Practical File

Subject: Design and Analysis of Algorithm

CSB 252

Name- Sumanjay

Roll- 161210040

CSE 2nd Year

Department of Computer Science Engineering

National Institute of Technology Delhi

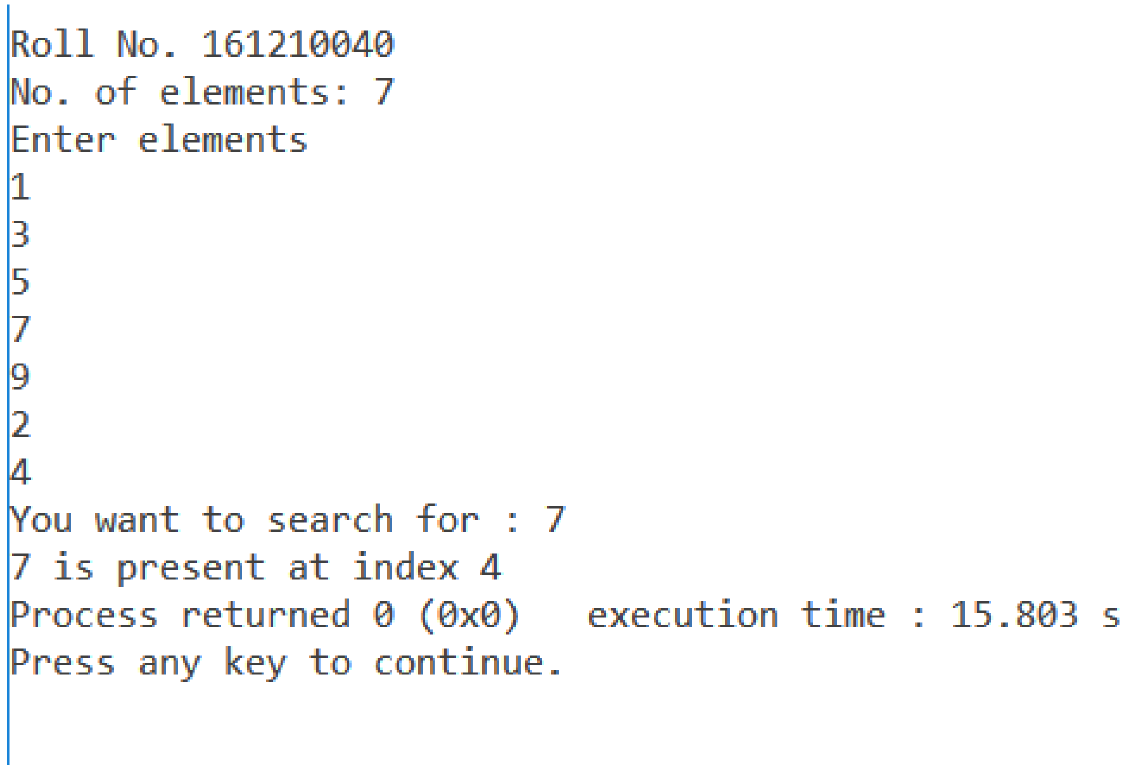
Index

Linear Search

//Source Code

```
#include<iostream>

using namespace std;

int LS(int ar[],int size,int val)

{

    int i;

    for(i=0;i<size;i++){

        if(ar[i]==val)

            return i;

    }

  return -1;

}

int main()

{

   int a[50],n,i,val,index;

   cout<<"No. of elements: ";

   cin>>n;

   cout<<"Enter elements\n";

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

      cin>>a[i];
  cout<<"No. you want to search for : ";

  cin>>val;

  index=LS(a,n,val);

  if(index==-1)

    cout<<"Not found!!";

  else

    cout<<val<<" is present at index "<<index+1;

  return 0;

}
```

<u>**Output**</u>



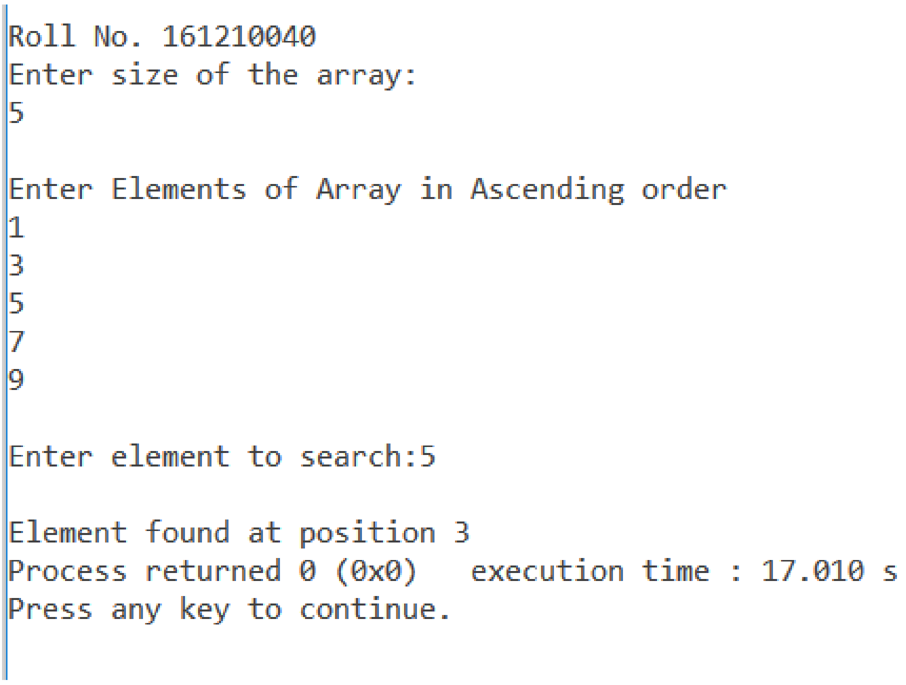
```
Roll No. 161210040
No. of elements: 7
Enter elements
1
3
5
7
9
2
4
You want to search for : 7
7 is present at index 4
Process returned 0 (0x0)   execution time : 15.803 s
Press any key to continue.
```

