / CLK_TCK);
   getch();
}
```

```
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 NameProgram for Heapsort Descending using Adjust/Heapify
Class: MCA -II (Division B)                  Lab: CA Lab V (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])
       return;
      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<<"\nENTER 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<<"\nAFTER HEAP SORTING\n";
   for(i=1;i<=n;i++)
   {
     if(i%8==0)
     cout<<"\n";
      cout<<"\t"<<b[i];
   }
    cout<<"\nTHE TIME COMPLEXITY IS :=>"<<((e-s) / CLK_TCK);
   getch();
}
//Output

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      46      43      41      40      38
        36      36      34      33      33      32      30      30
        30      30      29      28      27      26      26      24
        23      22      22      22      21      20      20      17
        13      10      8       8       8       8       8       8
        8       6       5       5       5       5       2       2
        1       0       0
THE TIME COMPLEXITY IS :=>0
```

```
---------------------------------------------------------------------
Assignment Name: Program for Binary Search
Class: MCA -II (Division B)                  Lab: CA Lab V (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<<"\nEnter the no. 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();
      int x,y;
      clock_t e,s;
      binary b;
      b.get();
      b.sort();
      cout<<"\n Sorted elements are"<<endl;
      b.put();
      cout<<"\nEnter elt u want to find="<<endl;
      cin>>x;
      s=clock();
      y=b.bsearch(x);
      cout<<"find="<<y<<endl;
      b.put();
      e=clock();
      cout<<"\n time coplexity="<<((e-s)/CLK_TCK);
      getch();
     }

     /* Enter the no. of elements100

      Sorted elements are
     32      61      79      211     665     705     830
     855     1085    1394    1478    2179    2279    2291    2428
     2611    3273    3303    3411    3596    3915    3941    3942
     4233    4343    4388    4808    4980    5064    5086    5288
     5378    5495    5596    5899    6677    6702    6889    7102
     7114    7536    7730    8098    8123    8172    8196    8279
     8461    8535    8885    9020    9160    9221    9226    9497
     9679    9821    9881    10631   10739   10765   10983   11284
     11286   11622   11939   11994   12069   12094   12467   12909
     12985   13233   13424   13710   13895   13964   14006   14026
     14400   14718   15162   15287   15744   16102   16217   16309
     16552   16790   17374   17510   17724   17786   18107   18464
     18997   19051   19190   19492   19673
     Enter elt u want to find=
     19051
     find=97
     32      61      79      211     665     705     830
     855     1085    1394    1478    2179    2279    2291    2428
     2611    3273    3303    3411    3596    3915    3941    3942
     4233    4343    4388    4808    4980    5064    5086    5288
```

```
5378    5495    5596    5899    6677    6702    6889    7102 */
--------------------------------------------------------------------
Assignment Name: Program for MAXMIN
Class: MCA -II (Division B)                  Lab: CA Lab V (DAA)
--------------------------------------------------------------------
#include<iostream.h>
#include<conio.h>
#include<stdlib.h>
#include<time.h>

class sai
{
int a[100],n,max,min;
public:
int i,j;
void get();
void put();
void maxmin(int,int,int&,int&);
void show();
void call();
};
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::call()
{
maxmin(1,n,0,0);
}
void sai::maxmin(int i,int j,int &max1,int &min1)
{

int mid,max2,min2;
if(i==j)
max1=min1=a[i];
else if(i==j-1)
{
 if(a[i]<a[j])
  {
```

```
max1=a[j];
min1=a[i];
}
else
{
max1=a[i];
min1=a[j];
}
}
else
{
mid=(i+j)/2;
maxmin(i,mid,max1,min1);
maxmin(mid+1,j,max2,min2);

if(max1<max2)
max1=max2;
if(min1>min2)
min1=min2;
}
 cout<<"\nmax "<<max<<"min "<<min;
max=max1;
min=min1;
}
void sai::show()
{
cout<<"\nThe maximum element is=>"<<max;
cout<<"\nThe minimum element is=>"<<min;
}
void main()
{
clrscr();
sai s;
clock_t e,l;
s.get();
s.put();
e=clock();
s.call();
l=clock();
s.show();
cout<<"\n The time compexity is =>"<<(l-e)/CLK_TCK;
getch();
}
```

