INSTITUTE OF TECHNOLOGY & MANAGEMENT
www.itmgoi.in

श्रेष्ठ इंडस्ट्री इन्टरफेस के लिए
CMAI, AICTE & RGPV
द्वारा पुरस्कृत

INSTITUTE OF TECHNOLOGY
& MANAGEMENT
GWALIOR • MP • INDIA

Laboratory Manual

Analysis And Design Of Algorithm
(IT-403)

For

Second Year Students
Department of Information Technology

Department of Information Technology

Vision of CSE Department

The department envisions to nurture students to become technologically proficient, research competent and socially accountable for the welfare of the society.

Mission of the CSE Department

- To provide high quality education through effective teaching-learning process emphasizing active participation of students.
- To build scientifically strong engineers to cater to the needs of industry, higher studies, research and startups.
- To awaken young minds ingrained with ethical values and professional behaviors for the betterment of the society.

Programme Educational Objectives

Graduates will be able to

- Our engineers will demonstrate application of comprehensive technical knowledge for innovation and entrepreneurship.
- Our graduates will employ capabilities of solving complex engineering problems to succeed in research and/or higher studies.
- Our graduates will exhibit team-work and leadership qualities to meet stakeholder business objectives in their careers.

- Our graduates will evolve in ethical and professional practices and enhance socioeconomic contributions to the society.

Programme Outcomes (POs)

Engineering Graduates will be able to:

1. **Engineering knowledge**: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **Problem analysis**: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
3. **Design/development of solutions**: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
4. **Conduct investigations of complex problems**: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
5. **Modern tool usage**: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
6. **The engineer and society**: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
7. **Environment and sustainability**: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
8. **Ethics**: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. **Individual and team work**: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
10. **Communication**: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

11. **Project management and finance**: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
12. **Life-long learning**: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

Course Outcomes

CO1 :	Learn, Apply & Analyze the complexity in Divide and Conquer techniques for suitable problems.
CO2 :	Apply and identify the Optimal Solution using the Greedy Approach for appropriate problems.
CO3 :	Compute and Analyze the problems by using Dynamic Programming approach.
CO4 :	Apply the concept of Backtracking and Branch & Bound for solving the suitable problems, and enhance the performance of the algorithm
CO5 :	Learn the concept of NP- completeness and Apply the various operations on tree & Graph data structures

Course	Course Outcomes	CO Attainment	P01	P02	P03	P04	P05	P06	P07	P08	P09	P010	PO11	P012	PSO1	PSO2	PSO3
CO1	Learn, Apply & Analyze the complexity in Divide and Conquer techniques for suitable problems		2	2	0	0	1	0	0	1	0	0	0	1	2	0	0
CO2	Apply and identify the Optimal Solution using the Greedy Approach for appropriate problems.		2	3	3	3	1	0	2	1	0	0	0	1	2	0	0
CO3	Compute and Analyze the problems by using Dynamic Programming approach		2	3	3	3	1	0	0	1	0	0	0	1	2	0	0
CO4	Apply the concept of Backtracking and Branch & Bound for solving the suitable problems, and enhance the performance of the algorithm		2	2	2	2	1	0	0	1	0	0	0	1	2	0	0
CO5	Learn the concept of NP completeness and Apply the various operations on tree & Graph data structures		2	1	1	1	0	0	0	1	1	0	0	1	0	0	0
Average			**2**	**2.2**	**2.25**	**2.25**	**0**	0	**2**	**0**	**1**	**0**	0	**1**	**2**	**0**	0

List Of Programs

1. Write a program for Iterative and Recursive Binary Search

1.a //Iterative binary search

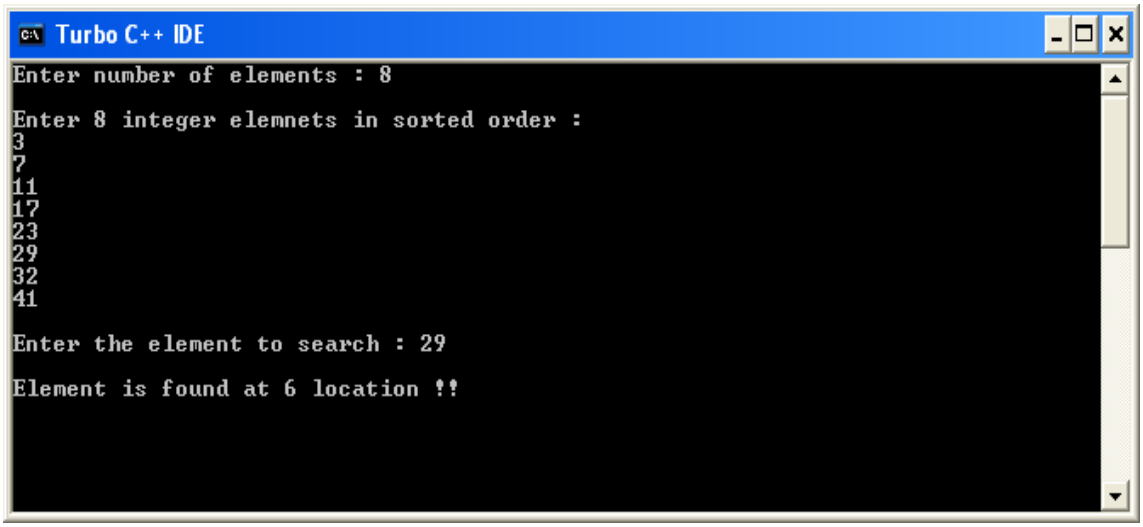
```
#include <iostream.h>
#include<conio.h>
int IterativeBinarySearch(int A[], int key, int l, int r)
{
 while (l <= r)
 {
  int m = (l + r) / 2;
  if (key < A[m])
  {
   r = m - 1;
  }
  else
   if (key > A[m])
   {
    l = m + 1;
   }
   else
   {
    return m;
      }
      }
       return -1;
       }
```

```
void main ()
 {
 clrscr( );
 int arr[20],n,i,e,loc;
 cout<<"Enter number of elements : ";
 cin>>n;
 cout<<"Enter "<<n<<" integer elemnets in sorted order :\n";
 for(i=0;i<n;i++)
  cin>>arr[i];
 cout<<"Enter the element to search : ";
 cin>>e;
 loc = IterativeBinarySearch(arr, e, 0, n-1);
 if(loc==-1)
  cout<<"Element not present !!";
 else
  cout<<"Element os found at "<<loc+1<<" location";
 getch( );
 }
```

