# Insertion Sort -
## Executed Code With Output -

```cpp
#include <stdio.h>
int main()
{
	int n, array[1000], c, d, t, flag = 0;
	printf("Enter number of elements: \n");
	scanf("%d", &n);
	printf("Enter %d integers\n", n);
	for (c = 0; c < n; c++)
		scanf("%d", &array[c]);
	for (c = 1; c <= n - 1; c++)
	{
		t = array[c];
		for (d = c - 1; d >= 0; d--)
		{
			if (array[d] > t)
			{
				array[d+1] = array[d];
				flag = 1;
			}
			else
				break;
		}
		if (flag)
			array[d+1] = t;
	}
	printf("Sorted list in ascending order: \n");
	for (c = 0; c <= n - 1; c++) {
		printf("%d\n", array[c]);
	}
	return 0;
}
```

Output:

```
C:\Users\Admin\Desktop\AOAL\IS.exe

Enter number of elements:
8
Enter 8 integers
11 7 17 3 9 29 85 9
Sorted list in ascending order:
3
7
9
9
11
17
29
85

------------------------------------
Process exited after 35.12 seconds with return value 0
Press any key to continue . . .
```

Compiler log:
- Warnings: 0
- Output Filename: C:\Users\Admin\Desktop\AOAL\IS.exe
- Output Size: 128.7705078125 KiB
- Compilation Time: 3.25s
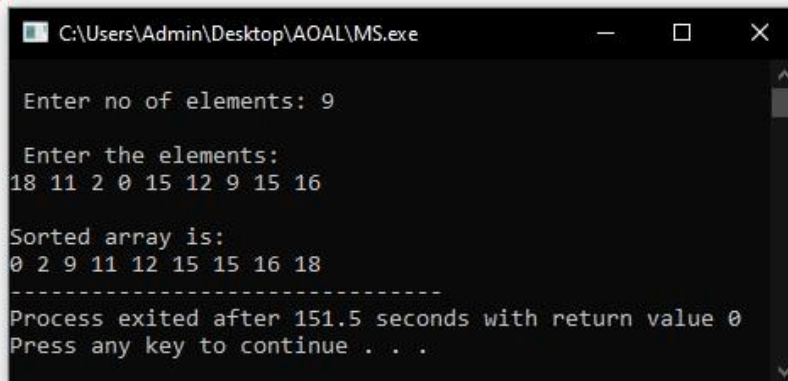
# Merge Sort -

# Executed Code With Output -

```c
1    #include<stdio.h>
2    void mergesort(int[], int, int);
3    void combine(int[], int, int, int);
4    int main()
5    {
6        int a[10], n, i;
7        printf("\n Enter no of elements: ");
8        scanf("%d", & n);
9        printf("\n Enter the elements: \n");
10       for (i = 0; i < n; i++)
11       scanf("%d", & a[i]);
12       mergesort(a, 0, n - 1);
13       printf("\nSorted array is: \n");
14       for (i = 0; i < n; i++)
15       printf("%d ", a[i]);
16       return 0;
17   }
18   void mergesort(int a[10], int low, int high)
19   {
20       int mid;
21       if (low < high)
22       {
23           mid = (low + high) / 2;
24           mergesort(a, low, mid);
25           mergesort(a, mid + 1, high);
26           combine(a, low, mid, high);
27       }
28   }
29   void combine(int a[10], int low, int mid, int high)
30   {
31       int i, j, k, temp[10];

32       i = low;
33       j = mid + 1;
34       k = low;
35       while (i <= mid && j <= high)
36       {
37           if (a[i] < a[j]) {
38               temp[k] = a[i];
39               k++;
40               i++;
41           }
42           else {
43               temp[k] = a[j];
44               k++;
45               j++;
46           }
47       }
48       while (i <= mid)
49       {
50           temp[k] = a[i];
51           k++;
52           i++;
53       }
54       while (j <= high)
55       {
56           temp[k] = a[j];
57           k++;
58           j++;
59       }
60       for (i = low; i <= high; i++)
61       a[i] = temp[i];
62   }
```

```
C:\Users\Admin\Desktop\AOAL\MS.exe                    —    □    ×

 Enter no of elements: 9

 Enter the elements:
18 11 2 0 15 12 9 15 16

Sorted array is:
0 2 9 11 12 15 15 16 18
------------------------------
Process exited after 151.5 seconds with return value 0
Press any key to continue . . .
```
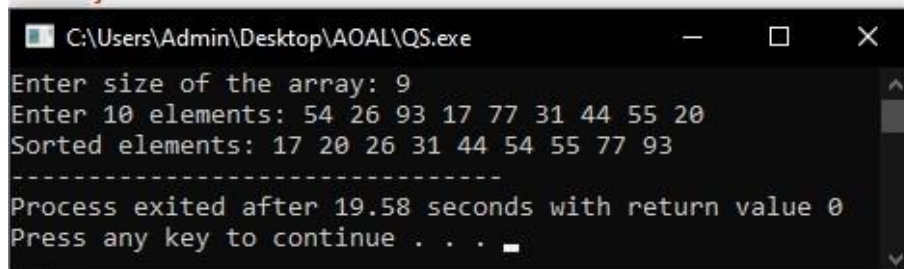
## Quick Sort -

## Executed Code With Output -

```c
1    #include<stdio.h>
2    void quicksort(int[10], int, int);
3    int main()
4    {
5        int a[20], n, i;
6        printf("Enter size of the array: ");
7        scanf("%d", & n);
8        printf("Enter %d elements: ", n+1);
9        for (i = 0; i < n; i++)
10       scanf("%d", & a[i]);
11       quicksort(a, 0, n - 1);
12       printf("Sorted elements: ");
13       for (i = 0; i < n; i++)
14       printf("%d ", a[i]);
15       return 0;
16   }
17   void quicksort(int a[10], int first, int last)
18   {
19       int pivot, j, temp, i;
20       if (first < last)
21       {
22           pivot = first;
23           i = first;
24           j = last;
25           while (i < j)
26           {
27               while (a[i] <= a[pivot] && i < last)
28               i++;
29               while (a[j] > a[pivot])
30               j--;
31               if (i < j)

32               {
33                   temp = a[i];
34                   a[i] = a[j];
35                   a[j] = temp;
36               }
37           }
38           temp = a[pivot];
39           a[pivot] = a[j];
40           a[j] = temp;
41           quicksort(a, first, j - 1);
42           quicksort(a, j + 1, last);
43       }
44   }
```

```
C:\Users\Admin\Desktop\AOAL\QS.exe                    —    □    ✕

Enter size of the array: 9
Enter 10 elements: 54 26 93 17 77 31 44 55 20
Sorted elements: 17 20 26 31 44 54 55 77 93
---------------------------------
Process exited after 19.58 seconds with return value 0
Press any key to continue . . . _
```
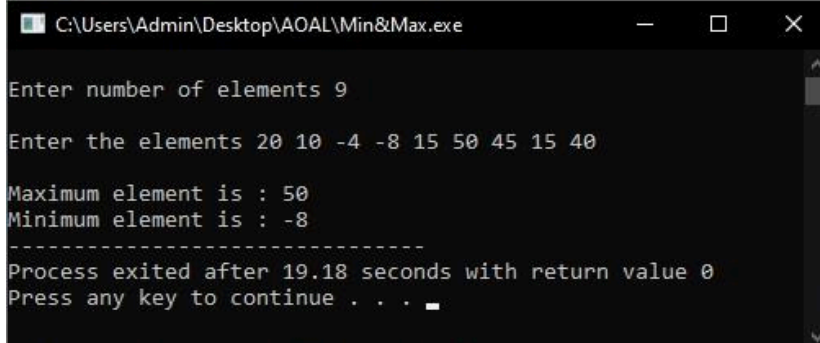
# Minimum and Maximum Element -
# Executed Code With Output -

```c
1    #include <stdio.h>
2    int a[10], max, min;
3    void maxmin(int i, int j)
4    {
5        int newmax, newmin, mid;
6        if (i == j)
7        max = min = a[i];
8        else
9        {
10           if (i == j - 1)
11           {
12               if (a[i] < a[j])
13               {
14                   max = a[j];
15                   min = a[i];
16               }
17               else
18               {