Binary Search

//Source Code

```
#include<iostream>

using namespace std;

int main() {

int search(int [],int,int);

int n,i,a[100],e,res;

cout<<"Roll Number. 161210040"<<endl;

cout<<"Enter size of the array:"<<endl;

cin>>n;

cout<<"\nEnter Elements of Array in Ascending order\n";

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

            cin>>a[i];

}

cout<<"\nEnter element to search:";    cin>>e;

res=search(a,n,e);

if(res!=-1)

  cout<<"\nElement found at position "<<res+1;

else

  cout<<"\nElement is not found....!!!";

return 0;

}

int search(int a[],int n,int e)

{

int f,l,m;
f=0;

l=n-1;

while(f<=l)    {

            m=(f+l)/2;

            if(e==a[m])

                        return(m);

            else if (e>a[m])

                        f=m+1;

            else

                        l=m-1;

}

   return -1;

}
```

Output



```
Roll No. 161210040
Enter size of the array:
5

Enter Elements of Array in Ascending order
1
3
5
7
9

Enter element to search:5

Element found at position 3
Process returned 0 (0x0)   execution time : 17.010 s
Press any key to continue.
```

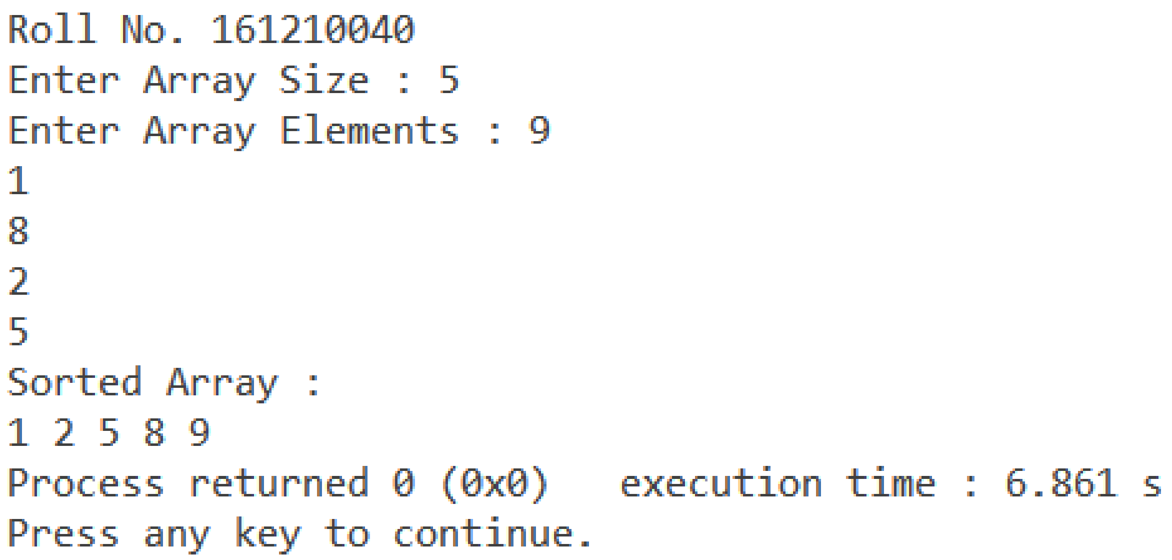
Insertion Sort

//Source Code

```
#include<iostream>

using namespace std;

int  main()

{

   int size, arr[50], i, j, temp;

   cout<<"Enter Array Size : ";

   cin>>size;

   cout<<"Enter Array Elements : ";

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

   {

      cin>>arr[i];

   }

   cout<<"Sorting array\n";

   for(i=1; i<size; i++)

   {

      temp=arr[i];

      j=i-1;

      while((temp<arr[j]) && (j>=0))

      {

         arr[j+1]=arr[j];

         j=j-1;

      }
```

```
        arr[j+1]=temp;
    }
    cout<<"Sorted Array : \n";
    for(i=0; i<size; i++)
    {
        cout<<arr[i]<<" ";
    }
    return 0;
}
```

Output



```
Roll No. 161210040
Enter Array Size : 5
Enter Array Elements : 9
1
8
2
5
Sorted Array :
1 2 5 8 9
Process returned 0 (0x0)   execution time : 6.861 s
Press any key to continue.
```