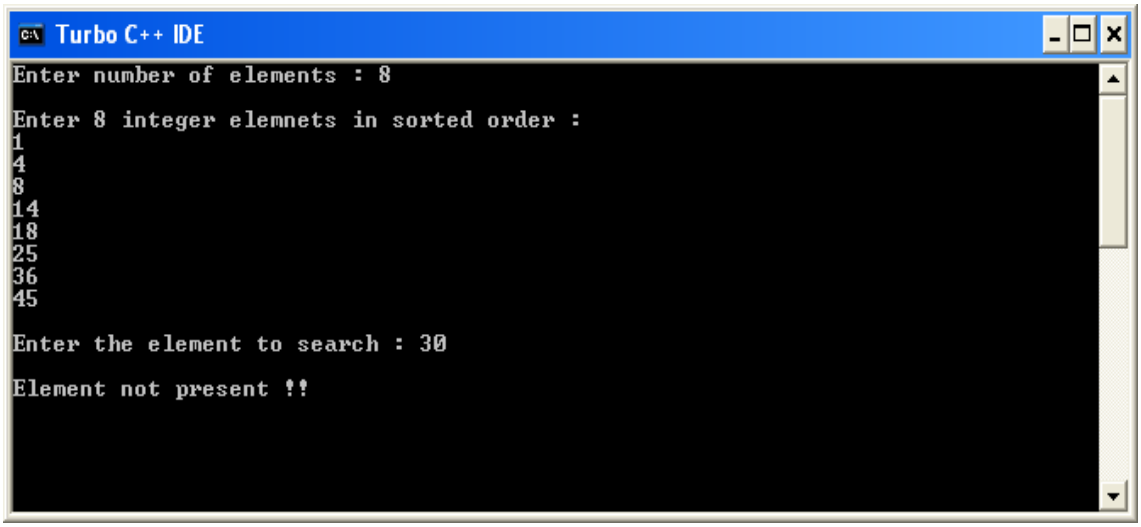
```
//Recursive binary search
#include<iostream.h>
#include<conio.h>
int binarysearch(int a[],int n,int low,int high)
 {
 int mid;
 if (low > high)
  return -1;
 mid = (low + high)/2;
 if(n == a[mid])
  {
  cout<<"The element is at position "<<mid+1;
  return 0;
  }
 if(n < a[mid])
  {
  high = mid - 1;
  binarysearch(a,n,low,high);
  }
 if(n > a[mid])
  {
  low = mid + 1;
  binarysearch(a,n,low,high);
  }
 }
void main( )
```

```
{
clrscr( );
int a[50];
int n,e,i,result;
cout<<"Enter the number of elements ";
cin>>n;
cout<<"Enter the elements :\n";
for(i=0;i<n;i++)
 cin>>a[i];
cout<<"Enter the element to be searched : ";
cin>>e;
result = binarysearch(a,e,0,n-1);
if(result == -1)
 cout<<"Element not found !!";
getch( );
}
```

Output of program for Iterative binary search (Program 1a)

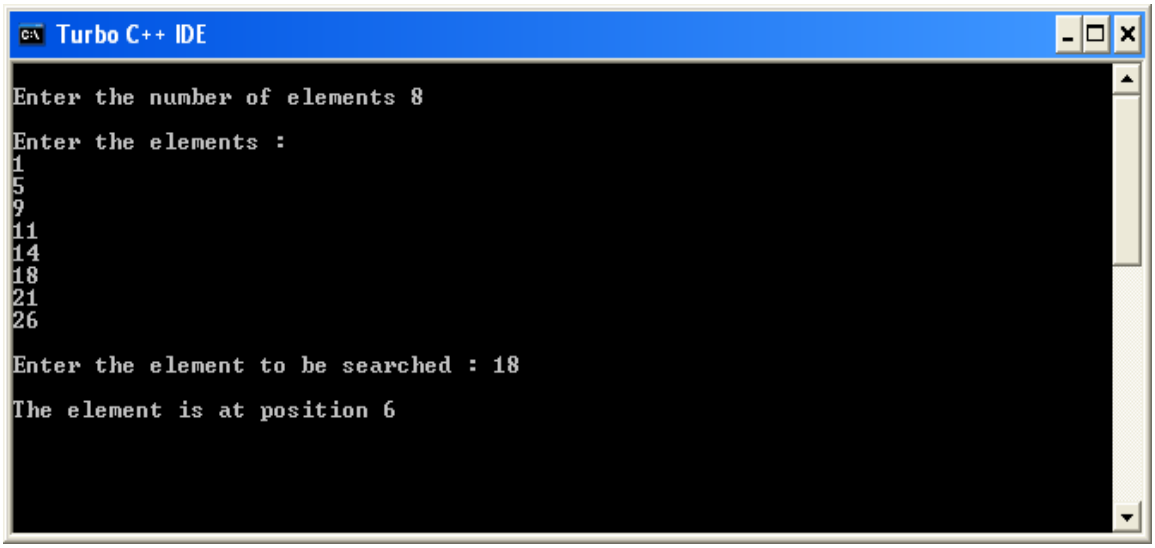


In case of positive search

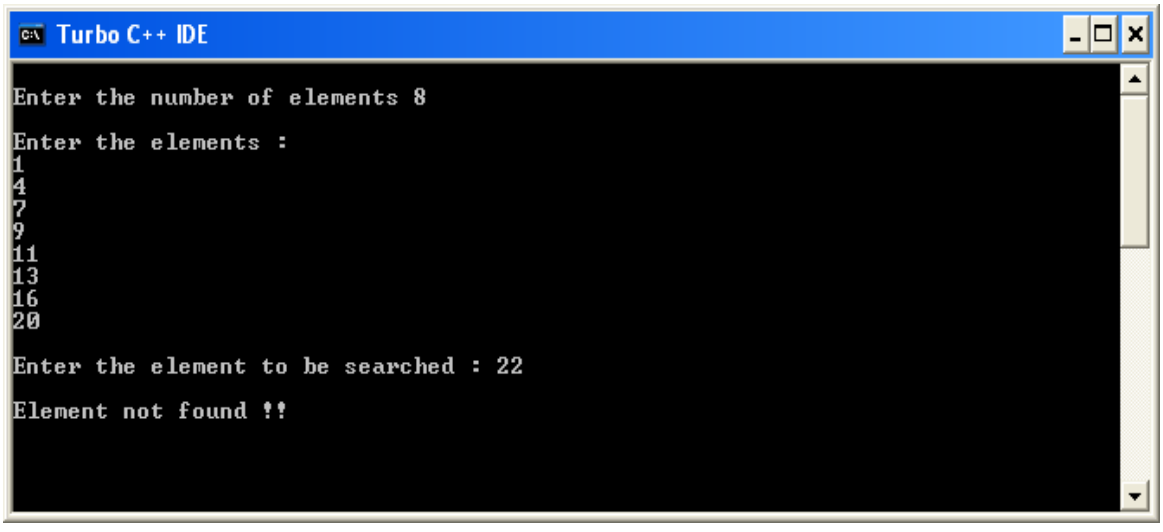


In case of negative search

Output of program for Recursive binary search (Program 1.b)