19                   max = a[i];
20                   min = a[j];
21               }
22           }
23           else
24           {
25               mid = (i + j) / 2;
26               maxmin(i, mid);
27               newmax = max;
28               newmin = min;
29               maxmin(mid + 1, j);
30               if (max < newmax)
31                   max = newmax;

32               if (min > newmin)
33                   min = newmin;
34           }
35       }
36   }
37   int main()
38   {
39       int n, i;
40       printf("\nEnter number of elements ");
41       scanf("%d", & n);
42       printf("\nEnter the elements ");
43       for (i = 0; i < n; i++)
44       scanf("%d", & a[i]);
45       maxmin(0, n - 1);
46       printf("\nMaximum element is : %d ", max);
47       printf("\nMinimum element is : %d ", min);
48       return 0;
49   }
```

```
C:\Users\Admin\Desktop\AOAL\Min&Max.exe                    —    □    ×

Enter number of elements 9

Enter the elements 20 10 -4 -8 15 50 45 15 40

Maximum element is : 50
Minimum element is : -8
---------------------------------
Process exited after 19.18 seconds with return value 0
Press any key to continue . . . _
```
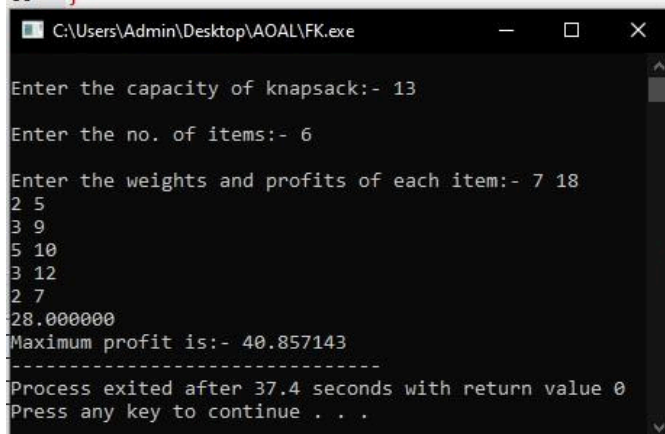
# Fractional Knapsack -
# Executed Code With Output -

```c
1    # include <stdio.h>
2    int main()
3    {
4        float weight[20], profit[20], capacity;
5        int n, i, j;
6        float ratio[20], fract = 1.0, tp = 0, temp;
7        printf("\nEnter the capacity of knapsack:- ");
8        scanf("%f", & capacity);
9        printf("\nEnter the no. of items:- ");
10       scanf("%d", & n);
11       printf("\nEnter the weights and profits of each item:- ");
12       for (i = 0; i < n; i++)
13       scanf("%f %f", & weight[i], & profit[i]);
14       for (i = 0; i < n; i++)
15       ratio[i] = profit[i] / weight[i];
16       for (i = 0; i < n; i++)
17       {
18           for (j = i + 1; j < n; j++)
19           {
20               if (ratio[i] < ratio[j])
21               {
22                   temp = ratio[j];
23                   ratio[j] = ratio[i];
24                   ratio[i] = temp;
25                   temp = weight[j];
26                   weight[j] = weight[i];
27                   weight[i] = temp;
28                   temp = profit[j];
29                   profit[j] = profit[i];
30                   profit[i] = temp;
31               }
32           }
33       }
34       for (i = 0; i < n; i++)
35       {
36           if (weight[i] > capacity)
37           break;
38           else
39           {
40               tp = tp + profit[i];
41               capacity = capacity - weight[i];
42           }
43       }
44       printf("%f ", tp);
45       if (i < n)
46       fract = capacity / weight[i];
47       tp = tp + (fract * profit[i]);
48       printf("\nMaximum profit is:- %f ", tp);
49       return 0;
50   }
```

```
C:\Users\Admin\Desktop\AOAL\FK.exe                    —    □    ×

Enter the capacity of knapsack:- 13

Enter the no. of items:- 6

Enter the weights and profits of each item:- 7 18
2 5
3 9
5 10
3 12
2 7
28.000000
Maximum profit is:- 40.857143
---------------------------------
Process exited after 37.4 seconds with return value 0
Press any key to continue . . .
```

# MST - Prim's Algorithm

# Executed Code With Output -

```c
1    #include <stdio.h>
2    #include <limits.h>
3    #define vertices 5
4    int minimum_key(int k[], int mst[])
5    {
6        int minimum  = INT_MAX, min,i;
7        for (i = 0; i < vertices; i++)
8            if (mst[i] == 0 && k[i] < minimum )
9                minimum = k[i], min = i;
10       return min;
11   }
12   void prim(int g[vertices][vertices])
13   {
14       int parent[vertices];
15       int k[vertices];
16       int mst[vertices];
17       int i, count,edge,v;
18       for (i = 0; i < vertices; i++)
19       {
20           k[i] = INT_MAX;
21           mst[i] = 0;
22       }
23       k[0] = 0;
24       parent[0] = -1;
25       for (count = 0; count < vertices-1; count++)
26       {
27           edge = minimum_key(k, mst);
28           mst[edge] = 1;
29           for (v = 0; v < vertices; v++)
30           {
31               if (g[edge][v] && mst[v] == 0 && g[edge][v] <  k[v])

32               {
33                   parent[v]  = edge, k[v] = g[edge][v];
34               }
35           }
36       }
37       printf("\n Edge \t  Weight\n");
38       for (i = 1; i < vertices; i++)
39       printf(" %d <-> %d    %d \n", parent[i], i, g[i][parent[i]]);
40
41   }
42   int main()
43   {
44       int g[vertices][vertices] = {{0, 0, 3, 0, 0},
45                                    {0, 0, 10, 4, 0},
46                                    {3, 10, 0, 2, 6},
47                                    {0, 4, 2, 0, 1},
48                                    {0, 0, 6, 1, 0},
49                                    };
50       prim(g);
51       return 0;
52   }
```

```
C:\Users\Admin\Desktop\AOAL\PA.exe          —    □    X

Edge     Weight
3 <-> 1    4
0 <-> 2    3
2 <-> 3    2
3 <-> 4    1


---------------------------------
Process exited after 0.4358 seconds with return value 0
Press any key to continue . . .
