

## CSA0672 – DAA – DAY 3

K Ganesh

192011132

### 1. Write a C program to merge sort using divide and Conquer

#### Program:

```
#include<stdio.h>

void mergesort(int a[],int i,int j);
void merge(int a[],int i1,int j1,int i2,int j2);

int main()
{
    int a[30],n,i;
    printf("Enter no of elements:");
    scanf("%d",&n);
    printf("Enter array elements:\n");
    for(i=0;i<n;i++)
    {
        scanf("%d",&a[i]);
    }
    mergesort(a,0,n-1);
    printf("Merge Sort : \n");
    for(i=0;i<n;i++)
    {
        printf("%d\n",a[i]);
    }
    return 0;
}
```

```

void mergesort(int a[],int i,int j)
{
int mid;
    if(i<j)
    {
        mid=(i+j)/2;
        mergesort(a,i,mid);
        mergesort(a,mid+1,j);
        merge(a,i,mid,mid+1,j);
    }
}

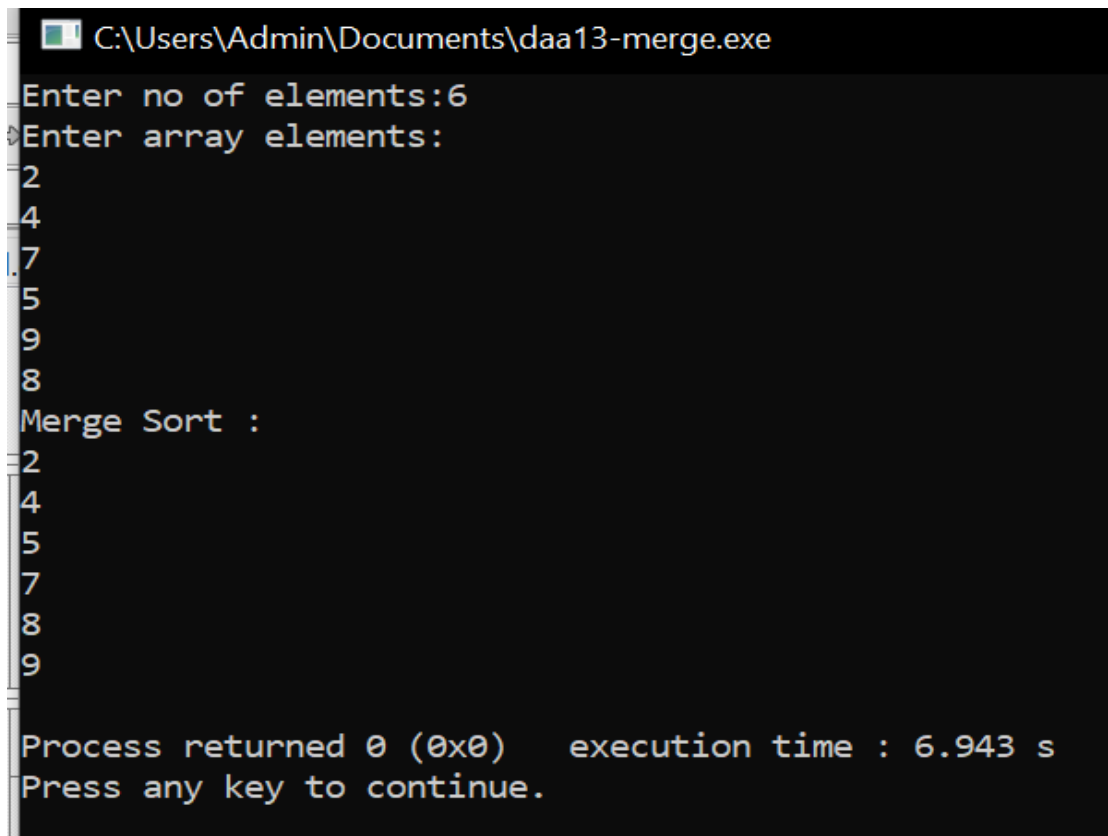
void merge(int a[],int i1,int j1,int i2,int j2)
{
    int temp[50];
    int i,j,k;
    i=i1;
    j=i2;
    k=0;
    while(i<=j1 && j<=j2)
    {
        if(a[i]<a[j])
        {
            temp[k++]=a[i++];
        }
        else
        {
            temp[k++]=a[j++];
        }
    }
}

```

```

    }
}
while(i<=j1)
{
    temp[k++]=a[i++];
}
while(j<=j2)
{
    temp[k++]=a[j++];
}
for(i=i1,j=0;i<=j2;i++,j++)
{
    a[i]=temp[j];
}
}

```



```

C:\Users\Admin\Documents\daa13-merge.exe
Enter no of elements:6
Enter array elements:
2
4
7
5
9
8
Merge Sort :
2
4
5
7
8
9
Process returned 0 (0x0)   execution time : 6.943 s
Press any key to continue.