In case of positive search



In case of negative search

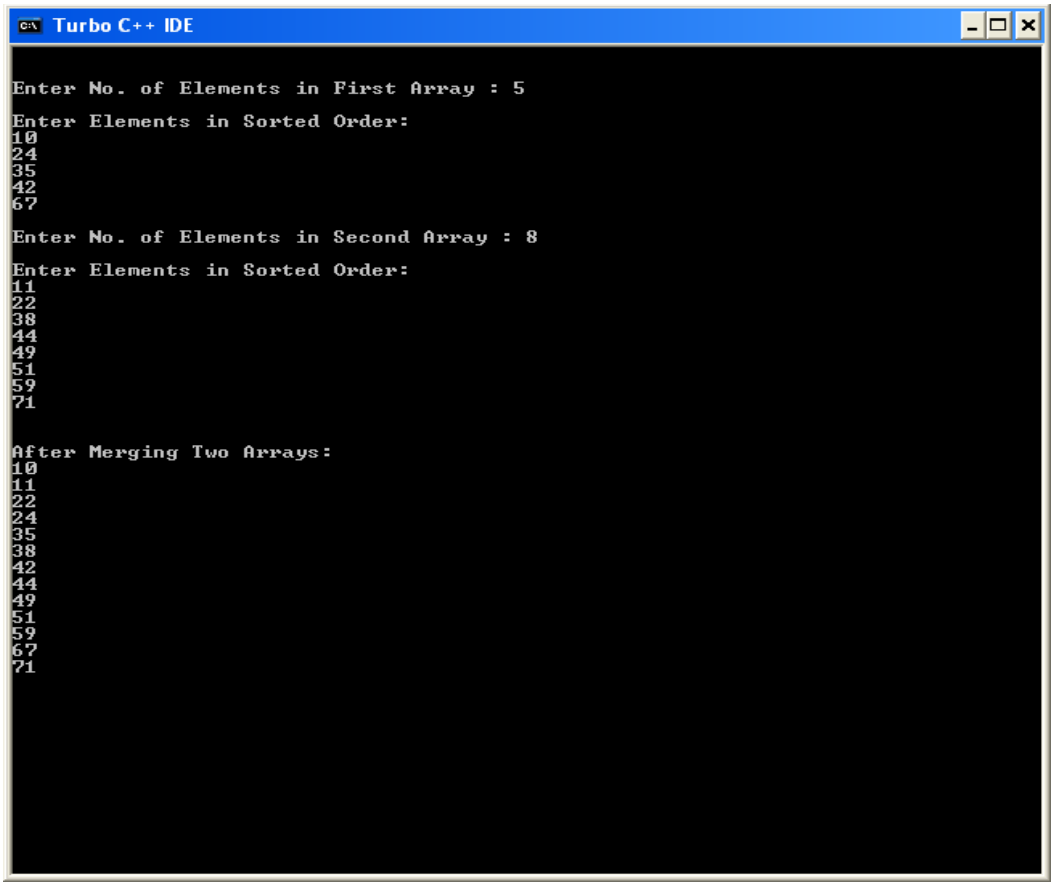
2. Write a program for Merge sort

```
//merge sort
#include<iostream.h>
#include<conio.h>
void main( )
{
clrscr( );
int a[100],b[100],c[100],n,m,i,j,k,s;
cout<<"\n\nEnter No. of Elements in First Array : ";
cin>>n;
cout<<"\nEnter Elements in Sorted Order:\n";
for(i=1;i<=n;i++)
{
cin>>a[i];
}
cout<<"\nEnter No. of Elements in Second Array : ";
cin>>m
cout<<"\nEnter Elements in Sorted Order:\n";
for(i=1;i<=m;i++)
{
cin>>b[i];
}
i=1,j=1;
for(k=1;k<=n+m;k++)
```

```
{
if(a[i]<b[j])          // Compare Elements of Array A and Array B
{
c[k]=a[i];
i++;
if(i>n)
goto b;
}
else
{
c[k]=b[j];
j++;
if(j>m)
goto a;
}
}
a:
for(s=i;s<=n;s++)       // Copy the Remaining Elements of Array A to C
{
k++;
c[k]=a[s];
}
b:
for(s=j;s<=m;s++)       // Copy the Remaining Elements of Array B to C
{
```

```
k++;
c[k]=b[s];
}
cout<<"\n\nAfter Merging Two Arrays:\n";
for(k=1;k<=n+m;k++)
  {
   cout<<c[k]<<endl;
  }
getch( );
}
```

Output of program for Merge sort (Program 2)

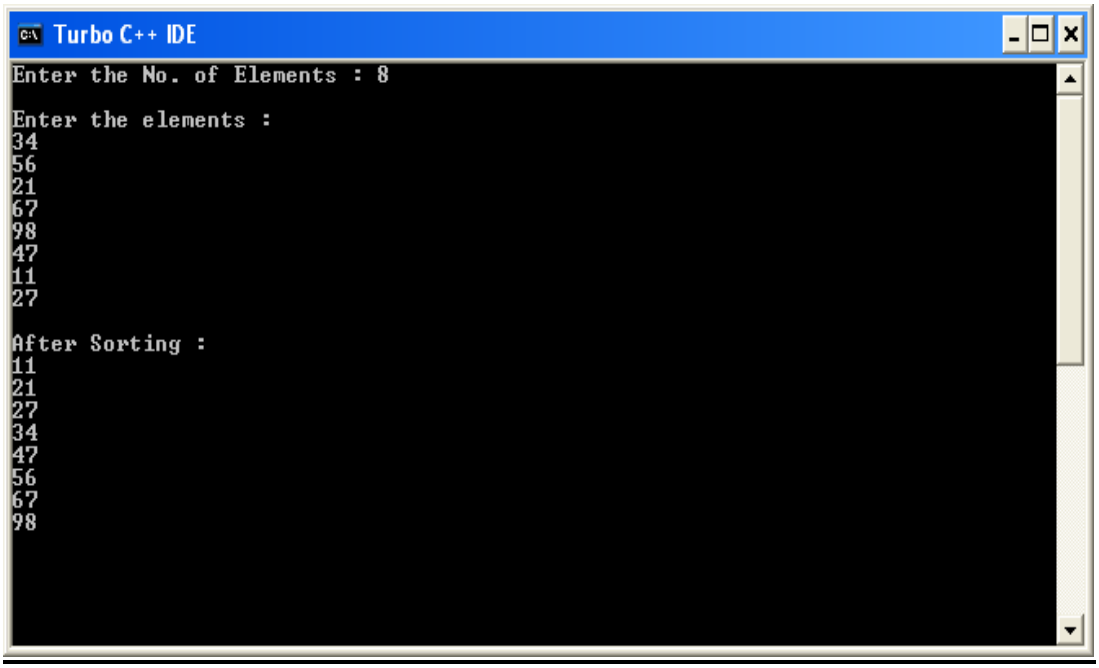


3. Write a program for Quick sort

```
//quick sort
#include<iostream.h>
#include<conio.h>
int Partition(int a[], int beg, int end)
{
 int p=beg, pivot=a[beg], loc;
 for(loc=beg+1;loc<=end;loc++)
  {
   if(pivot>a[loc])
    {
     a[p]=a[loc];
     a[loc]=a[p+1];
     a[p+1]=pivot;
     p=p+1;
     }
     }
   return p;
}
 void QuickSort(int a[], int beg, int end)
 {
 if(beg<end)
    {
     int p=Partition(a,beg,end);
    QuickSort(a,beg,p-1);
    QuickSort(a,p+1,end);
```

```
    }
  }
void main( )
 {
  clrscr( );
  int a[100],i,n,beg,end;
  cout<<"Enter the No. of Elements : ";
  cin>>n;
  cout<<"\nEnter the elements : \n";
  for(i=1;i<=n;i++)
     cin>>a[i];
  beg=1;
  end=n;
  QuickSort(a,beg,end);
  cout<<"\nAfter Sorting : \n";
  for(i=1;i<=n;i++)
     cout<<a[i]<<endl;
  getch( );
  }
```