```

# MST - Kruskal's Algorithm

# Executed Code With Output -

```
1   #include <stdio.h>
2   #define MAX 30
3   typedef struct edge {
4     int u, v, w;
5   } edge;
6   typedef struct edge_list {
7     edge data[MAX];
8     int n;
9   } edge_list;
10  edge_list elist;
11  int Graph[MAX][MAX], n;
12  edge_list spanlist;
13  void kruskalAlgo();
14  int find(int belongs[], int vertexno);
15  void applyUnion(int belongs[], int c1, int c2);
16  void sort();
17  void print();
18  void kruskalAlgo() {
19    int belongs[MAX], i, j, cno1, cno2;
20    elist.n = 0;
21    for (i = 1; i < n; i++)
22      for (j = 0; j < i; j++) {
23        if (Graph[i][j] != 0) {
24          elist.data[elist.n].u = i;
25          elist.data[elist.n].v = j;
26          elist.data[elist.n].w = Graph[i][j];
27          elist.n++;
28        }
29      }
30    sort();
31    for (i = 0; i < n; i++)

32      belongs[i] = i;
33    spanlist.n = 0;
34    for (i = 0; i < elist.n; i++) {
35      cno1 = find(belongs, elist.data[i].u);
36      cno2 = find(belongs, elist.data[i].v);
37      if (cno1 != cno2) {
38        spanlist.data[spanlist.n] = elist.data[i];
39        spanlist.n = spanlist.n + 1;
40        applyUnion(belongs, cno1, cno2);
41      }
42    }
43  }
44  int find(int belongs[], int vertexno) {
45    return (belongs[vertexno]);
46  }
47  void applyUnion(int belongs[], int c1, int c2) {
48    int i;
49
50    for (i = 0; i < n; i++)
51      if (belongs[i] == c2)
52        belongs[i] = c1;
53  }
54  void sort() {
55    int i, j;
56    edge temp;
57    for (i = 1; i < elist.n; i++)
58      for (j = 0; j < elist.n - 1; j++)
59        if (elist.data[j].w > elist.data[j + 1].w) {
60          temp = elist.data[j];
61          elist.data[j] = elist.data[j + 1];
62          elist.data[j + 1] = temp;
```

```
63           }
64     }
65     void print() {
66        int i, cost = 0;
67        for (i = 0; i < spanlist.n; i++) {
68           printf("\n%d - %d : %d", spanlist.data[i].u, spanlist.data[i].v, spanlist.data[i].w);
69           cost = cost + spanlist.data[i].w;
70        }
71        printf("\nSpanning tree cost: %d", cost);
72     }
73     int main() {
74        int i, j, total_cost;
75        n = 6;
76
77        Graph[0][0] = 0;
78        Graph[0][1] = 4;
79        Graph[0][2] = 4;
80        Graph[0][3] = 0;
81        Graph[0][4] = 0;
82        Graph[0][5] = 0;
83        Graph[0][6] = 0;
84
85        Graph[1][0] = 4;
86        Graph[1][1] = 0;
87        Graph[1][2] = 2;
88        Graph[1][3] = 0;
89        Graph[1][4] = 0;
90        Graph[1][5] = 0;
91        Graph[1][6] = 0;
92
93        Graph[2][0] = 4;
```

```
94      Graph[2][1] = 2;
95      Graph[2][2] = 0;
96      Graph[2][3] = 3;
97      Graph[2][4] = 4;
98      Graph[2][5] = 0;
99      Graph[2][6] = 0;
100
101     Graph[3][0] = 0;
102     Graph[3][1] = 0;
103     Graph[3][2] = 3;
104     Graph[3][3] = 0;
105     Graph[3][4] = 3;
106     Graph[3][5] = 0;
107     Graph[3][6] = 0;
108
109     Graph[4][0] = 0;
110     Graph[4][1] = 0;
111     Graph[4][2] = 4;
112     Graph[4][3] = 3;
113     Graph[4][4] = 0;
114     Graph[4][5] = 0;
115     Graph[4][6] = 0;
116
117     Graph[5][0] = 0;
118     Graph[5][1] = 0;
119     Graph[5][2] = 2;
120     Graph[5][3] = 0;
121     Graph[5][4] = 3;
122     Graph[5][5] = 0;
123     Graph[5][6] = 0;
124
125     kruskalAlgo();
126     print();
127 }
```

```
C:\Users\Admin\Desktop\AOAL\KA.exe                    —    □    ✕


2 - 1 : 2
5 - 2 : 2
3 - 2 : 3
4 - 3 : 3
1 - 0 : 4
Spanning tree cost: 14
--------------------------------
Process exited after 0.4115 seconds with return value 0
Press any key to continue . . .
```

- Warnings: 0
- Output Filename: C:\Users\Admin\Desktop\AOAL\KA.exe
- Output Size: 130.1455078125 KiB
- Compilation Time: 2.14s

## APSP - Floyd Warshall Algorithm
## Executed Code With Output -

```c
#include <stdio.h>
#define n 4
#define INF 999
int main()
{
    int A[n][n]={0, 5, 9, INF},{INF, 0, 1, INF},{INF, INF, 0, 2},{INF, 3, INF, 0}};
    int i, j, k;
    for(k = 0; k < n; k++)
    {
        for (i = 0; i < n; i++)
        {
            for (j = 0; j < n; j++)
            {
                if (A[i][k] + A[k][j] < A[i][j])
                    A[i][j] = A[i][k] + A[k][j];
            }
        }
    }
    printf("\n The shortest distances from every pair of vertices are:\n");
    for (i = 0; i < n; i++)
    {
        for (j = 0; j < n; j++)
        {
            if (A[i][j] == INF)
                printf("%s\t", "INF");
            else
                printf("%d\t", A[i][j]);
        }
        printf("\n");
    }
}
```

Compile Log:
```
- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\Admin\Desktop\AOAL\APSP.exe
- Output Size: 129.7724609375 KiB
- Compilation Time: 1.86s
```

Console output:
```
The shortest distances from every pair of vertices are:
0      5      6      8
INF    0      1      3
INF    5      0      2
INF    3      4      0

Process exited after 0.3764 seconds with return value 0
Press any key to continue . . .