```
/*Output:=>

 Enter the size of array=>87

 Show the element of array=>
10      3       335     33      355     217     536
195     700     949     274     444     108     698     564
41      165     815     685     764     827     959     219
426     952     839     923     810     451     604     661
599     549     720     113     406     121     671     474
491     564     344     868     264     179     423     694
163     538     645     623     3       787     268     461
386     376     581     603     279     170     805     294
333     408     240     413     54      494     983     1
409     69      73      254     974     355     404     197
197     211     249     758     889     905     735     461
The maximum element is=>974
The minimum element is=>1
The time compexity is =>0
```

```
-------------------------------------------------------------------
Assignment Name: Program for Ascending Merge Sort
Class: MCA –II (Division B)                     Lab: CA Lab V (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<<"Enter the Element:";
          cin>>n;
          h=n-1;
          l=0;
          m.read();
          cout<<"\n\nDisplay the Array Element=\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();
}
/*
Enter the Element:
100


Display the Array Element=

19804   17255   18957   16141   6787    2154    9927    8705
17349   14813   15565   3901    14024   8516    19648   18370
4055    11292   4992    1110    7476    5595    12493   3866
14456   18640   5358    6888    16266   15283   7916    16900
4270    15312   18368   4183    1284    7031    4857    15628
19699   19002   4237    10636   10738   1835    12910   16762
19219   1880    9108    11936   8200    18722   15998   10181
7883    8327    4978    17678   13152   3734    8547    16043
4115    15164   2118    19729   14034   1409    4154    4385
19007   14394   13994   13082   13790   8770    16758   17246
8118    18431   14688   19216   7916    11807   14317   13262
15767   7836    8826    11958   15006   4461    4977    14786
11832   1552    13991   5838
After Sorting=
1110    1284    1409    1552    1835    1880    2118    2154
3734    3866    3901    4055    4115    4154    4183    4237
4270    4385    4461    4857    4977    4978    4992    5358
5595    5838    6787    6888    7031    7476    7836    7883
7916    7916    8118    8200    8327    8516    8547    8705
8770    8826    9108    9927    10181   10636   10738   11292
11807   11832   11936   11958   12493   12910   13082   13152
13262   13790   13991   13994   14024   14034   14317   14394
14456   14688   14786   14813   15006   15164   15283   15312
15565   15628   15767   15998   16043   16141   16266   16758
16762   16900   17246   17255   17349   17678   18368   18370
18431   18640   18722   18957   19002   19007   19216   19219
19648   19699   19729   19804
Time Com.=   0
*/
```

```
-------------------------------------------------------------------
Assignment Name: Program for Descending Merge Sort
Class: MCA -II (Division B)                    Lab: CA Lab V (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<<"Enter the Element:";
        cin>>n;
        h=n-1;
        l=0;
        m.read();
        cout<<"\n\nDisplay the Array Element=\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==>


Enter the Element:
100

Display the Array Element=

11640   9884    9870    18044   7923    7011    8060    6568
2456    14320   19449   13744   18477   8470    9689    12819
2542    19612   19459   9428    18308   13331   3589    16134
3329    11998   19117   1064    13236   7421    11678   7530
7668    6644    19583   11495   2341    2803    4967    13768
1624    12879   11387   13803   15999   14254   17233   1966
18776   1926    4346    17166   6439    18256   11193   6701
17009   17143   9993    15216   10780   14475   17170   19625
7817    6752    16489   16901   6587    9746    4273    1574
18096   19075   16054   19844   19912   4562    4060    17170
4441    269     4882    6201    5842    2055    6719    11942
19944   14561   15569   1021    1548    19477   9735    11926
13490   6978    2030    13104

After Sorting=
19944   19912   19844   19625   19612   19583   19477   19459
19449   19117   19075   18776   18477   18308   18256   18096
18044   17233   17170   17170   17166   17143   17009   16901
16489   16134   16054   15999   15569   15216   14561   14475
14320   14254   13803   13768   13744   13490   13331   13236
13104   12879   12819   11998   11942   11926   11678   11640
11495   11387   11193   10780   9993    9884    9870    9746
9735    9689    9428    8470    8060    7923    7817    7668
7530    7421    7011    6978    6752    6719    6701    6644
6587    6568    6439    6201    5842    4967    4882    4562
4441    4346    4273    4060    3589    3329    2803    2542
2456    2341    2055    2030    1966    1926    1624    1574
1548    1064    1021    269