```

## 2. Write a C program to find max-min using divide and Conquer

### Program:

```
#include<stdio.h>
void mergesort(int a[],int i,int j);
void merge(int a[],int i1,int j1,int i2,int j2);
int main()
{
    int a[30],n,i;
    printf("Enter no of elements:");
    scanf("%d",&n);
    printf("Enter array elements:\n");
    for(i=0;i<n;i++)
    {
        scanf("%d",&a[i]);
    }
    mergesort(a,0,n-1);
    printf("\nMin : %d",a[0]);
    printf("\nMax : %d",a[n-1]);
    return 0;
}

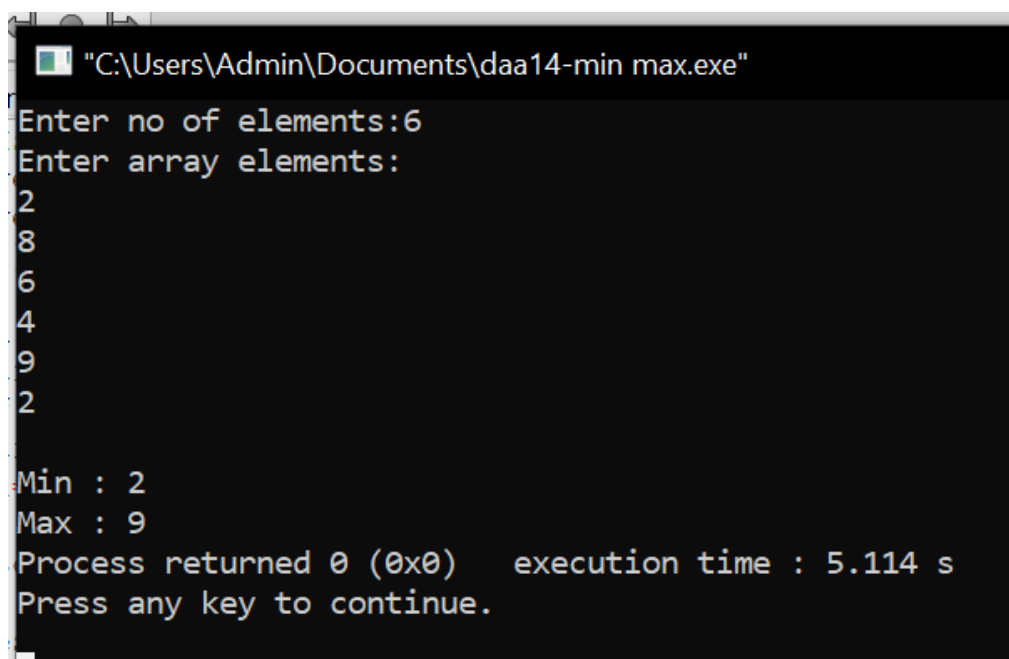
void mergesort(int a[],int i,int j)
{
    int mid;
    if(i<j)
    {
        mid=(i+j)/2;
        mergesort(a,i,mid);
        mergesort(a,mid+1,j);
        merge(a,i,mid,mid+1,j);
    }
}

void merge(int a[],int i1,int j1,int i2,int j2)
{
    int temp[50];
    int i,j,k;
    i=i1;
    j=i2;
```

```

k=0;
while(i<=j1 && j<=j2)
{
    if(a[i]<a[j])
    {
        temp[k++]=a[i++];
    }
    else
    {
        temp[k++]=a[j++];
    }
}
while(i<=j1)
{
    temp[k++]=a[i++];
}
while(j<=j2)
{
    temp[k++]=a[j++];
}
for(i=i1,j=0;i<=j2;i++,j++)
{
    a[i]=temp[j];
}
}

```



```

"C:\Users\Admin\Documents\daa14-min max.exe"
Enter no of elements:6
Enter array elements:
2
8
6
4
9
2

Min : 2
Max : 9
Process returned 0 (0x0)   execution time : 5.114 s
Press any key to continue.

```

**3. Write a program to compute container loader Problem for the given values and estimate time complexity.**

**N=8 be total no of containers having weights (w1, w2, w3,...w8) = [ 50, 100, 30, 80, 90, 200, 150, 20 ]. Capacity value = 100**