```
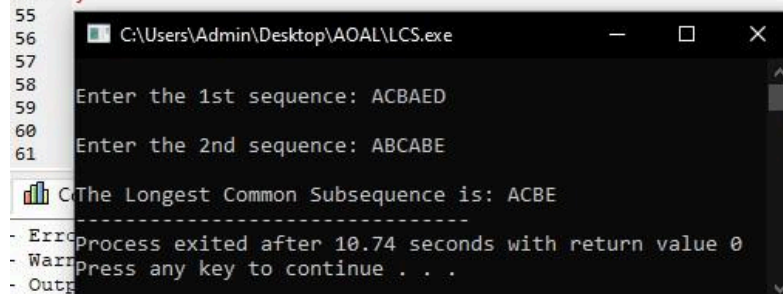
## Longest Common Subsequence -
## Executed Code With Output -

```c
1    #include<stdio.h>
2    #include<string.h>
3    char x[20], y[20], b[20][20];
4    void print_lcs(int i, int j)
5    {
6        if(i==0 || j==0)
7        return;
8        if(b[i][j]=='c')
9        {
10           print_lcs(i-1,j-1);
11           printf("%c",x[i-1]);
12       }
13       else if(b[i][j]=='u')
14           print_lcs(i-1,j);
15       else
16           print_lcs(i,j-1);
17   }
18   int main()
19   {
20       int i,j,m,n,c[20][20];
21       printf("\nEnter the 1st sequence: ");
22       scanf("%s",x);
23       printf("\nEnter the 2nd sequence: ");
24       scanf("%s",y);
25       printf("\nThe Longest Common Subsequence is: ");
26       m=strlen(x);
27       n=strlen(y);
28       for(i=0;i<=m;i++)
29       c[i][0]=0;
30       for(i=0;i<=n;i++)

32       for(i=1;i<=m;i++)
33       {
34           for(j=1;j<=n;j++)
35           {
36               if(x[i-1]==y[j-1])
37               {
38                   c[i][j]=c[i-1][j-1]+1;
39                   b[i][j]='c';
40               }
41               else if(c[i-1][j]>=c[i][j-1])
42               {
43                   c[i][j]=c[i-1][j];
44                   b[i][j]='u';
45               }
46               else
47               {
48                   c[i][j]=c[i][j-1];
49                   b[i][j]='l';
50               }
51           }
52       }
53       print_lcs(m,n);
54   }
55
56
57
58
59
60
61
```

```
C:\Users\Admin\Desktop\AOAL\LCS.exe          —    □    ×

Enter the 1st sequence: ACBAED

Enter the 2nd sequence: ABCABE

The Longest Common Subsequence is: ACBE
--------------------------------
Process exited after 10.74 seconds with return value 0
Press any key to continue . . .
```

```c
#include <stdio.h>                                    //AIML08_PRATHAMESH
void subsetSum(int set[], int subSet[], int n, int subSize, int total, int nodeCount ,int sum)
{
        int i;
        if(total==sum && subSize >1)
        {
                printf("\n");
                for(i = 0; i < subSize; i++)
                        printf(" %d",subSet[i]);
                subsetSum(set,subSet,n,subSize-1,total-set[nodeCount],nodeCount+1,sum);
                //for other subsets
        return;
        }
        else
        {
                for(i=nodeCount; i < n; i++ )
                { //find node along breadth
                        subSet[subSize] = set[i];
                        subsetSum(set,subSet,n,subSize+1,total+set[i],i+1,sum);
                        //do for (next node in depth
                }
        }
}
int main()
{
        int weights[50], size, sum,i,n;
        int subSet[50]; //create subset array to pass parameter of subsetSum
        printf("\n Enter the sum : ");
        scanf("%d",&sum);
        printf("\n Enter the no. of elements: ");
        scanf("%d",&n);
        printf("\n Enter the of elements: ");
        for(i=0;i<n;i++)
        scanf("%d",&weights[i]);
        subsetSum(weights, subSet, n, 0, 0, 0, sum);
}
```

```
Select C:\Users\Admin\Desktop\AOAL\SOS.exe                    —    □    ×

Enter the sum : 20

Enter the no. of elements: 6

Enter the of elements: 3 5 7 8 9 15

3 8 9
5 7 8
5 15
-----------------------------------
Process exited after 15.88 seconds with return value 0
Press any key to continue . . .
```

```c
#include<stdio.h>                                    //AIML08_PRATHAMESH
#include<conio.h>
int m=0,n=4;
int cal(int temp[10][10],int t[10][10])
{
        int i,j,m=0;
        for(i=0;i < n;i++)
                for(j=0;j < n;j++)
                {
                        if(temp[i][j]!=t[i][j])
                        m++;
                }
        return m;
}
int check(int a[10][10],int t[10][10])
{
        int i,j,f=1;
        for(i=0;i < n;i++)
                for(j=0;j < n;j++)
                        if(a[i][j]!=t[i][j])
                                f=0;
        return f;
}
int main()
{
        int p,i,j,n=4,a[10][10],t[10][10],temp[10][10],r[10][10];
        int m=0,x=0,y=0,d=1000,dmin=0,l=0;
        printf("\nEnter the matrix to be solved,space with zero :\n");
        for(i=0;i < n;i++)
                for(j=0;j < n;j++)
                        scanf("%d",&a[i][j]);

        printf("\nEnter the target matrix,space with zero :\n");
        for(i=0;i < n;i++)
                for(j=0;j < n;j++)
                        scanf("%d",&t[i][j]);

        printf("\nEntered Matrix is :\n");
        for(i=0;i < n;i++)
        {
```

```c
        for(j=0;j < n;j++)
                printf("%d\t",a[i][j]);
        printf("\n");
}
printf("\nTarget Matrix is :\n");
for(i=0;i < n;i++)
{
        for(j=0;j < n;j++)
                printf("%d\t",t[i][j]);
        printf("\n");
}
while(!(check(a,t)))
{
        l++;
        d=1000;
        for(i=0;i < n;i++)
                for(j=0;j < n;j++)
                {
                        if(a[i][j]==0)
                        {
                                x=i;
                                y=j;
                        }
                }
        //To move upwards
        for(i=0;i < n;i++)
                for(j=0;j < n;j++)
                        temp[i][j]=a[i][j];
        if(x!=0)
        {
                p=temp[x][y];
                temp[x][y]=temp[x-1][y];
                temp[x-1][y]=p;
        }
        m=cal(temp,t);
        dmin=l+m;