Time Com.=   0
*/
```

```
---------------------------------------------------------------------
Assignment Name: Program for Ascending Quick Sort
Class: MCA -II (Division B)                    Lab: CA Lab V (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=>\n";
cin>>n;
   for(int i=1;i<=n;i++)
   {
   a[i]=random(20000);
   }
}
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+1);
  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;
int j=p;
do
  {
     do
      i=i+1;
      while(a[i]<v);
       do
       j=j-1;
     while(a[j]>v);
     if(i<j)
     {
     p=a[i];
     a[i]=a[j];
     a[j]=p;
     }
     }while(i<=j);
      a[m]=a[j];
      a[j]=v;
      return j;
}
void main()
{
clrscr();
clock_t s,e;
q_sort q;
q.get();
cout<<"\n display  the elment of array before sort=>\n\n";
q.put();
s=clock();
q.quick_sort(1,n);
e=clock();
cout<<"\n display  the elment of array before sort=>\n\n";
q.put();
cout<<"\n the time complexity=>"<<(e-s)/CLK_TCK;
getch();
}
 /*
 OUTPUT==>
 enter the size of array=>
100

 enter the element in array=>
 display  the elment of array before sort=>

211     79      6702    665     7114    4343    10739
3915    14006   18997   5495    8885    2179    13964   11286
830     3303    16309   13710   15287   16552   19190   4388
8535    19051   16790   18464   16217   9020    12094   13233
11994   10983   14400   2279    8123    2428    13424   9497
9821    11284   6889    17374   5288    3596    8461    13895
3273    10765   12909   12467   61      15744   5378    9221
7730    7536    11622   12069   5596    3411    16102   5899
6677    8172    4808    8279    1085    9881    19673   32
8196    1394    1478    5086    19492   7102    8098    3942
```

```
3941    4233    4980    15162   17786   18107   14718   9226
10631   705     2611    9160    9679    11939   5064    2291
14026   12985   17724   17510   855
 display  the elment of array before sort=>

32      61      79      211     665     705     830
855     1085    1394    1478    2179    2279    2291    2428
2611    3273    3303    3411    3596    3915    3941    3942
4233    4343    4388    4808    4980    5064    5086    5288
5378    5495    5596    5899    6677    6702    6889    7102
7114    7536    7730    8098    8123    8172    8196    8279
8461    8535    8885    9020    9160    9221    9226    9497
9679    9821    9881    10631   10739   10765   10983   11284
11286   11622   11939   11994   12069   12094   12467   12909
12985   13233   13424   13710   13895   13964   14006   14026
14400   14718   15162   15287   15744   16102   16217   16309
16552   16790   17374   17510   17724   17786   18107   18464
18997   19051   19190   19492   19673

Time complexity=>0
*/
```

```
-------------------------------------------------------------------
Assignment Name: Program for Descending Quick Sort
Class: MCA -II (Division B)                       Lab: CA Lab V (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;
cout<<"\n enter the element in array=>";
   for(int i=1;i<=n;i++)
   {
   a[i]=random(20000);
   }
}
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+1);
  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;
int j=p;
  do
  {
     do
      i=i+1;
      while(a[i]>v);
       do
       j=j-1;
     while(a[j]<v);
     if(i<j)
     {
     p=a[i];
     a[i]=a[j];
     a[j]=p;
     }
     }while(i<=j);
      a[m]=a[j];
      a[j]=v;
      return j;
}
void main()
{
clrscr();
clock_t s,e;
q_sort q;
q.get();
cout<<"\n display  the elment of array before sort=>\n\n";
q.put();
s=clock();
q.quick_sort(1,n);
e=clock();
cout<<"\n display  the elment of array before sort=>\n\n";
q.put();
cout<<"\n the time complexity=>"<<(e-s)/CLK_TCK;
getch();
}
/*
OUTPUT==>
 enter the size of array=>
100

 enter the element in array=>
 display  the elment of array before sort=>

211     79      6702    665     7114    4343    10739
3915    14006   18997   5495    8885    2179    13964   11286
830     3303    16309   13710   15287   16552   19190   4388
8535    19051   16790   18464   16217   9020    12094   13233
11994   10983   14400   2279    8123    2428    13424   9497
9821    11284   6889    17374   5288    3596    8461    13895
3273    10765   12909   12467   61      15744   5378    9221
7730    7536    11622   12069   5596    3411    16102   5899
6677    8172    4808    8279    1085    9881    19673   32
8196    1394    1478    5086    19492   7102    8098    3942
3941    4233    4980    15162   17786   18107   14718   9226
10631   705     2611    9160    9679    11939   5064    2291
```