Output of program for Quick sort (Program 3)



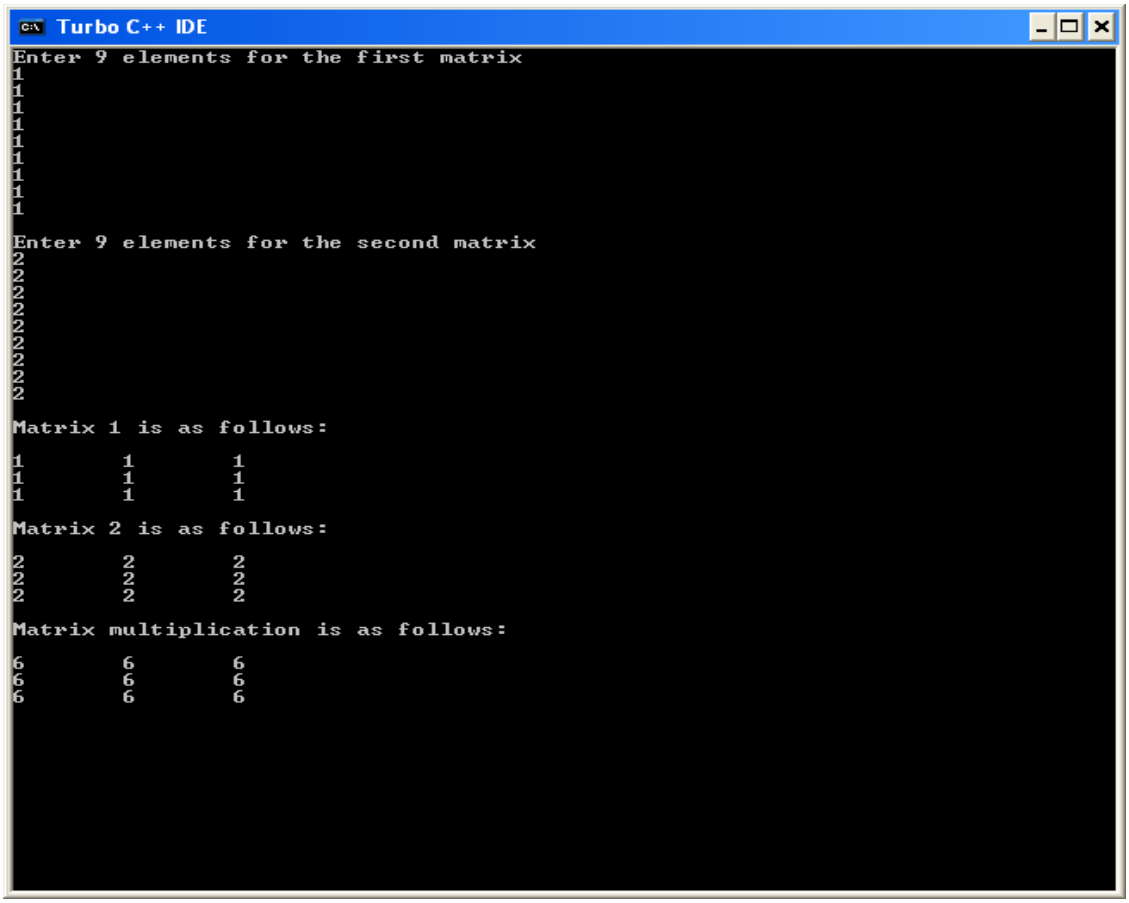
4. Write a program for Matrix multiplication

```
#include<iostream.h>
#include<conio.h>
void main( )
 {
  clrscr( );
  int a[3][3],b[3][3],c[3][3],i,j,k,s=0;
  //input of first matrix
  cout<<"Enter 9 elements for the first matrix\n";
  for(i=0;i<3;i++)
    {
    for(j=0;j<3;j++)
     cin>>a[i][j];
    }
  //input of second matrix
  cout<<"\nEnter 9 elements for the second matrix\n";
  for(i=0;i<3;i++)
   {
    for(j=0;j<3;j++)
       {cin>>b[i][j];
        }
        }
  //display first matrix
  cout<<"\nMatrix 1 is as follows:\n\n";
```

```
for(i=0;i<3;i++)
 {
 for(j=0;j<3;j++)
 {cout<<a[i][j]<<"\t";
 }
 cout<<endl;
 }
 //display Second matrix
cout<<"\nMatrix 2 is as follows:\n\n";
for(i=0;i<3;i++)
{
 for(j=0;j<3;j++)
 {cout<<b[i][j]<<"\t";
 }
 cout<<endl;
 }
 //calculate multiplication
 for(i=0;i<3;i++)
 {
 for(j=0;j<3;j++)
 {
 for(k=0;k<3;k++)
 {
 s+=a[i][k]*b[k][j];
 c[i][j]=s;
```

```
 }
 s=0;
 }
 }
//display multiplication
cout<<"\nMatrix multiplication is as follows:\n\n";
for(i=0;i<3;i++)
{
for(j=0;j<3;j++)
{cout<<c[i][j]<<"\t";
 }
 cout<<endl;
  }
  getch( );
  }
```

Output of program for matrix multiplication (Program 4)