        if(dmin < d)
        {
                d=dmin;
                for(i=0;i < n;i++)
```

```c
                for(j=0;j < n;j++)
                        r[i][j]=temp[i][j];
}
//To move downwards
for(i=0;i < n;i++)
        for(j=0;j < n;j++)
                temp[i][j]=a[i][j];
if(x!=n-1)
{
        p=temp[x][y];
        temp[x][y]=temp[x+1][y];
        temp[x+1][y]=p;
}
m=cal(temp,t);
dmin=l+m;
if(dmin < d)
{
        d=dmin;
        for(i=0;i < n;i++)
                for(j=0;j < n;j++)
                        r[i][j]=temp[i][j];
}
//To move right side
for(i=0;i < n;i++)
        for(j=0;j < n;j++)
                temp[i][j]=a[i][j];
if(y!=n-1)
{
        p=temp[x][y];
        temp[x][y]=temp[x][y+1];
        temp[x][y+1]=p;
}
m=cal(temp,t);
dmin=l+m;
if(dmin < d)
{
        d=dmin;
        for(i=0;i < n;i++)
                for(j=0;j < n;j++)
                        r[i][j]=temp[i][j];
```

```c
                }
                //To move left
                for(i=0;i < n;i++)
                        for(j=0;j < n;j++)
                                temp[i][j]=a[i][j];
                if(y!=0)
                {
                        p=temp[x][y];
                        temp[x][y]=temp[x][y-1];
                        temp[x][y-1]=p;
                }
                m=cal(temp,t);
                dmin=l+m;
                if(dmin < d)
                {
                        d=dmin;
                        for(i=0;i < n;i++)
                                for(j=0;j < n;j++)
                                        r[i][j]=temp[i][j];
                }
                printf("\nCalculated Intermediate Matrix Value :\n");
                for(i=0;i < n;i++)
                {
                        for(j=0;j < n;j++)
                          printf("%d\t",r[i][j]);
                        printf("\n");
                }
                for(i=0;i < n;i++)
                        for(j=0;j < n;j++)
                        {
                          a[i][j]=r[i][j];
                          temp[i][j]=0;
                        }
                printf("Minimum cost : %d\n",d);
        }
}
//AIML51_SUDHAM
```

```
C:\Users\Admin\Desktop\AOAL\15-PP.exe

Enter the matrix to be solved,space with zero :
1 2 3 4
5 6 0 8
9 10 7 11
13 14 15 12

Enter the target matrix,space with zero :
1 2 3 4
5 6 7 8
9 10 11 12
13 14 15 0

Entered Matrix is :
1       2       3       4
5       6       0       8
9       10      7       11
13      14      15      12

Target Matrix is :
1       2       3       4
5       6       7       8
9       10      11      12
13      14      15      0

Calculated Intermediate Matrix Value :
1       2       3       4
5       6       7       8
9       10      0       11
13      14      15      12
Minimum cost : 4

Calculated Intermediate Matrix Value :
1       2       3       4
5       6       7       8
9       10      11      0
13      14      15      12
Minimum cost : 4

Calculated Intermediate Matrix Value :
1       2       3       4
5       6       7       8
9       10      11      12
13      14      15      0
Minimum cost : 3


---------------------------------
Process exited after 53.09 seconds with return value 0
Press any key to continue . . . _
```

```c
#include<stdio.h>                                    //AIML08_PRATHAMESH
#include<string.h>
int main ()
{
        char txt[80], pat[80], flag=0;
        int q = 101; // prime number
        int d = 256;
        printf ("Enter the text string \n");
        scanf ("%s", txt);
        printf ("Enter the pattern to be searched \n");
        scanf ("%s", pat);
        int M = strlen(txt);
        int N = strlen (pat);
        int i, j;
        int p = 0; // hash value for pattern
        int t = 0; // hash value for text
        int h = 1;
        for (i = 0; i < N - 1; i++)
                h = (h * d) % q;
        for (i = 0; i < N; i++)
        {
                p = (d * p + pat[i]) % q;
                t = (d * t + txt[i]) % q;
        }
        for (i = 0; i <= M - N; i++)
        {
                if (p == t)
                {
                        for (j = 0; j < N; j++)
                        {
                                if (txt[i + j] != pat[j])
                                break;
                        }
                        if (j == N)
                        {
                                printf ("Pattern found at index %d \n", i);
```
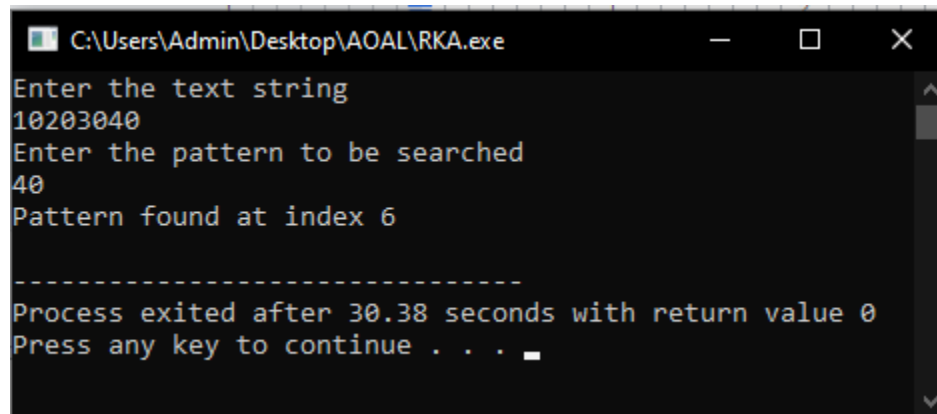
```
                        flag=1;
                }
        }
        if (i < M - N)
        {
                t = (d * (t - txt[i] * h) + txt[i + N]) % q;
                if (t < 0)
                t = (t + q);
        }
    }
    if (flag==0)
    printf("\n Pattern not found\n");
}
```



```
C:\Users\Admin\Desktop\AOAL\RKA.exe

Enter the text string
10203040
Enter the pattern to be searched
40
Pattern found at index 6


----------------------------------
Process exited after 30.38 seconds with return value 0
Press any key to continue . . .
```

```c
#include <stdio.h>
int main()
{
  int n, array[1000], c, d, t, flag = 0;
  printf("Enter number of elements: \n");
  scanf("%d", &n);
  printf("Enter %d integers\n", n);
  for (c = 0; c < n; c++)
    scanf("%d", &array[c]);
  for (c = 1 ; c <= n - 1; c++)
  {
    t = array[c];
    for (d = c - 1 ; d >= 0; d--)
       {
      if (array[d] > t)
         {
        array[d+1] = array[d];