```
14026   12985   17724   17510   855
 display  the elment of array before sort=>

19673   19492   19190   19051   18997   18464   18107
17786   17724   17510   17374   16790   16552   16309   16217
16102   15744   15287   15162   14718   14400   14026   14006
13964   13895   13710   13424   13233   12985   12909   12467
12094   12069   11994   11939   11622   11286   11284   10983
10765   10739   10631   9881    9821    9679    9497    9226
9221    9160    9020    8885    8535    8461    8279    8196
8172    8123    8098    7730    7536    7114    7102    6889
6702    6677    5899    5596    5495    5378    5288    5086
5064    4980    4808    4388    4343    4233    3942    3941
3915    3596    3411    3303    3273    2611    2428    2291
2279    2179    1478    1394    1085    855     830     705
665     211     79      61      32
 Time complexity=>0
*/
```

```
--------------------------------------------------------------------
Assignment Name: Stranssen’s Matrix Multiplication
Class: MCA –II (Division B)                  Lab: CA Lab V (DAA)
--------------------------------------------------------------------
#include<iostream.h>
#include<conio.h>
#include<time.h>
class mmul
{
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 mmul::get()
{
cout<<"enter the matrix1=";
for( i=1;i<=2;i++)
for( j=1;j<=2;j++)
cin>>a[i][j];

cout<<"enter the matrix2=";
for( i=1;i<=2;i++)
for( j=1;j<=2;j++)
cin>>b[i][j];
}
void mmul::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 mmul::put()
{
for(int i=1;i<=2;i++)
{
for(int j=1;j<=2;j++)
cout<<c[i][j]<<" ";
cout<<" \n";
}
}
void main()
{
clrscr();
mmul m;
clock_t e,s;
m.get();
s=clock();
m.formula();
cout<<"\n output="<<endl;
m.put();
e=clock();
cout<<"\n Time comlexity="<<((e-s)/CLK_TCK);
getch();
}
/*  OUTPUT :=
enter the matrix1=
1 1
1 1
enter the matrix2=
2 2
2 2

 output=
4 4
4 4

 Time comlexity=0
   */
```

```
------------------------------------------------------------------
Assignment Name: Program for knapsack solution
Class: MCA -II (Division B)                    Lab: CA Lab V (DAA)
------------------------------------------------------------------
#include<iostream.h>
#include<conio.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<<"Enter the ele 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<<"------------------";
}
void main()
{
clrscr();
knapsack k;
k.get();
k.order();
k.knap(m,n);
k.show();
getch();
}


/*output:-
Enter the ele Size& Sack Size
3 20

Enter the Profit=>
25 24 15

Enter the Weight=>
18 15 10
1 0.5 0
---------------
Profit=>31.5
------------------
*/
```

```
-------------------------------------------------------------------
Assignment Name: Minimum cost spanning tree using Prims Algorithm
                 (Greddy Approch)
Class: MCA -II (Division B)                  Lab: CA Lab V (DAA)
-------------------------------------------------------------------
# include<iostream.h>
# include<conio.h>
# define SIZE 20
# define INFINITY 32767