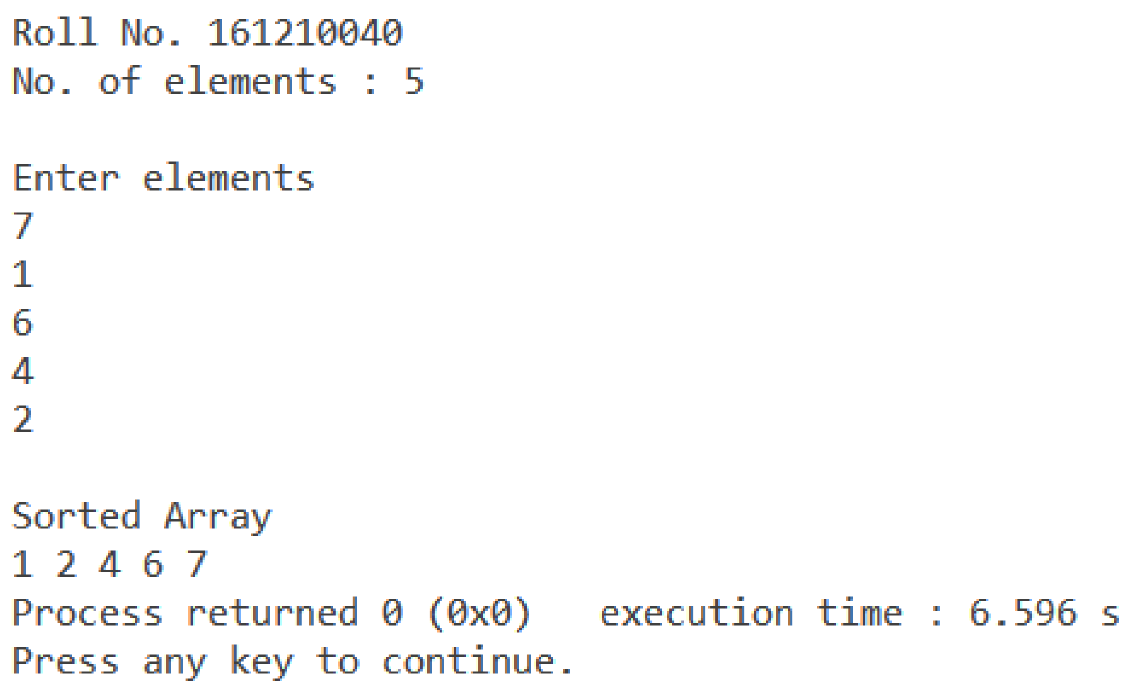
Quick Sort

//Source Code

```
#include <iostream>

using namespace std;

void quick_sort(int[],int,int);

int partition(int[],int,int);

int main()

{

   int a[50],n,i;

   cout<<"Roll No. 161210040";

   cout<<"\nNo. of elements : ";

   cin>>n;

   cout<<"\nEnter elements\n";


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

      cin>>a[i];


   quick_sort(a,0,n-1);

   cout<<"\nSorted Array\n";


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

      cout<<a[i]<<" ";

   return 0;

}
```

```
void quick_sort(int a[],int l,int u)
{
   int j;
   if(l<u)
   {
      j=partition(a,l,u);
      quick_sort(a,l,j-1);
      quick_sort(a,j+1,u);
   }
}

int partition(int a[],int l,int u)
{
   int v,i,j,temp;
   v=a[l];
   i=l;
   j=u+1;
   do
   {  do
         i++;
      while(a[i]<v&&i<=u);
      do
         j--;
      while(v<a[j]);
      if(i<j)
      {
        temp=a[i];
        a[i]=a[j];
        a[j]=temp;
      }
   }
while(i<j);
   a[l]=a[j];
   a[j]=v;
   return(j);
}
```