        flag = 1;
       }
      else
        break;
    }
    if (flag)
      array[d+1] = t;
  }
  printf("Sorted list in ascending order: \n");
  for (c = 0; c <= n - 1; c++) {
    printf("%d\n", array[c]);
  }
  return 0;
}
```

```
C:\Users\Admin\Desktop\AOAL\IS.exe                    —    □    ×

Enter number of elements:
8
Enter 8 integers
11 7 17 3 9 29 85 9
Sorted list in ascending order:
3
7
9
9
11
17
29
85

---------------------------------
Process exited after 53.13 seconds with return value 0
Press any key to continue . . . _
```

```c
#include<stdio.h>
void mergesort(int[], int, int);
void combine(int[], int, int, int);
int main()
{
        int a[10], n, i;
        printf("\n Enter no of elements: ");
        scanf("%d", & n);
        printf("\n Enter the elements: \n");
        for (i = 0; i < n; i++)
        scanf("%d", & a[i]);
        mergesort(a, 0, n - 1);
        printf("\nSorted array is: \n");
        for (i = 0; i < n; i++)
        printf("%d ", a[i]);
        return 0;
}
void mergesort(int a[10], int low, int high)
{
        int mid;
        if (low < high)
        {
                mid = (low + high) / 2;
                mergesort(a, low, mid);
                mergesort(a, mid + 1, high);
                combine(a, low, mid, high);
        }
}
void combine(int a[10], int low, int mid, int high)
{
        int i, j, k, temp[10];
        i = low;
        j = mid + 1;
        k = low;
        while (i <= mid && j <= high)
        {
                if (a[i] < a[j]) {
                        temp[k] = a[i];
                        k++;
                        i++;
```
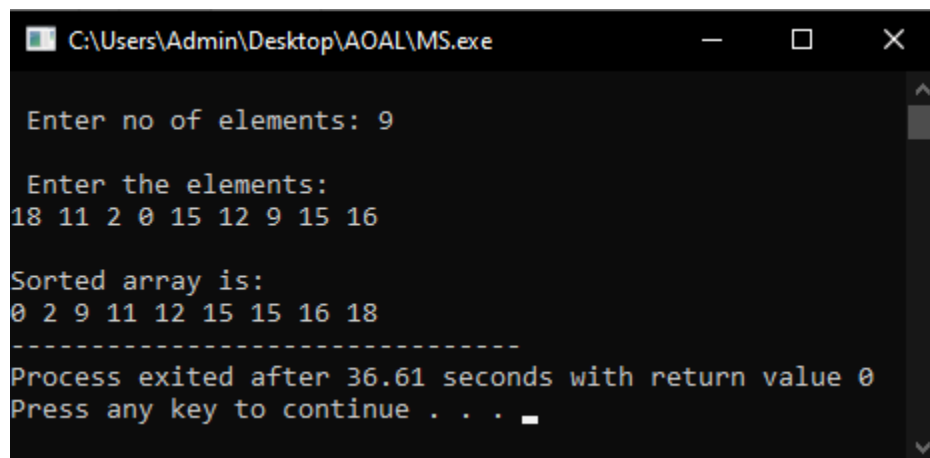
```
            }
            else {
                    temp[k] = a[j];
                    k++;
                    j++;
            }
        }
        while (i <= mid)
        {
                temp[k] = a[i];
                k++;
                i++;
        }
        while (j <= high)
        {
                temp[k] = a[j];
                k++;
                j++;
        }
        for (i = low; i <= high; i++)
        a[i] = temp[i];
}
```

```c
#include<stdio.h>
void quicksort(int[10], int, int);
int main()
{
        int a[20], n, i;
        printf("Enter size of the array: ");
        scanf("%d", & n);
        printf("Enter %d elements: ", n+1);
        for (i = 0; i < n; i++)
        scanf("%d", & a[i]);
        quicksort(a, 0, n - 1);
        printf("Sorted elements: ");
        for (i = 0; i < n; i++)
        printf("%d ", a[i]);
        return 0;
}
void quicksort(int a[10], int first, int last)
{
        int pivot, j, temp, i;
        if (first < last)
        {
                pivot = first;
                i = first;
                j = last;
                while (i < j)
                {
                        while (a[i] <= a[pivot] && i < last)
                        i++;
                        while (a[j] > a[pivot])
                        j--;
                        if (i < j)
                        {
                                temp = a[i];
```

```
                    a[i] = a[j];
                    a[j] = temp;
               }
          }
          temp = a[pivot];
          a[pivot] = a[j];
          a[j] = temp;
          quicksort(a, first, j - 1);
          quicksort(a, j + 1, last);
     }
}
```

```
C:\Users\Admin\Desktop\AOAL\QS.exe                    —    □    ✕

Enter size of the array: 9
Enter 10 elements: 54 26 93 17 77 31 44 55 20
Sorted elements: 17 20 26 31 44 54 55 77 93
--------------------------------
Process exited after 28.56 seconds with return value 0
Press any key to continue . . . ▄
```

```c
#include <stdio.h>
int a[10], max, min;
void maxmin(int i, int j)
{
        int newmax, newmin, mid;
        if (i == j)
        max = min = a[i];
        else
        {
                if (i == j - 1)
                {
                        if (a[i] < a[j])
                        {
                                max = a[j];
                                min = a[i];
                        }
                        else
                        {
                                max = a[i];
                                min = a[j];
                        }
                }
                else
                {
                        mid = (i + j) / 2;
                        maxmin(i, mid);
                        newmax = max;
                        newmin = min;
                        maxmin(mid + 1, j);
                        if (max < newmax)
                                max = newmax;
                        if (min > newmin)
                                min = newmin;
```
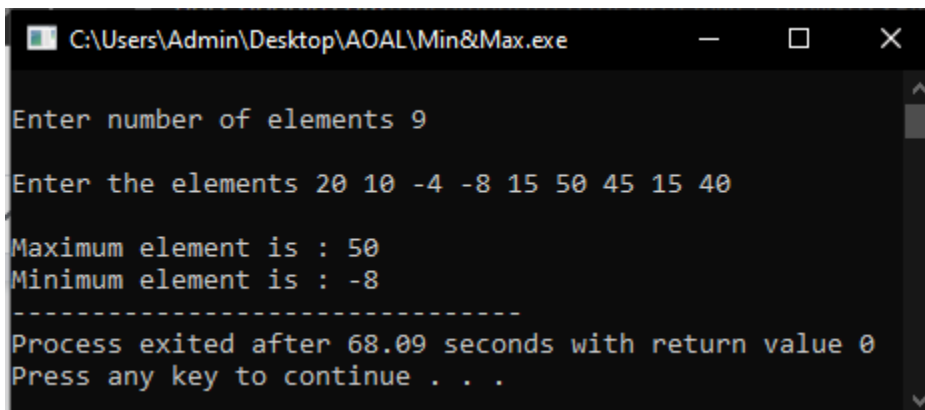
```c
            }
        }
}
int main()
{
        int n, i;
        printf("\nEnter number of elements ");
        scanf("%d", & n);
        printf("\nEnter the elements ");
        for (i = 0; i < n; i++)
        scanf("%d", & a[i]);
        maxmin(0, n - 1);
        printf("\nMaximum element is : %d ", max);
        printf("\nMinimum element is : %d ", min);
        return 0;
}
```



```
C:\Users\Admin\Desktop\AOAL\Min&Max.exe                    —    □    ×

Enter number of elements 9

Enter the elements 20 10 -4 -8 15 50 45 15 40

Maximum element is : 50
Minimum element is : -8
---------------------------------
Process exited after 68.09 seconds with return value 0
Press any key to continue . . .