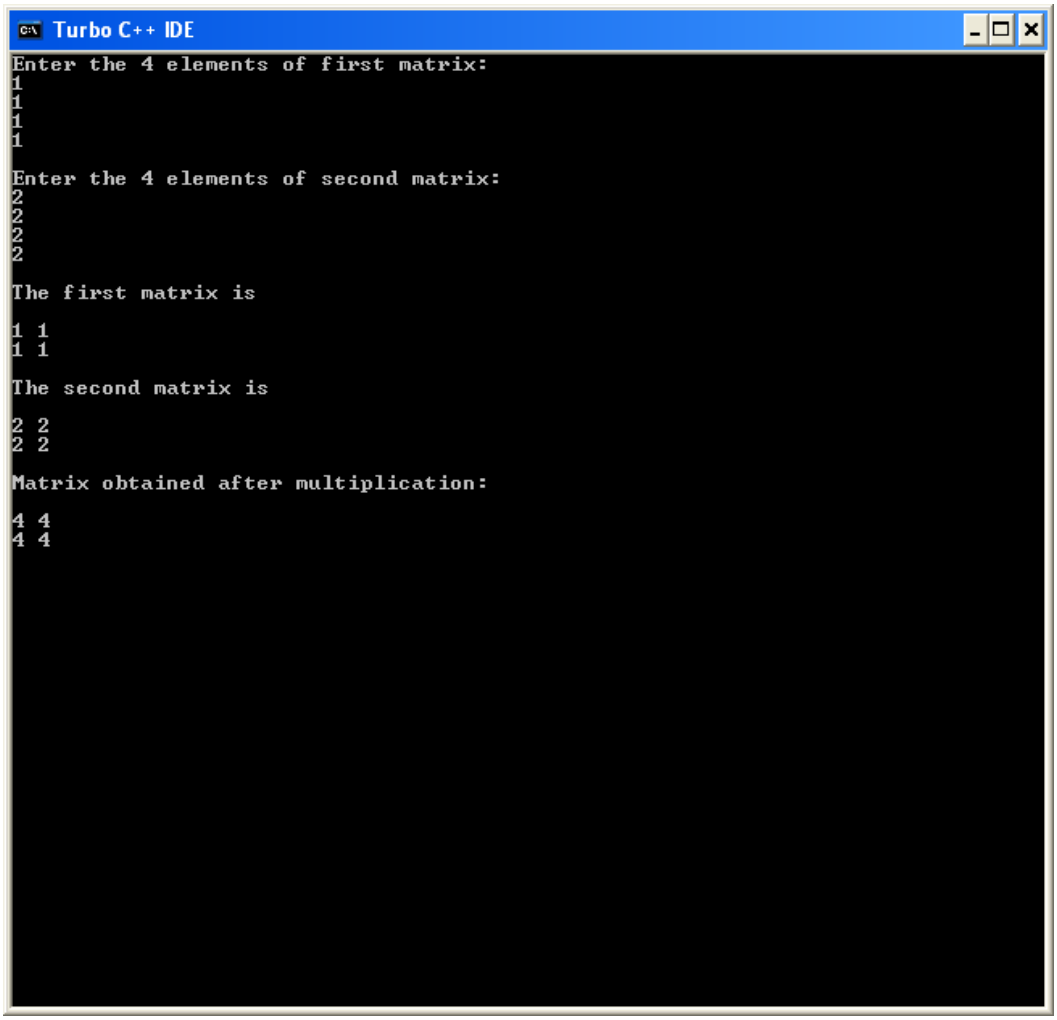


5. Write a program for Strassen's Matrix multiplication

```
#include<iostream.h>
#include<conio.h>
void main()
 {
 clrscr();
 int a[2][2],b[2][2],c[2][2],i,j;
 int m1,m2,m3,m4,m5,m6,m7;
 cout<<"Enter the 4 elements of first matrix: \n";
 for(i=0;i<2;i++)
 for(j=0;j<2;j++)
 cin>>a[i][j];
 cout<<"\nEnter the 4 elements of second matrix: \n";
 for(i=0;i<2;i++)
 for(j=0;j<2;j++)
  cin>>b[i][j];
 cout<<"\nThe first matrix is\n\n";
 for(i=0;i<2;i++)
 {
  for(j=0;j<2;j++)
  cout<<a[i][j]<<" ";
   cout<<endl;
   }
 cout<<"\nThe second matrix is\n\n";
 for(i=0;i<2;i++)
```

```
{
for(j=0;j<2;j++)
cout<<b[i][j]<<" ";
cout<<endl;
}
m1= (a[0][0] + a[1][1])*(b[0][0]+b[1][1]);
m2= (a[1][0]+a[1][1])*b[0][0];
m3= a[0][0]*(b[0][1]-b[1][1]);
m4= a[1][1]*(b[1][0]-b[0][0]);
m5= (a[0][0]+a[0][1])*b[1][1];
m6= (a[1][0]-a[0][0])*(b[0][0]+b[0][1]);
m7= (a[0][1]-a[1][1])*(b[1][0]+b[1][1]);
c[0][0]=m1+m4-m5+m7;
c[0][1]=m3+m5;
c[1][0]=m2+m4;
c[1][1]=m1-m2+m3+m6;
cout<<"\nMatrix obtained after multiplication: \n\n";
for(i=0;i<2;i++)
{
for(j=0;j<2;j++)
cout<<c[i][j]<<" ";
cout<<endl;
}
getch();
}
```

Output of program for Strassen's matrix multiplication (Program 5)



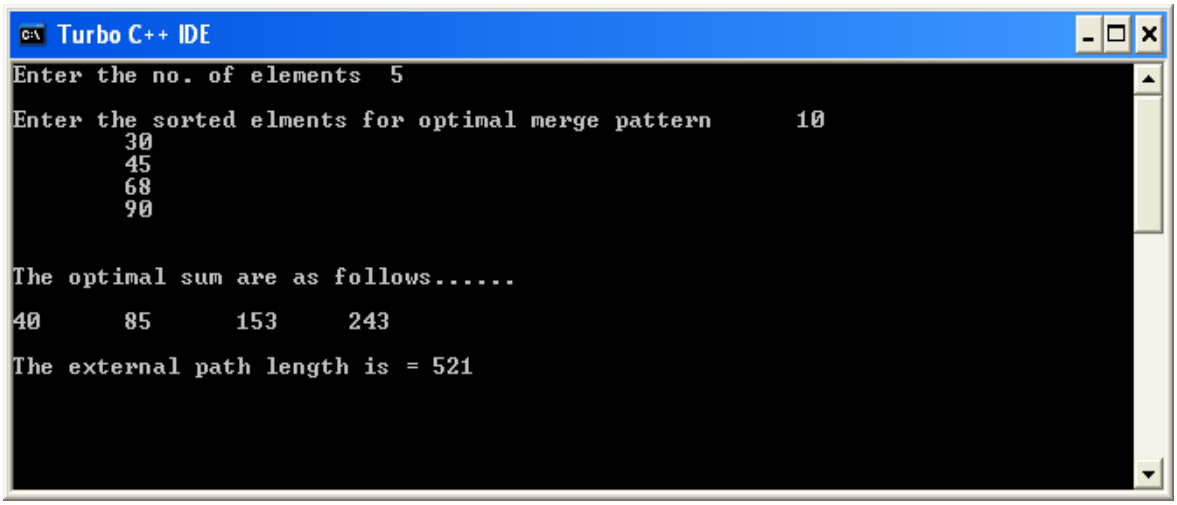
6. Write a program for optimal merge patterns

```
#include<iostream.h>
#include<conio.h>
 void main( )
{
clrscr( );
int i,k,a[10],c[10],n,l;
cout<<"Enter the no. of elements\t";
cin>>n;
cout<<"\nEnter the sorted elments for optimal merge pattern";
 for(i=0;i<n;i++)
{
 cout<<"\t";
cin>>a[i];
 }
i=0;
k=0;
 c[k]=a[i]+a[i+1];
i=2;
while(i<n)
{
k++;
 if((c[k-1]+a[i])<=(a[i]+a[i+1]))
```

```
{
c[k]=c[k-1]+a[i];
}
else
{
c[k]=a[i]+a[i+1];
 i=i+2;
while(i<n)
{
 k++;
if((c[k-1]+a[i])<=(c[k-2]+a[i]))
 {
c[k]=c[k-1]+a[i];
 }
else
 {
c[k]=c[k-2]+a[i];
 }
i++;
 }
}
i++;
 }
k++;
c[k]=c[k-1]+c[k-2];
```

```
 cout<<"\n\nThe optimal sum are as follows.....\n\n";
for(k=0;k<n-1;k++)
{
cout<<c[k]<<"\t";
}
 l=0;
for(k=0;k<n-1;k++)
 {
l=l+c[k];
}
cout<<"\n\nThe external path length is = "<<l;
 getch( );
 }
```

Output of program for optimal merge patterns (Program 6)