Output



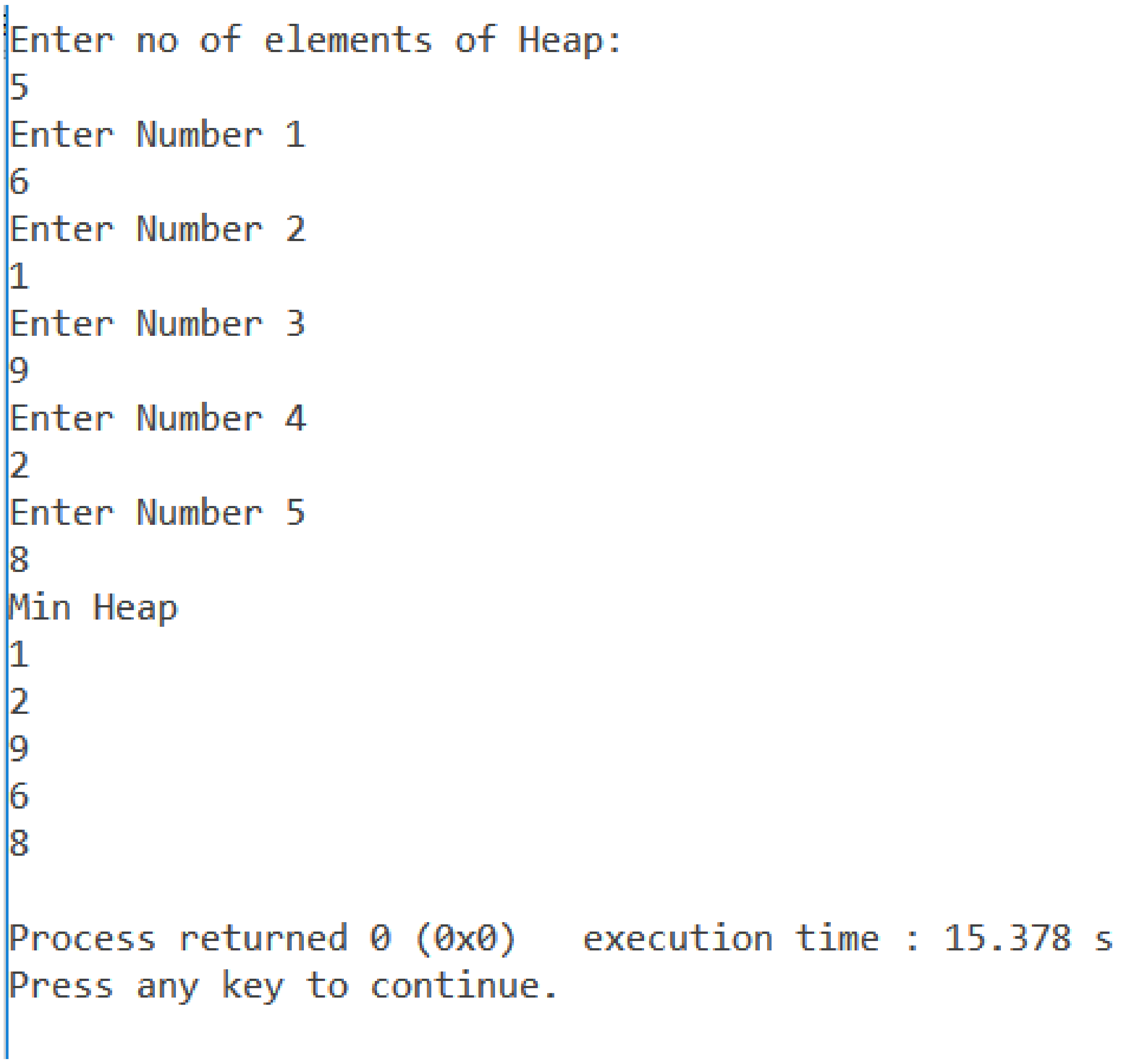
```
Roll No. 161210040
No. of elements : 5

Enter elements
7
1
6
4
2

Sorted Array
1 2 4 6 7
Process returned 0 (0x0)   execution time : 6.596 s
Press any key to continue.
```

Min Heap

//Source Code

```
#include <iostream>

using namespace std;

void min_heapify(int *a,int i,int n)

{

   int j, temp;

   temp = a[i];

   j = 2 * i;

   while (j <= n)

   {

      if (j < n && a[j+1] < a[j])

         j = j + 1;

      if (temp < a[j])         break;

      else if (temp >= a[j])

      {

         a[j/2] = a[j];

         j = 2 * j;

      }

   }

   a[j/2] = temp;

   return;

}

void build_minheap(int *a, int n)

{
   int i;

   for(i = n/2; i >= 1; i--)

   {

      min_heapify(a,i,n);

   }

}

int main()

{

   int n, i, x;

   cout<<"Enter no of elements of Heap:\n";

   cin>>n;

   int a[20];

   for (i = 1; i <= n; i++)

   {

      cout<<"Enter Number "<<(i)<<endl;

      cin>>a[i];

   }

   build_minheap(a, n);

   cout<<"Min Heap\n";

   for (i = 1; i <= n; i++)

   {

      cout<<a[i]<<endl;

   }

   return 0;

}
```

Output



```
Enter no of elements of Heap:
5
Enter Number 1
6
Enter Number 2
1
Enter Number 3
9
Enter Number 4
2
Enter Number 5
8
Min Heap
1
2
9
6
8

Process returned 0 (0x0)   execution time : 15.378 s
Press any key to continue.
```