```
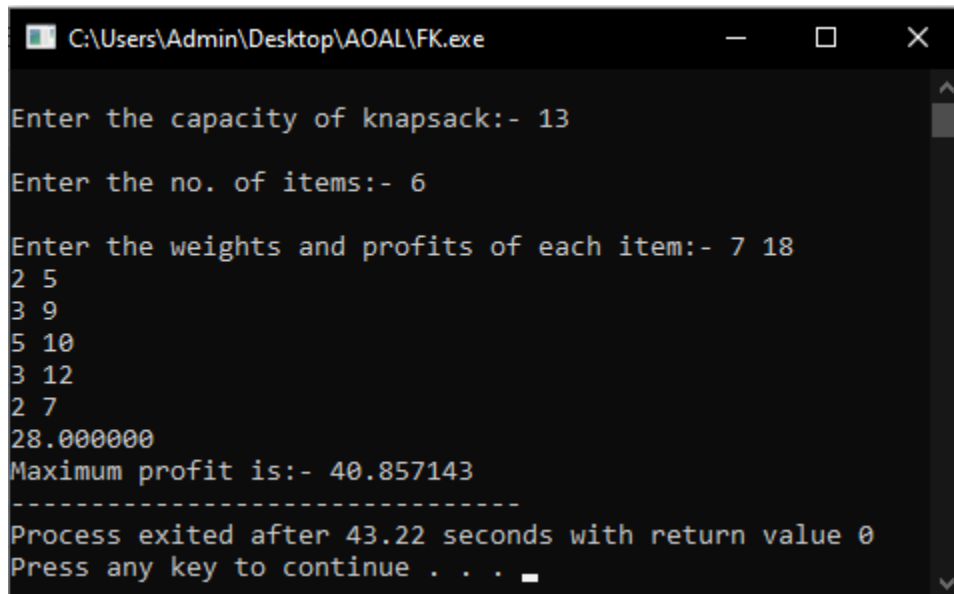
```c
# include <stdio.h>
int main()
{
        float weight[20], profit[20], capacity;
        int n, i, j;
        float ratio[20], fract = 1.0, tp = 0, temp;
        printf("\nEnter the capacity of knapsack:- ");
        scanf("%f", & capacity);
        printf("\nEnter the no. of items:- ");
        scanf("%d", & n);
        printf("\nEnter the weights and profits of each item:- ");
        for (i = 0; i < n; i++)
        scanf("%f %f", & weight[i], & profit[i]);
        for (i = 0; i < n; i++)
        ratio[i] = profit[i] / weight[i];
        for (i = 0; i < n; i++)
        {
                for (j = i + 1; j < n; j++)
                {
                        if (ratio[i] < ratio[j])
                        {
                                temp = ratio[j];
                                ratio[j] = ratio[i];
                                ratio[i] = temp;
                                temp = weight[j];
                                weight[j] = weight[i];
                                weight[i] = temp;
                                temp = profit[j];
                                profit[j] = profit[i];
                                profit[i] = temp;
                        }
                }
        }
```

```c
for (i = 0; i < n; i++)
{
        if (weight[i] > capacity)
        break;
        else
        {
                tp = tp + profit[i];
                capacity = capacity - weight[i];
        }
}
printf("%f ", tp);
if (i < n)
fract = capacity / weight[i];
tp = tp + (fract * profit[i]);
printf("\nMaximum profit is:- %f ", tp);
return 0;
}
```

```
C:\Users\Admin\Desktop\AOAL\FK.exe                    —    □    ✕

Enter the capacity of knapsack:- 13

Enter the no. of items:- 6

Enter the weights and profits of each item:- 7 18
2 5
3 9
5 10
3 12
2 7
28.000000
Maximum profit is:- 40.857143
-------------------------------
Process exited after 43.22 seconds with return value 0
Press any key to continue . . . _
```

```c
#include <stdio.h>
#include <limits.h>
#define vertices 5
int minimum_key(int k[], int mst[])
{
    int minimum  = INT_MAX, min,i;
    for (i = 0; i < vertices; i++)
        if (mst[i] == 0 && k[i] < minimum )
            minimum = k[i], min = i;
    return min;
}
void prim(int g[vertices][vertices])
{
    int parent[vertices];
    int k[vertices];
    int mst[vertices];
    int i, count,edge,v;
    for (i = 0; i < vertices; i++)
    {
        k[i] = INT_MAX;
        mst[i] = 0;
    }
    k[0] = 0;
    parent[0] = -1;
    for (count = 0; count < vertices-1; count++)
    {
        edge = minimum_key(k, mst);
        mst[edge] = 1;
        for (v = 0; v < vertices; v++)
        {
            if (g[edge][v] && mst[v] == 0 && g[edge][v] <  k[v])
            {
                parent[v]  = edge, k[v] = g[edge][v];
```

```c
            }
        }
    }
    printf("\n Edge \t  Weight\n");
    for (i = 1; i < vertices; i++)
    printf(" %d <-> %d    %d \n", parent[i], i, g[i][parent[i]]);


}
int main()
{
    int g[vertices][vertices] = {{0, 0, 3, 0, 0},
                        {0, 0, 10, 4, 0},
                        {3, 10, 0, 2, 6},
                        {0, 4, 2, 0, 1},
                        {0, 0, 6, 1, 0},
                        };
    prim(g);
    return 0;
}
```
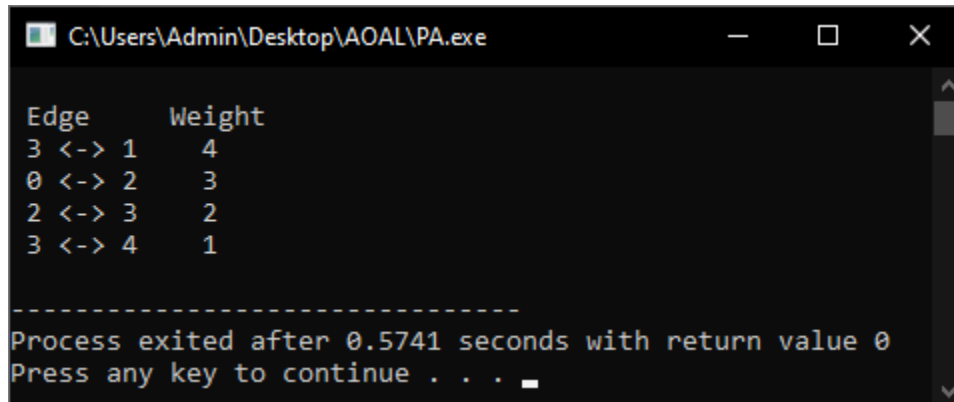
```c
#include <stdio.h>
#define MAX 30
typedef struct edge {
  int u, v, w;
} edge;
typedef struct edge_list {
  edge data[MAX];
  int n;
} edge_list;
edge_list elist;
int Graph[MAX][MAX], n;
edge_list spanlist;
void kruskalAlgo();
int find(int belongs[], int vertexno);
void applyUnion(int belongs[], int c1, int c2);
void sort();
void print();
void kruskalAlgo() {
  int belongs[MAX], i, j, cno1, cno2;
  elist.n = 0;
  for (i = 1; i < n; i++)
    for (j = 0; j < i; j++) {
      if (Graph[i][j] != 0) {
        elist.data[elist.n].u = i;
        elist.data[elist.n].v = j;
        elist.data[elist.n].w = Graph[i][j];
        elist.n++;
      }
    }
  sort();
  for (i = 0; i < n; i++)
    belongs[i] = i;
  spanlist.n = 0;
  for (i = 0; i < elist.n; i++) {
    cno1 = find(belongs, elist.data[i].u);
    cno2 = find(belongs, elist.data[i].v);
```

```c
    if (cno1 != cno2) {
      spanlist.data[spanlist.n] = elist.data[i];
      spanlist.n = spanlist.n + 1;