/*This function finds the minimal spanning tree by Prim's Algorithm
*/

void Prim(int G[][SIZE], int nodes)
{
      int select[SIZE], i, j, k;
      int min_dist, v1, v2,total=0;

for (i=0 ; i<nodes ; i++)   // Initialize the selected vertices
list
                select[i] = 0;

      cout<<"\n\n The Minimal Spanning Tree Is :\n";
      select[0] = 1;
      for (k=1 ; k<nodes ; k++)
      {
               min_dist = INFINITY;
     for (i=0 ; i<nodes ; i++) // Select an edge such that one
vertex is
     {                        // selected and other is not and the
edge
      for (j=0 ; j<nodes ; j++) // has the least weight.
      {
if (G[i][j] && ((select[i] && !select[j]) || (!select[i] &&
select[j])))
               {
                if (G[i][j] < min_dist)//obtained edge with minimum
wt
                     {
                         min_dist = G[i][j];
                         v1 = i;
                         v2 = j;   //picking up those vertices
                     }
               }
          }
      }
      cout<<"\n Edge "<<v1 <<" --> "<<v2 <<" weight "<<min_dist;
      select[v1] = select[v2] = 1;
      total =total+min_dist;
      }
     cout<<"\n\n\t Total Path Length Is "<<total;
}
```

```
void main()
{
      int G[SIZE][SIZE], nodes;
      int v1, v2, length, i, j, n;

      clrscr();
      cout<<"\n\t Prim'S Algorithm\n";

      cout<<"\n Enter Number of Nodes in The Graph  ";
      cin>>nodes;
      cout<<"\n Enter Number of Edges in The Graph  ";
      cin>>n;

      for (i=0 ; i<nodes ; i++)       // Initialize the graph
          for (j=0 ; j<nodes ; j++)
               G[i][j] = 0;
      //entering weighted graph
      cout<<"\n Enter edges and weights \n";
      for (i=0 ; i<n; i++)
      {
               cout<<"\n Enter Edge by V1 and V2 :";
               cin>>v1>>v2;
               cout<<"\n Enter corresponding weight :";
               cin>>length;
               G[v1][v2] = G[v2][v1] = length;
      }

      cout<<"\n\t";
      Prim(G,nodes);
      getch();
}

 /*        Prim'S Algorithm

 Enter Number of Nodes in The Graph  5

 Enter Number of Edges in The Graph  7

 Enter edges and weights

 Enter Edge by V1 and V2 :0 1

 Enter corresponding weight :10

 Enter Edge by V1 and V2 :1 2

 Enter corresponding weight :1

 Enter Edge by V1 and V2 :2 3

 Enter corresponding weight :2

 Enter Edge by V1 and V2 :3 4
 Enter corresponding weight :3

 Enter Edge by V1 and V2 :4 0

 Enter corresponding weight :5

 Enter Edge by V1 and V2 :1 3

 Enter corresponding weight :6

 Enter Edge by V1 and V2 :4 2

 Enter corresponding weight :7



 The Minimal Spanning Tree Is :

 Edge 0 --> 4 weight 5
 Edge 3 --> 4 weight 3
 Edge 2 --> 3 weight 2
 Edge 1 --> 2 weight 1

         Total Path Length Is 11

 */
```

```
------------------------------------------------------------------
Assignment Name: minimum cost spaning tree using prims algorithm
Class: MCA -II (Division B)                  Lab: CA Lab V (DAA)
------------------------------------------------------------------
#include<iostream.h>
#include<conio.h>

int n;
class single
{
    int
v,cost[10][10],i,j,s[10],e[10],near1[10],t[10][3],m,minedge,k,l,min
cost;
    int jindex;
    float dist[10];
    public:
        void get();
        void prim();
        void display();
};
void single::get()
{      m=1;
 minedge=9999;
   cout<<"enter the no of vertices\n";
   cin>>n;
   cout<<"enter the adecancy 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;
      else
      {
      e[m]=cost[i][j];
      if(e[m]<minedge)
     {
     minedge=e[i];k=i;l=j;
     }

      }


   }
}
void single::prim()
{
t[1][1]=k;t[1][2]=l;
mincost=cost[k][l];
for(i=1;i<=n;i++)
{
if(cost[i][l]<cost[i][k])
near1[i]=l;
else
near1[i]=k;
}
near1[k]=near1[l]=0;
int minj=9999;
for(i=2;i<=n-1;i++)
{             minj=9999;

     for(j=1;j<=n;j++)
     {
          if(near1[j]!=0)
          {
          if(cost[j][near1[j]]<minj)
          {
          minj=cost[j][near1[j]];
          jindex=j;
          }
          }
     }
     t[i][1]=jindex;
     t[i][2]=near1[jindex];
     mincost=mincost+cost[jindex][near1[jindex]];
     near1[jindex]=0;
     for(int k1=1;k1<=n;k1++)
     {
     if (near1[k1]!=0 && cost[k1][near1[k1]] > cost[k1][jindex] )
     near1[k1]=jindex;
     }