7. Write a program for Huffman coding

```
#include<stdio.h>
#include<string.h>
#include<conio.h>
#include<stdlib.h>
#define MAX 10
struct link
{
 int freq;
 char ch[MAX];
 struct link* right;
 struct link* left;
};
typedef struct link node;
void sort(node *[], int);
node* create(char[], int);
void sright(node *[], int);
void Assign_Code(node*, int [], int);
void Delete_Tree(node *);

main()
{
 node* ptr, * head;
 int i, n, total = 0, u, c[15];
 char str[MAX];
 node* a[12];
```

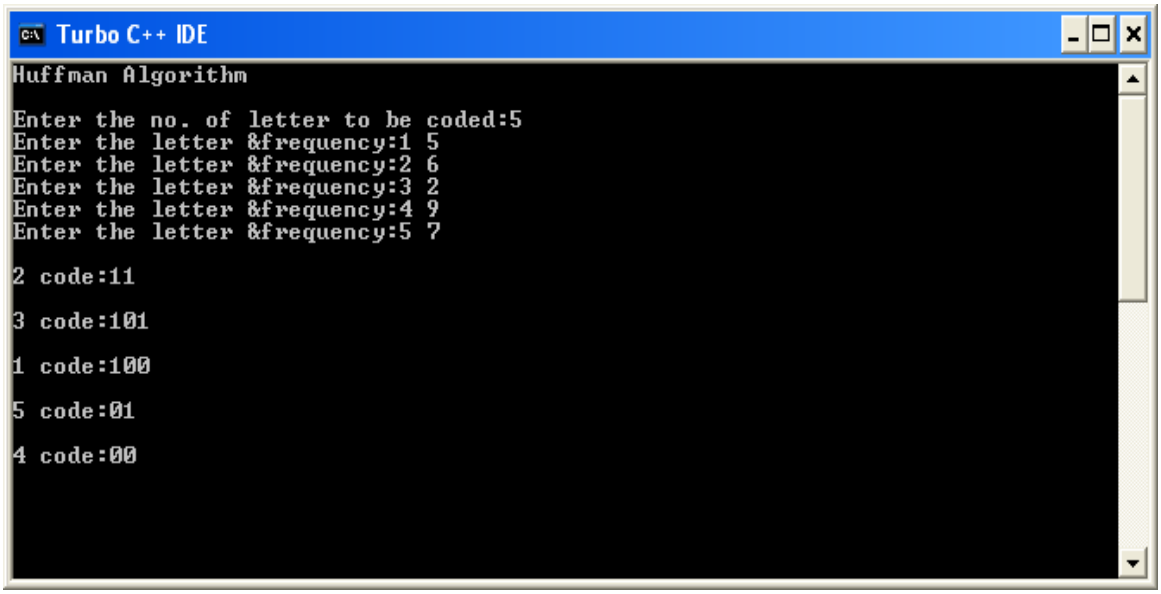
```
int freq;
clrscr( );
cout<<"Huffman Algorithm\n";
cout<<"\nEnter the no. of letter to be coded:";
cin>>n;
for (i = 0; i < n; i++)
{
 cout<<"Enter the letter &frequency:";
 cin>> str>>freq;
 a[i] = create(str, freq);
}
while (n > 1)
{
 sort(a, n);
  u = a[0]->freq + a[1]->freq;
  strcpy(str,a[0]->ch);
  strcat(str,a[1]->ch);
  ptr = create(str, u);
  ptr->right = a[1];
  ptr->left = a[0];
   a[0] = ptr;
    sright(a, n);
    n--;
   }
    Assign_Code(a[0], c, 0);
    getch();
            Delete_Tree(a[0]);
```

```
}
node* create(char a[], int x)
{
node* ptr;
ptr = (node *) malloc(sizeof(node));
ptr->freq = x;
strcpy( ptr->ch , a);
ptr->right = ptr->left = NULL;
return(ptr);
}
void sort(node* a[], int n)
{
int i, j;
node* temp;
for (i = 0; i < n - 1; i++)
for (j = i; j < n; j++)
if (a[i]->freq > a[j]->freq)
        {
temp = a[i];
a[i] = a[j];
a[j] = temp;
}
}
void sright(node* a[], int n)
{
int i;
for (i = 1; i < n - 1; i++)
```

```
a[i] = a[i + 1];
}
void Assign_Code(node* tree, int c[], int n)
{
 int i;
  if ((tree->left == NULL) && (tree->right == NULL))
  {
    cout<<"\n"<<tree->ch<<" code:";
    for (i = 0; i < n; i++)
    {
    cout<<c[i];
    }
    cout<<"\n";
    }
   else
    {
     c[n] = 1;
     n++;
     Assign_Code(tree->left, c, n);
     c[n - 1] = 0;
      Assign_Code(tree->right, c, n);
    }
     }
   void Delete_Tree(node * root)
   {
   if(root!=NULL)
   {
```

```
    Delete_Tree(root->left);
    Delete_Tree(root->right);
    free(root);
  }
}
```

Output of program for Huffman coding (Program 7)