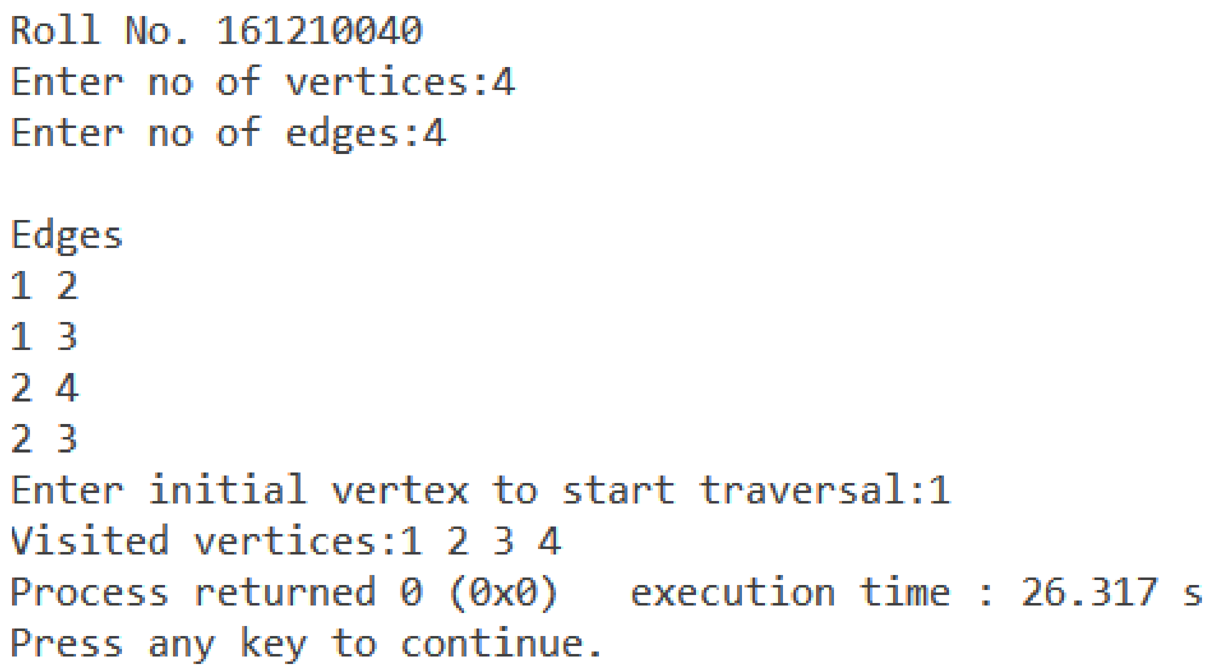
Breadth First Search

//Source Code

```
#include<iostream>

using namespace std;

int cost[10][10],i,j,k,n,qu[10],front,rare,v,visit[10],visited[10];

int main()

{

   int m;

   cout<<"Roll No. 161210040\n";

   cout <<"Enter no of vertices:";

   cin >> n;

   cout <<"Enter no of edges:";

   cin >> m;

   cout <<"\nEdges \n";

   for(k=1; k<=m; k++)

   {

      cin >>i>>j;

      cost[i][j]=1;

   }

   cout <<"Enter initial vertex to start traversal:";

   cin >>v;

   cout <<"Visited vertices:";

   cout <<v<<" ";

   visited[v]=1;

   k=1;
```

```
    while(k<n)

    {

        for(j=1; j<=n; j++)

            if(cost[v][j]!=0 && visited[j]!=1 && visit[j]!=1)

            {

                visit[j]=1;

                qu[rare++]=j;

            }

        v=qu[front++];

        cout<<v <<" ";

        k++;

        visit[v]=0;

        visited[v]=1;

    }

    return 0;

}
```

Output



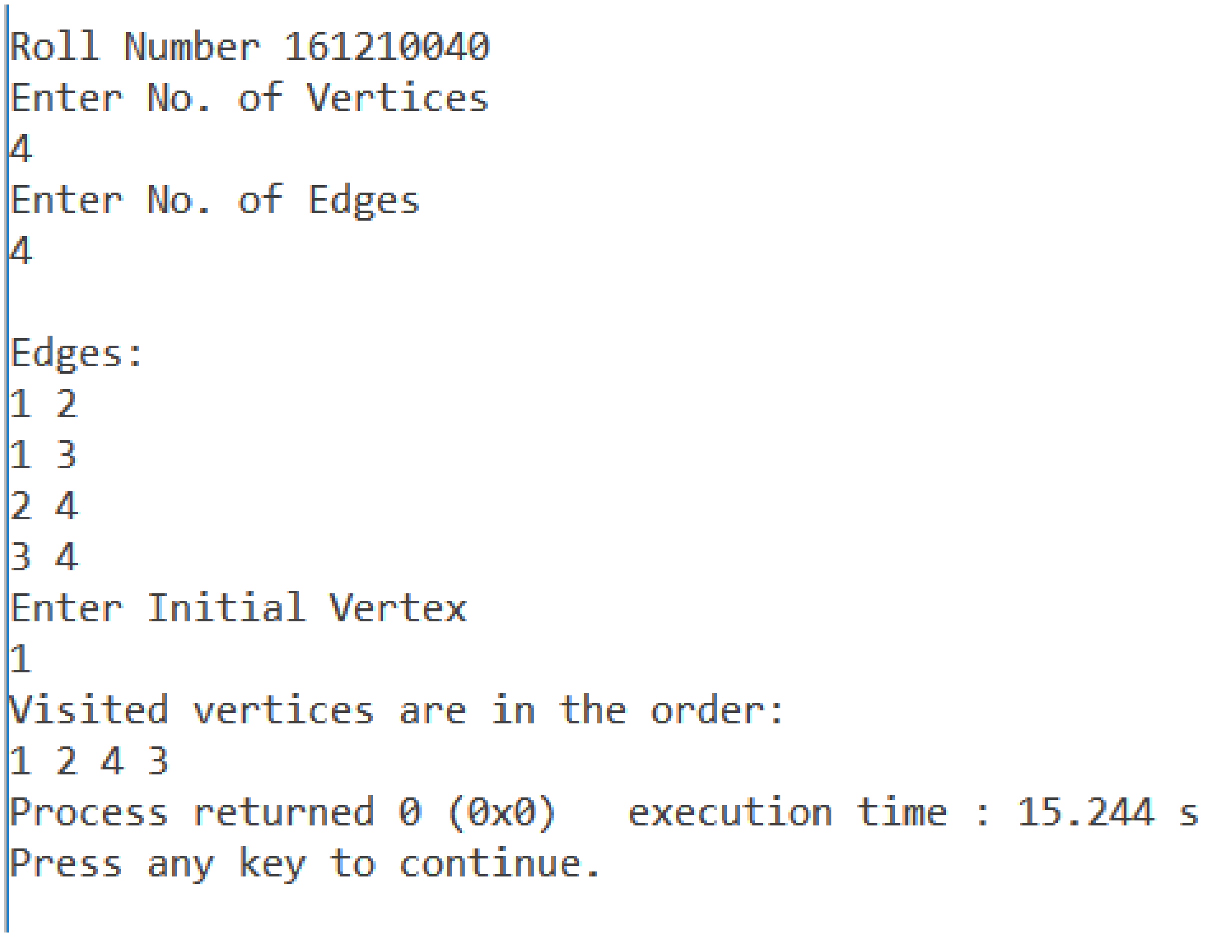
```
Roll No. 161210040
Enter no of vertices:4
Enter no of edges:4

Edges
1 2
1 3
2 4
2 3
Enter initial vertex to start traversal:1
Visited vertices:1 2 3 4
Process returned 0 (0x0)   execution time : 26.317 s
Press any key to continue.
```