}
cout<<"\nMincost="<<mincost;

  }
void single::display()
{
   cout<<endl;
   cout<<"\nMinimum Spanning Tree Path as follow\n";
   cout<<t[1][1]<<"->"<<t[1][2];

   for(i=2;i<n;i++)
   {
   cout<<"->";
   cout<<t[i][1];
   }


}
void main()
{
   single d;
   clrscr();
   d.get();
   d.prim();
   d.display();
   getch();
}
/* OUTPUT :-
 enter the no of vertices
7
enter the adecancy matrix
-1 28 -1 -1 -1 10 -1
28 -1 16 -1 -1 -1 14
-1 16 -1 12 -1 -1 -1
-1 -1 12 -1 22 -1 18
-1 -1 -1 22 -1 25 24
10 -1 -1 -1 25 -1 -1
-1 14 -1 18 24 -1 -1

Mincost=99

Minimum Spanning Tree Path as follow
1->6->5->4->3->2->7
*/
```

```
--------------------------------------------------------------------
Assignment Name: Prog.to Demostrate Kruskal Algorithm.
Class: MCA -II (Division B)                   Lab: CA Lab V (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=0;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=0;
 sum=0;
 for(i=0;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][0]=v1;//storing the minimum edge in array tree[]
                tree[k][1]=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=0;i<tot_nodes-1;i++)
           {
                cout<<tree[i][0];
                cout<<" - ";
                cout<<tree[i][1];
                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=0;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
Graph Creation by adjacency matrix
 Enter Total number of nodes: 5

 Enter Total number of edges: 7

 Enter Edge in (V1 V2)form 0 1

 Enter Corresponding Cost 10

 Enter Edge in (V1 V2)form 0 3

 Enter Corresponding Cost 6

 Enter Edge in (V1 V2)form 0 4

 Enter Corresponding Cost 5

 Enter Edge in (V1 V2)form 1 2

 Enter Corresponding Cost 1

 Enter Edge in (V1 V2)form 2 4

 Enter Corresponding Cost 7

 Enter Edge in (V1 V2)form 2 3

 Enter Corresponding Cost 2

 Enter Edge in (V1 V2)form 3 4

 Enter Corresponding Cost 3

 Spanning tree is...
--------------------------
[1 - 2][2 - 3][3 - 4][0 - 4]
--------------------------
```

```
Cost of Spanning Tree is = 11      */
-------------------------------------------------------------------
Assignment Name: Program for Single Source shortest path
Class: MCA –II (Division B)                  Lab: CA Lab V (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<<"enter the no. of vertices=\n";
    cin>>n;
    cout<<"enter 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 Name: Program for All Pair Shortest Path
Class: MCA -II (Division B)                  Lab: CA Lab V (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";
 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 element3

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 Name: Prog.to Demostrate Breath First Traversal
Class: MCA -II (Division B)                Lab: CA Lab V (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();
};

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];
     }
}

void bfstree::bfs()
{
     reach[1]=1;
     f++;
     r++;
     q[r]=index;
     cout<<"\nBFS is ";
     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();
     b.dbfs();
     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

BFS is 1        2       3       4       6       5
```

```
--------------------------------------------------------------------
Assignment Name: Prog.to Demostrate Depth First Traversal
Class: MCA -II (Division B)                     Lab: CA Lab V (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=0;i<n;i++)
      visited[i]=0;
     cout<<"\nEnter the adjancy matrix:";
     for(i=0;i<n;i++)
     {
      for(j=0;j<n;j++)
          cin>>a[i][j];
     }
     dfs(0);
}
void dfstree::dfs(int v)
{
     int k;
     visited[v]=1;
     cout<<"\t"<<v+1;
     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 node5
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 Name: Prog. to Demostrate Topological Sort
Class: MCA -II (Division B)                 Lab: CA Lab V (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 no. 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";
   for(int k=1;k<=n;k++)
   {
   cout<<"\n Indegree  :\n";
   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 no. of vertices7