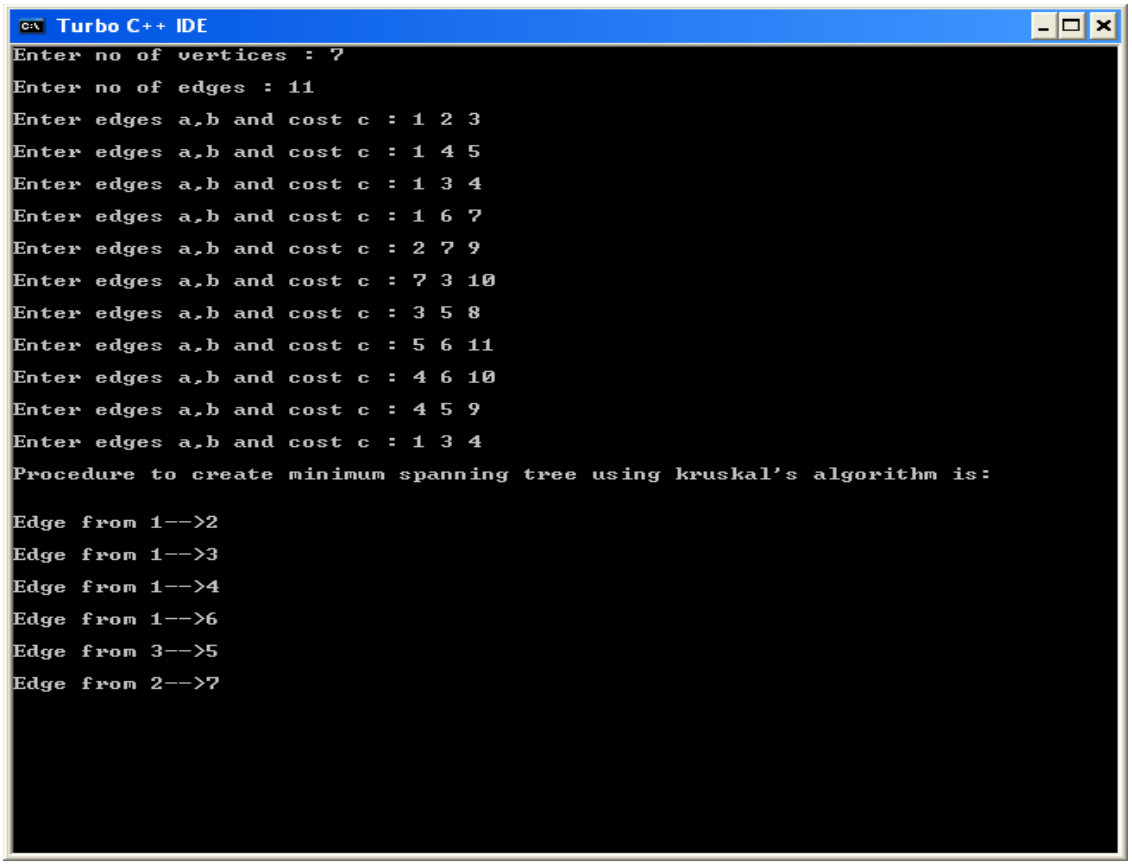
8. Write a program for minimum spanning trees using kruskal's algorithm

```
#include<iostream.h>
#include<conio.h>
int cost[10][10],i,j,k,n,m,c,visit,visited[10],l,v,count,count1,vst,p;
void main( )
 {  clrscr( );
 int dup1,dup2;
 cout<<"Enter no of vertices : ";
 cin>>n;
 cout<<"\nEnter no of edges : ";
 cin>>m;
 for(k=1;k<=m;k++)
 {  cout <<"\nEnter edges a,b and cost c : ";
  cin>>i>>j>>c;
  cost[i][j]=c;
  cost[j][i]=c;
 }
 cout<<"\nProcedure to create minimum spanning tree using kruskal's  algorithm is: \n";
 for(i=1;i<=n;i++)
 for(j=1;j<=n;j++)
 if(cost[i][j]==0)
cost[i][j]=31999;
 visit=1;
 while(visit<n)
 {
```

```
 v=31999;
 for(i=1;i<=n;i++)
for(j=1;j<=n;j++)
 if(cost[i][j]!=31999 && cost[i][j]<v && cost[i][j]!=-1 )
 {  int count =0;
 for(p=1;p<=n;p++)
 {  if(visited[p]==i || visited[p]==j)
  count++;
 }
if(count >= 2)
 {
 for(p=1;p<=n;p++)
 if(cost[i][p]!=31999 && p!=
  dup1=p;
 for(p=1;p<=n;p++)
  if(cost[j][p]!=31999 && p!=i)
 dup2=p;
 if(cost[dup1][dup2]==-1)
 continue;
 }
 l=i;
 k=j;
v=cost[i][j];
}
 cout <<"\n\nEdge from " <<l <<"-->"<<k;
cost[l][k]=-1;
cost[k][l]=-1;
```

```
visit++;
int count=0;
count1=0;
for(i=1;i<=n;i++)
  {
if(visited[i]==l)
count++;
if(visited[i]==k)
count1++;
}
 if(count==0)
 visited[++vst]=l;
  if(count1==0)
 visited[++vst]=k;
 }
getch( );
}
```

Output of program for Kruskal's algorithm (Program 8)



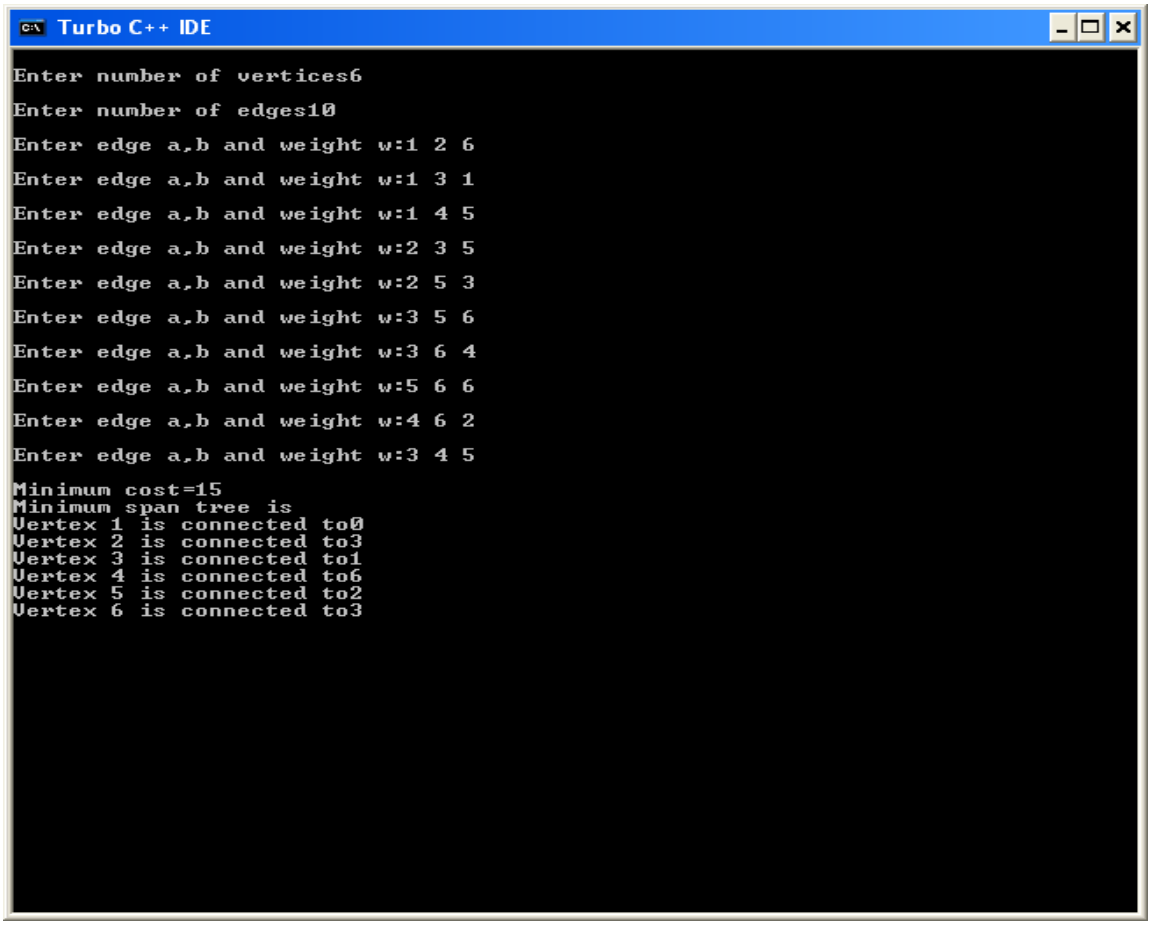
9. Write a program for minimum spanning trees using prim's algorithm

```
#include<iostream.h>
#include<conio.h>
int weight[20][20],visited[20],d[20],p[20];
int v,e;
void creategraph( )
{
int i,j,a,b,w;
cout<<"\nEnter number of vertices";
cin>>v;
cout<<"\nEnter number of edges";
cin>>e;
for(i=1;i<=v;i++)
for(j=1;j<=v;j++)
weight[i][j]=0;
for(i=1;i<=v;i++)
{  p[i]=visited[i]=0;
d[i]=32767;
}
for(i=1;i<=e;i++)
{
cout<<"\nEnter edge a,b and weight w:";
cin>>a>>b>>w;
 weight[a][b]=weight[b][a]=w;
 }
}
```

```
void prim( )
 {
 int current,totalvisited,mincost,i;
 current=1;d[current]=0;
 totalvisited=1;
 visited[current]=1;
 while(totalvisited!=v)
 {  for(i=1;i<=v;i++)
  {
  if(weight[current][i]!=0)
  if(visited[i]==0)
  if(d[i]>weight[current][i])
  {
d[i]=weight[current][i];
p[i]=current;
   }
  }
 mincost=32767;
for(i=1;i<=v;i++)
 {
if(visited[i]==0)
if(d[i]<mincost)
{
  mincost=d[i];
  current=i;
 }
 }
```

```
 visited[current]=1;
 totalvisited++;
}
mincost=0;
for(i=1;i<=v;i++)
   mincost+=d[i];
 cout<<"\nMinimum cost="<<mincost;
cout<<"\nMinimum span tree is";
for(i=1;i<=v;i++)
  cout<<"\nVertex "<<i<<" is connected to"<<p[i];
}
void main( )
{ clrscr( );
creategraph( );
prim( );
getch( );
}
```