Depth First Search

//Source Code

```
#include<iostream>

using namespace std;

int cost[10][10],i,j,k,n,stk[10],top,v,visit[10],visited[10];

int main()

{

    int m;

    cout<<"Roll Number 161210040\n";

    cout <<"Enter No. of Vertices";

    cin >> n;

    cout <<"Enter No. of Edges";

    cin >> m;

    cout <<"\nEdges: \n";

    for(k=1; k<=m; k++)

    {

        cin >>i>>j;

        cost[i][j]=1;

    }

    cout <<"Enter Initial Vertex";

    cin >>v;

    cout <<"Visited vertices are in the order: "<<endl;

    cout << v <<" ";
    visited[v]=1;

    k=1;

    while(k<n)

    {

        for(j=n; j>=1; j--) if(cost[v][j]!=0 && visited[j]!=1 && visit[j]!=1)

            {

                visit[j]=1;

                stk[top]=j;

                top++;

            }

        v=stk[--top];

        cout<<v << " ";

        k++;

        visit[v]=0;

        visited[v]=1;

    }

    return 0;

}
```

Output


```
Roll Number 161210040
Enter No. of Vertices
4
Enter No. of Edges
4

Edges:
1 2
1 3
2 4
3 4
Enter Initial Vertex
1
Visited vertices are in the order:
1 2 4 3
Process returned 0 (0x0)   execution time : 15.244 s
Press any key to continue.
```