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 Name: Program for finding maximum using rules of removal
                 of  Recursion
Class: MCA -II (Division B)                     Lab: CA Lab V (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<<"\n ENTER THE NUMBER OF ELEMENTS:=>";
  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;
      return k;
}
void maximum::print()
{
  cout<<"\nMAX POSITION:"<<k<<endl;
  cout<<"\nMAX ELEMENT :"<<a[k];
}
main()
{
  clrscr();
  maximum m;
  m.read();
  m.max(1);
  m.print();
  getch();
  return 0;
}

//Output

 ENTER THE NUMBER OF ELEMENTS:=>50
346     130     10982   1090    11656   7117    17595   6415
22948   31126   9004    14558   3571    22879   18492   1360
5412
26721   22463   25047   27119   31441   7190    13985   31214
27509
30252   26571   14779   19816   21681   19651   17995   23593
3734
13310   3979    21995   15561   16092   18489   11288   28466
8664
5892    13863   22766   5364    17639   21151
MAX POSITION:22

MAX ELEMENT :31441
```

```
-------------------------------------------------------------------
Assignment Name: Program for searching an element using rules of
                 Removal Of recursion
Class: MCA -II (Division B)                  Lab: CA Lab V (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<<"\nEnter 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<<"\nEnter The Initial Position:";
   cin>>p;
   cout<<"\nEnter 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<<"\nElement 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<<"\nItem 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

Enter The Number Of Elements:100

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 Name: Program for Binomial Coefficient
Class: MCA -II (Division B)                      Lab: CA Lab V (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<<"Enter Level : ";
          cin>>n;
          for(int i=0;i<=n;i++)
          cout<<binomial(n,i)<<"\t";
          getch();
}

//Output

Enter Level : 5
1       5       10      10      5       1
```

```
-------------------------------------------------------------------
Assignment Name: Program for finding Binomial Coefficient using
                 Rules of Removal of recursion
Class: MCA -II (Division B)                  Lab: CA Lab V (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<<"\nENTER THE VALUE OF N :";
 cin>>n;
 cout<<"\nENTER THE VALUE OF M :";
 cin>>m;
 binomial(n,m);
 cout<<endl<<"BINOMIAL COEFFIENT 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

ENTER THE VALUE OF N :5

ENTER THE VALUE OF M :1

BINOMIAL COEFFIENT IS :5
```

```
-----------------------------------------------------------------
Assignment Name: Program for n Queen for all solution
Class: MCA -II (Division B)                    Lab: CA Lab V (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 Name: Program for n Queen for in equivalent solution
Class: MCA -II (Division B)                Lab: CA Lab V (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()
 {
     for(int i = 1;i <= n;i++)
     {
          cout<<"\n";
          for(int j = 1;j <= n;j++)
          {
               if(x[i] == j)
                    cout<<x[i]<<"\t";
                 // else
                //  cout<<"*\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();
 }
```

```
Enter number of queen : 4

Solution Number 1 :
2       4       1       3
```

```
------------------------------------------------------------------
Assignment Name: Program for Graph Coloring
Class: MCA -II (Division B)                   Lab: CA Lab V (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 is1     2       1
Colour of graph is2     1       2
```

```
------------------------------------------------------------------
Assignment Name: Program for code1 using postfix expression
Class: MCA -II (Division B)                   Lab: CA Lab V (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 Name: Program for code2
Class: MCA -II (Division B)                Lab: CA Lab V (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;
      icnt=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
```

```
/*****************************************************************
*
Implementation of 0/1 knapsack problem using Dynamic programming
******************************************************************
*/
#include<iostream.h>
#include<conio.h>
#include<stdlib.h>
int table[5][6];
int w[]={0,2,3,4,5};
int v[]={0,3,4,5,6};
int W=5;
int n=4;

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;
 }
}
}
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...";
  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(j<w[i])
          {
               table[i][j]=table[i-1][j];
          }
          else if(j>=w[i])
          {
               val1=table[i-1][j];
               val2=v[i]+table[i-1][j-w[i]];
               table[i][j]=max(val1,val2);
          }
       }
       }
       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.DKP(n,W,w,v);
getch();
}
```

```
//PROGRAM FOR LCS


#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 upright 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()
{
     char ch;
     do
     {
           lcs();
           cout<<"\nContinue(y/n):";
           cin>>ch;
     }
     while(ch=='y'||ch=='Y');
}
```