      applyUnion(belongs, cno1, cno2);
    }
  }
}
int find(int belongs[], int vertexno) {
  return (belongs[vertexno]);
}
void applyUnion(int belongs[], int c1, int c2) {
  int i;

  for (i = 0; i < n; i++)
    if (belongs[i] == c2)
      belongs[i] = c1;
}
void sort() {
  int i, j;
  edge temp;
  for (i = 1; i < elist.n; i++)
    for (j = 0; j < elist.n - 1; j++)
      if (elist.data[j].w > elist.data[j + 1].w) {
        temp = elist.data[j];
        elist.data[j] = elist.data[j + 1];
        elist.data[j + 1] = temp;
      }
}
void print() {
  int i, cost = 0;
  for (i = 0; i < spanlist.n; i++) {
    printf("\n%d - %d : %d", spanlist.data[i].u, spanlist.data[i].v, spanlist.data[i].w);
    cost = cost + spanlist.data[i].w;
  }
  printf("\nSpanning tree cost: %d", cost);
}
```

```c
int main() {
  int i, j, total_cost;
  n = 6;

  Graph[0][0] = 0;
  Graph[0][1] = 4;
  Graph[0][2] = 4;
  Graph[0][3] = 0;
  Graph[0][4] = 0;
  Graph[0][5] = 0;
  Graph[0][6] = 0;

  Graph[1][0] = 4;
  Graph[1][1] = 0;
  Graph[1][2] = 2;
  Graph[1][3] = 0;
  Graph[1][4] = 0;
  Graph[1][5] = 0;
  Graph[1][6] = 0;

  Graph[2][0] = 4;
  Graph[2][1] = 2;
  Graph[2][2] = 0;
  Graph[2][3] = 3;
  Graph[2][4] = 4;
  Graph[2][5] = 0;
  Graph[2][6] = 0;

  Graph[3][0] = 0;
  Graph[3][1] = 0;
  Graph[3][2] = 3;
  Graph[3][3] = 0;
  Graph[3][4] = 3;
  Graph[3][5] = 0;
  Graph[3][6] = 0;
```
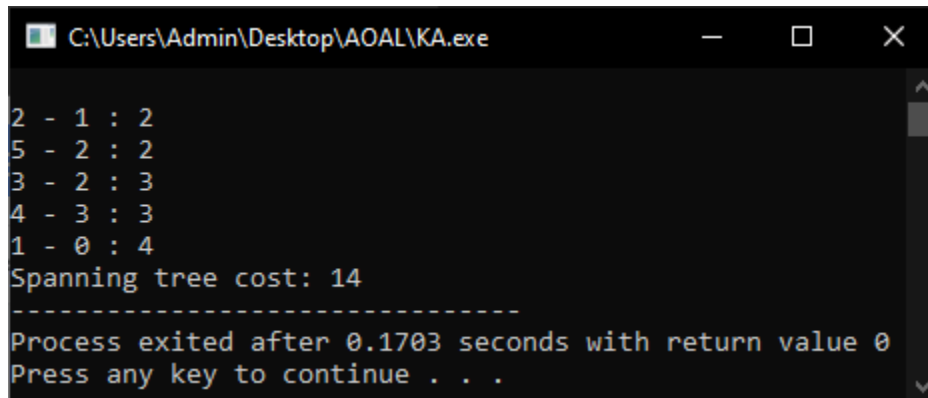
```
Graph[4][0] = 0;
Graph[4][1] = 0;
Graph[4][2] = 4;
Graph[4][3] = 3;
Graph[4][4] = 0;
Graph[4][5] = 0;
Graph[4][6] = 0;

Graph[5][0] = 0;
Graph[5][1] = 0;
Graph[5][2] = 2;
Graph[5][3] = 0;
Graph[5][4] = 3;
Graph[5][5] = 0;
Graph[5][6] = 0;

kruskalAlgo();
print();
}
```



```
2 - 1 : 2
5 - 2 : 2
3 - 2 : 3
4 - 3 : 3
1 - 0 : 4
Spanning tree cost: 14
--------------------------------
Process exited after 0.1703 seconds with return value 0
Press any key to continue . . .
```

```c
#include <stdio.h>
#define n 4
#define INF 999
int main()
{
    int A[n][n] ={ {0, 5, 9, INF},{INF, 0, 1, INF},{INF, INF, 0, 2},{INF, 3, INF, 0}};
    int i, j, k;
    for(k = 0; k < n; k++)
    {
        for (i = 0; i < n; i++)
        {
            for (j = 0; j < n; j++)
            {
                if (A[i][k] + A[k][j] < A[i][j])
                    A[i][j] = A[i][k] + A[k][j];
            }
        }
    }
    printf("\n The shortest distances from every pair of vertices are:\n");
    for (i = 0; i < n; i++)
    {
        for (j = 0; j < n; j++)
        {
            if (A[i][j] == INF)
                printf("%s\t", "INF");
            else
                printf("%d\t", A[i][j]);
        }
        printf("\n");
    }
}
```

```
The shortest distances from every pair of vertices are:
0        5        6        8
INF      0        1        3
INF      5        0        2
INF      3        4        0


--------------------------------
Process exited after 0.7998 seconds with return value 0
Press any key to continue . . .
```

```c
#include<stdio.h>
#include<string.h>
char x[20], y[20], b[20][20];
void print_lcs(int i, int j)
{
        if(i==0 || j==0)
        return;
        if(b[i][j]=='c')
        {
                print_lcs(i-1,j-1);
                printf("%c",x[i-1]);
        }
        else if(b[i][j]=='u')
                print_lcs(i-1,j);
        else
                print_lcs(i,j-1);
}
int main()
{
        int i,j,m,n,c[20][20];
        printf("\nEnter the 1st sequence: ");
        scanf("%s",x);
        printf("\nEnter the 2nd sequence: ");
        scanf("%s",y);
        printf("\nThe Longest Common Subsequence is: ");
        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++)
        {
```

```c
        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';
            }
            else if(c[i-1][j]>=c[i][j-1])
            {
                c[i][j]=c[i-1][j];
                b[i][j]='u';
            }
            else
            {
                c[i][j]=c[i][j-1];
                b[i][j]='l';
            }
        }
    }
    print_lcs(m,n);
}
```
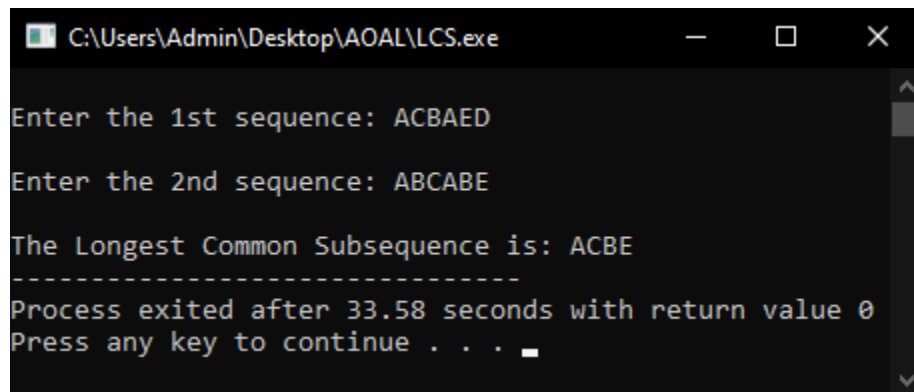


```
Enter the 1st sequence: ACBAED

Enter the 2nd sequence: ABCABE

The Longest Common Subsequence is: ACBE
------------------------------
Process exited after 33.58 seconds with return value 0
Press any key to continue . . . _
```