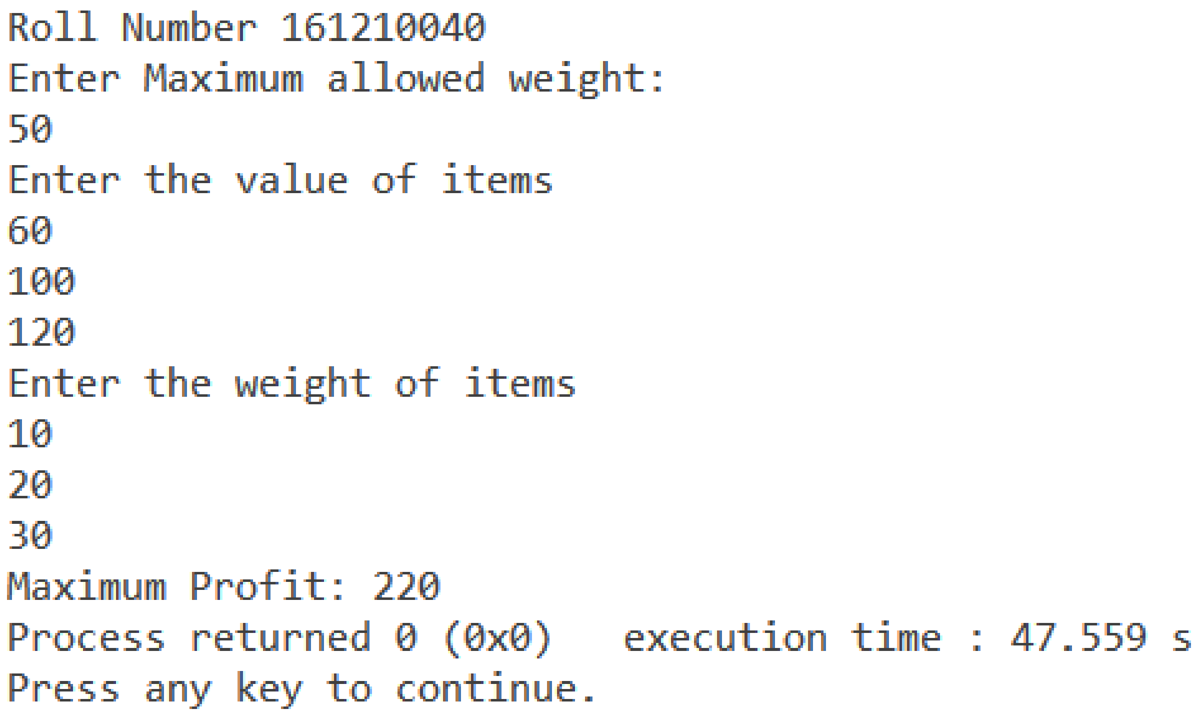
Fractional Knapsack

//Source Code

```
#include<iostream>

using namespace std;

int max(int a, int b)

{

   return (a > b)? a : b;

}


int knapSack(int W, int wt[], int val[], int n)

{

   if (n == 0 || W == 0)

       return 0;


   else if (wt[n-1] > W)

       return knapSack(W, wt, val, n-1);


   else

       return max( val[n-1] + knapSack(W-wt[n-1], wt, val, n-1),knapSack(W, wt, val, n-1));

}

int main()

{

   int W;

   int val[3],wt[3];
```

```
    cout<<"Roll Number 161210040\n";

    cout<<"Enter Maximum allowed weight: \n";

    cin>>W;


    cout<<"Enter the value of items\n";

    for (int i=0; i<3; i++)

    {

        cin>>val[i];

    }

    cout<<"Enter the weight of items\n";

    for (int i=0; i<3; i++)

    {

        cin>>wt[i];

    }


    int n = sizeof(val)/sizeof(val[0]);

    cout<< "Maximum Profit: " <<knapSack(W, wt, val, n);


}
```

Output



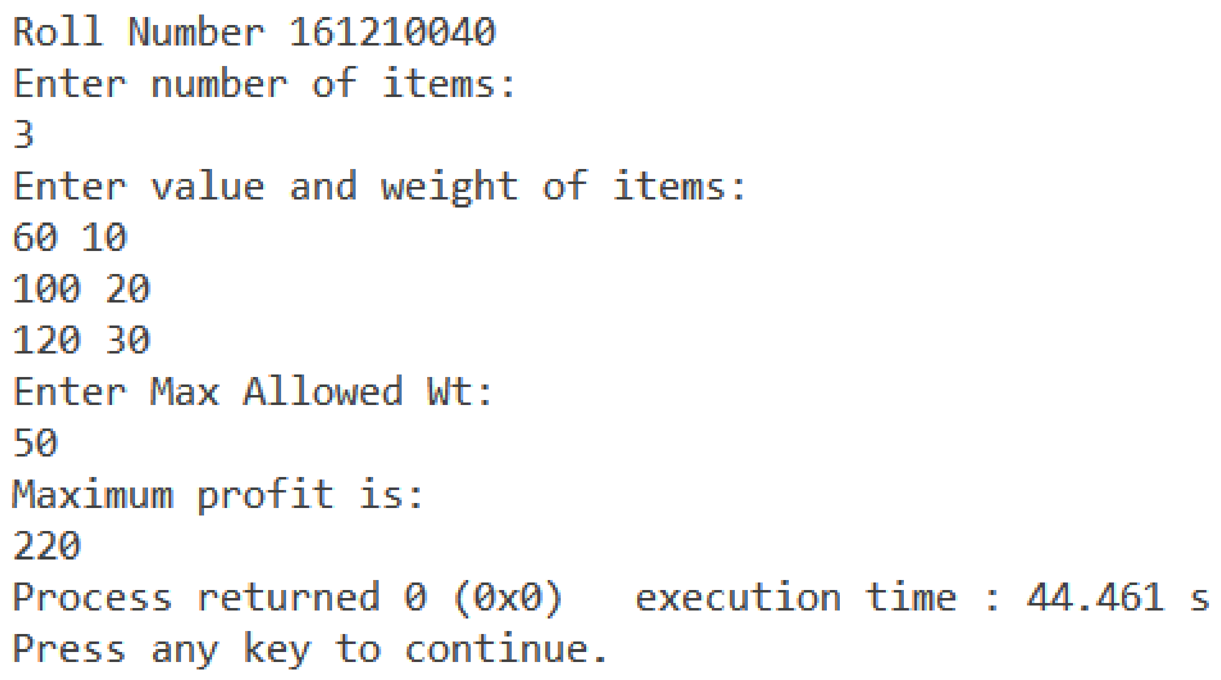
```
Roll Number 161210040
Enter Maximum allowed weight:
50
Enter the value of items
60
100
120
Enter the weight of items
10
20
30
Maximum Profit: 220
Process returned 0 (0x0)   execution time : 47.559 s
Press any key to continue.
```

0/1 Knapsack

//Source Code

```
#include<iostream>

using namespace std;

int max(int a, int b)

{

   return (a > b)? a : b;

}

int knapSack(int W, int wt[], int val[], int n)

{

   int i, w;

   int K[n+1][W+1];

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

   {

      for (w = 0; w <= W; w++)

      {

         if (i==0 || w==0)

            K[i][w] = 0;

         else if (wt[i-1] <= w)

            K[i][w] = max(val[i-1] + K[i-1][w-wt[i-1]],  K[i-1][w]);

         else

            K[i][w] = K[i-1][w];

      }

   }

   return K[n][W];
}

int main()

{

   cout<<"Roll Number 161210040\n";

   int i, n, val[20], wt[20], W;

   cout<<"Enter number of items:"<<endl;

   cin>>n;

   cout<<"Enter value and weight of items:\n";

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

   {

      cin>>val[i]>>wt[i];

   }

   cout<<"Enter Max Allowed Wt:"<<endl;

   cin>>W;

   cout<<"Maximum profit is:"<<endl;

   cout<< knapSack(W, wt, val, n);

   return 0;

}
```

Output



```
Roll Number 161210040
Enter number of items:
3
Enter value and weight of items:
60 10
100 20
120 30
Enter Max Allowed Wt:
50
Maximum profit is:
220
Process returned 0 (0x0)   execution time : 44.461 s
Press any key to continue.
```