Output of program for prim's algorithm (Program 9)

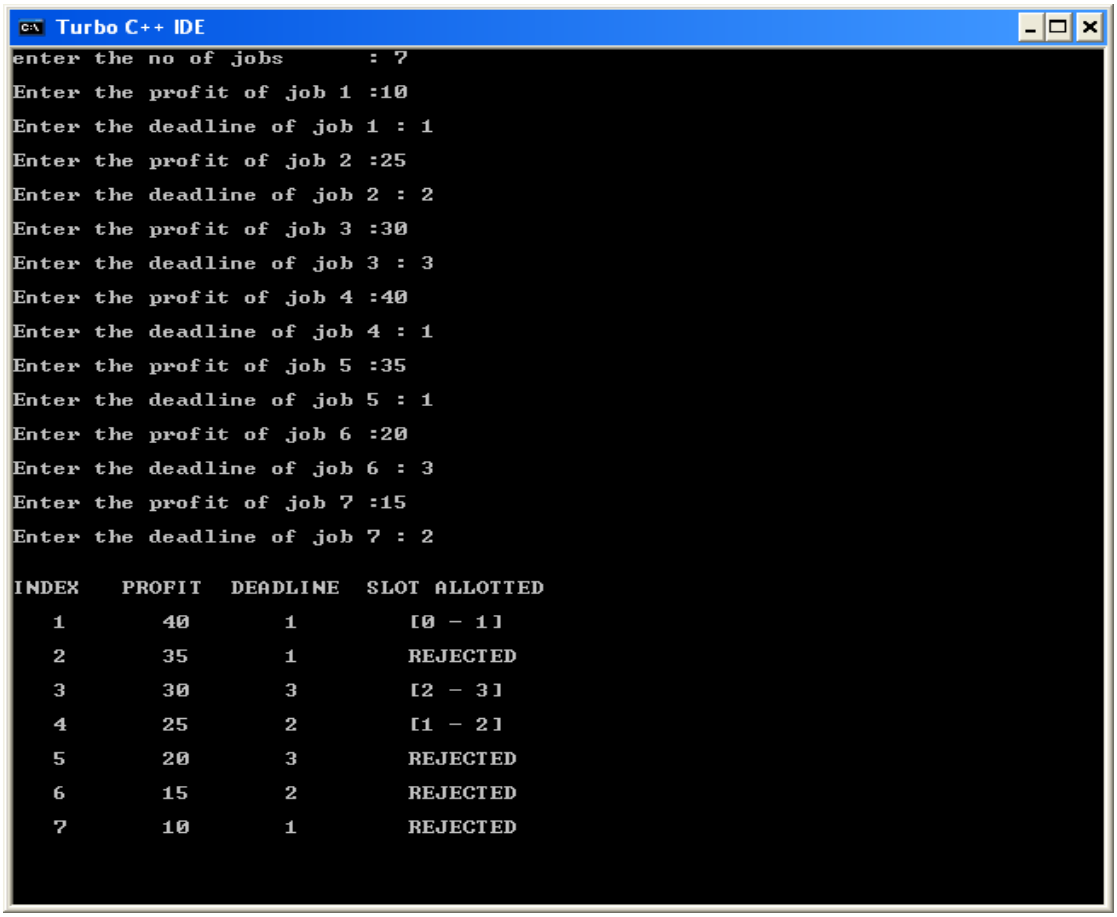


10.Write a program for job sequencing problem with deadline

```
 #include<conio.h>
#include<iostream.h>
int n,i,j,k,t;
int check(int s[],int p)
{  int ptr=0;
for(int i=0;i<n;i++)
{if(s[i]==p)
ptr++;
}
 if(ptr==0)
  return 1;
 else
  return 0;
 }
int main()
{
clrscr();
 cout<<"enter the no of jobs      : ";
cin>>n;
int slot[10],profit,p[10],d[10];
 for(i=0;i<n;i++)
 {
 cout<<"\nEnter the profit of job "<<i+1<<" :";
```

```
  cin>>p[i];
  cout<<"\nEnter the deadline of job "<<i+1<<" : ";
  cin>>d[i];
  }

 for(i=0;i<n;i++)
for(j=i+1;j<n;j++)
if(p[i]<p[j])
{ t=p[i];
  p[i]=p[j];
  p[j]=t;
  t=d[i];
  d[i]=d[j];
  d[j]=t;
}

 for(i=0;i<n;i++)
slot[i]=0;
for(i=0;i<n;i++)
for(j=d[i];j>0;j--)
{if(check(slot,j)==1)
{slot[i]=j;
 break;}
  }
```

```
    cout<<"\n\nINDEX   PROFIT  DEADLINE  SLOT ALLOTTED ";

    for(i=0;i<n;i++)

    {

    if(slot[i]>0)

    cout<<"\n\n  "<<i+1<<"      "<<p[i]<<"      "<<d[i]<<"      ["<<slot[i]-1<<" -
"<<slot[i]<<"]";

    else

    cout<<"\n\n  "<<i+1<<"      "<<p[i]<<"      "<<d[i]<<"      REJECTED";

    }

     getch();

  }
```


Turbo C++ IDE
enter the no of jobs : 7
Enter the profit of job 1 :10
Enter the deadline of job 1 : 1
Enter the profit of job 2 :25
Enter the deadline of job 2 : 2
Enter the profit of job 3 :30
Enter the deadline of job 3 : 3
Enter the profit of job 4 :40
Enter the deadline of job 4 : 1
Enter the profit of job 5 :35
Enter the deadline of job 5 : 1
Enter the profit of job 6 :20
Enter the deadline of job 6 : 3
Enter the profit of job 7 :15
Enter the deadline of job 7 : 2
INDEX PROFIT DEADLINE SLOT ALLOTTED
1 40 1 [0 - 1]
2 35 1 REJECTED
3 30 3 [2 - 3]
4 25 2 [1 - 2]
5 20 3 REJECTED
6 15 2 REJECTED
7 10 1 REJECTED