Travelling Salesman Problem

//Source Code

```
#include<stdio.h>

int a[10][10],visited[10],n,cost=0;

void get()

{

        int i,j;

        printf("Enter No. of Cities: ");

        scanf("%d",&n);

        printf("\nEnter Cost Matrix\n");

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

        {

                printf("\nEnter Elements of Row # : %d\n",i+1);

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

                        scanf("%d",&a[i][j]);

                visited[i]=0;

        }

        printf("\n\nThe cost list is:\n\n");

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

        {

                printf("\n\n");

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

                        printf("\t%d",a[i][j]);

        }
```

```
}
int least(int c)
{
        int i,nc=999;
        int min=999,kmin;
        for(i=0;i < n;i++)
        {
                if((a[c][i]!=0)&&(visited[i]==0))
                        if(a[c][i] < min)
                        {
                                min=a[i][0]+a[c][i];
                                kmin=a[c][i];
                                nc=i;
                        }
        }
        if(min!=999)
                cost+=kmin;
        return nc;
}
void mincost(int city)
{
        int i,ncity;
        visited[city]=1;
        printf("%d --> ",city+1);
        ncity=least(city);
        if(ncity==999)
        {
                ncity=0;
                printf("%d",ncity+1);
                cost+=a[city][ncity];
                return;
        }
        mincost(ncity);
}
void put()
{
        printf("\n\nMinimum cost:\n\n");
  printf("\n");
        printf("%d",cost);
}

int main()
{
        get();
        printf("\n\nThe Path is:\n\n");
        mincost(0);
        put();
  return 0;
}
```

Output

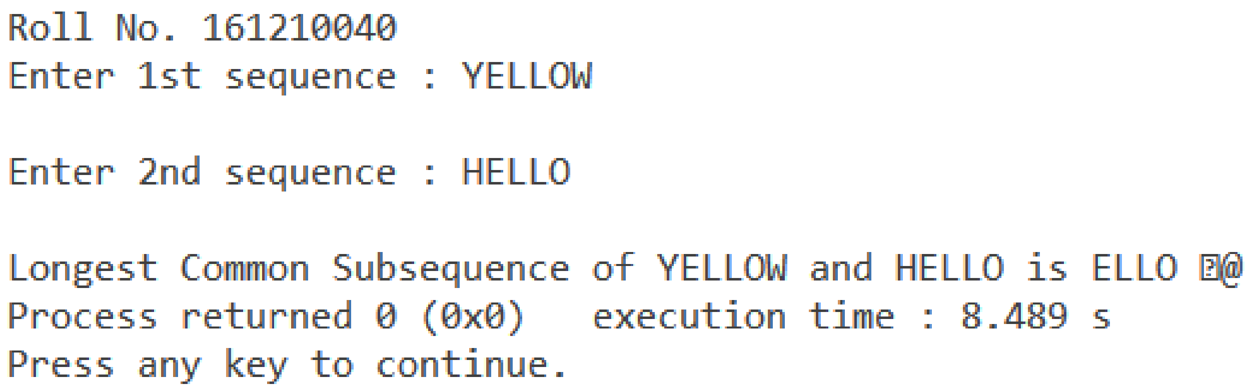
```
Enter No. of Cities: 4

Enter Cost Matrix

Enter Elements of Row # : 1
0 4 1 3

Enter Elements of Row # : 2
4 0 2 1

Enter Elements of Row # : 3
1 0 2 5

Enter Elements of Row # : 4
3 1 5 0


The cost list is:



        0       4       1       3

        4       0       2       1

        1       0       2       5

        3       1       5       0

The Path is:

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

Minimum cost:


11
Process returned 0 (0x0)   execution time : 91.680 s
Press any key to continue.
```

Longest Common Subsequence

//Source Code

```
#include <iostream>

#include <string.h>

using namespace std;

void lcs( char *X, char *Y, int m, int n )

{

  int L[m+1][n+1];

  for (int i=0; i<=m; i++)

  {

   for (int j=0; j<=n; j++)

   {

     if (i == 0 || j == 0)

       L[i][j] = 0;

     else if (X[i-1] == Y[j-1])

       L[i][j] = L[i-1][j-1] + 1;

     else

       L[i][j] = max(L[i-1][j], L[i][j-1]);

   }

  }

  int index = L[m][n];

  char lcs[index+1];

  lcs[index] = ' ';

  int i = m, j = n;

  while (i > 0 && j > 0) {
      if (X[i-1] == Y[j-1])
      {
         lcs[index-1] = X[i-1];
         i--; j--; index--;
      }
      else if (L[i-1][j] > L[i][j-1])
        i--;
      else
        j--;
   }
   cout << "\nLongest Common Subsequence of " << X << " and " << Y << " is " << lcs;
}
int main()
{  char x[20],y[20];
   int m,n;
   cout<<"Roll No. 161210040\n";
   cout<<"Enter 1st sequence : ";
   cin>>x;
   cout<<"\nEnter 2nd sequence : ";
   cin>>y;
   m=strlen(x);
   n=strlen(y);
   lcs(x,y,m,n);
   return 0;
}
```

Output



```
Roll No. 161210040
Enter 1st sequence : YELLOW

Enter 2nd sequence : HELLO

Longest Common Subsequence of YELLOW and HELLO is ELLO �@
Process returned 0 (0x0)   execution time : 8.489 s
Press any key to continue.
```