**Program:**

```
#include<stdio.h>

int main()
{
    int c=0;
    int n,e,w[20],w1[20],x[20],i,j,k,j1=0;

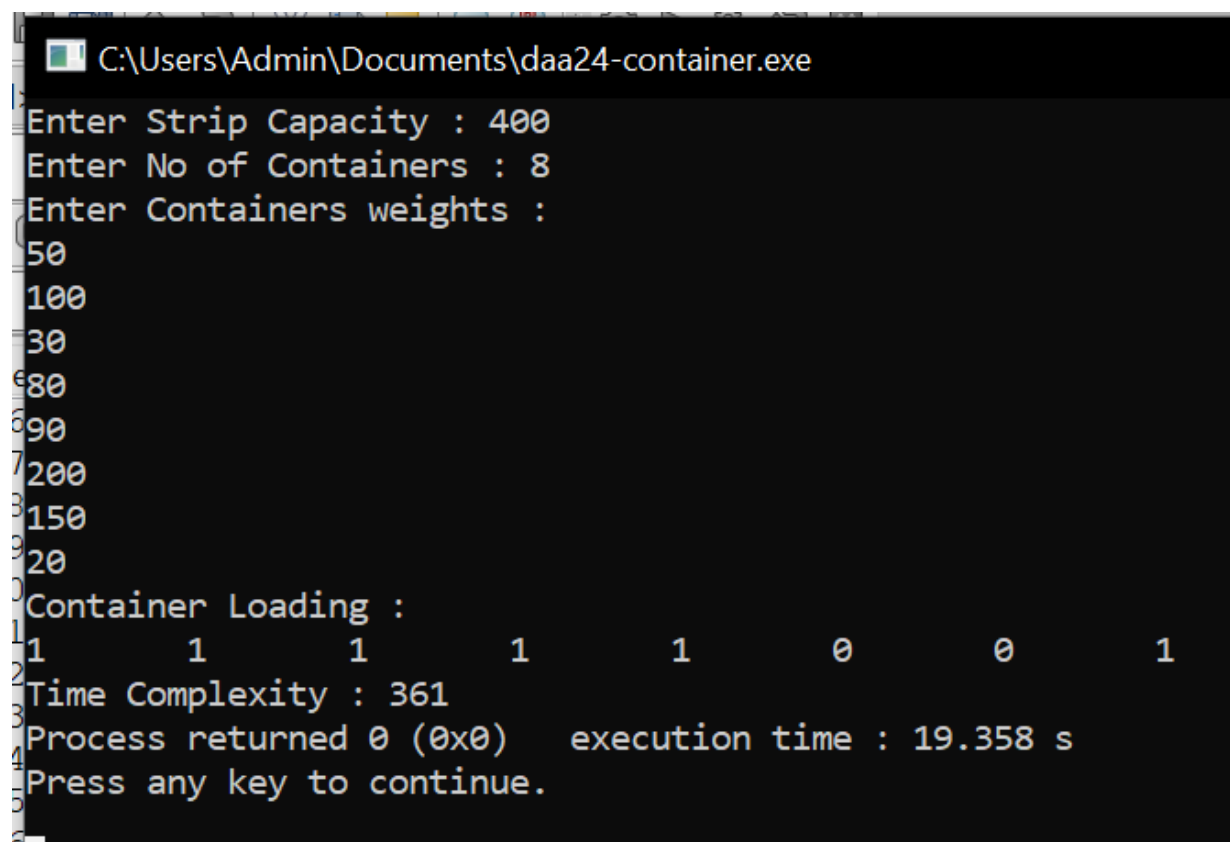
    c++;
    printf("Enter Strip Capacity : ");
    scanf("%d",&e);
    printf("Enter No of Containers : ");
    scanf("%d",&n);
    printf("Enter Containers weights : \n");
    for(i=0;i<n;i++)
    {
        c++;
        scanf("%d",&w[i]);
    }
    c++;
    for(i=0;i<n;i++)
    {
        c++;
        x[i]=0;
```

```
}  
  
c++;  
for(i=0;i<n;i++)  
{  
    c++;  
    w1[i]=w[i];  
}  
  
c++;  
for(i=0;i<n;i++)  
{  
    c++;  
    for(j=0;j<n;j++)  
    {  
        c++;  
        c++;  
        if(w[i]<w[j])  
        {  
            k=w[i];  
            c++;  
            w[i]=w[j];  
            c++;  
            w[j]=k;  
            c++;  
        }  
    }  
}  
  
c++;
```

```
}  
c++;  
for(i=0;i<n;i++)  
{  
    c++;  
    c++;  
    if(e>w[i])  
    {  
        e=e-w[i];  
        c++;  
        for(j=0;j<n;j++)  
        {  
            c++;  
            c++;  
            if(w[i]==w1[j])  
            {  
                x[j]=1;  
                c++;  
            }  
        }  
        c++;  
    }  
}  
c++;  
printf("Container Loading :\n");  
for(i=0;i<n;i++)
```



```
{  
    c++;  
    printf("%d\t",x[i]);  
}  
c++;  
printf("\nTime Complexity : %d",c);  
}
```



```
C:\Users\Admin\Documents\daa24-container.exe  
1 Enter Strip Capacity : 400  
2 Enter No of Containers : 8  
3 Enter Containers weights :  
4 50  
5 100  
6 30  
7 80  
8 90  
9 200  
0 150  
1 20  
2 Container Loading :  
3 1      1      1      1      1      0      0      1  
4 Time Complexity : 361  
5 Process returned 0 (0x0) execution time : 19.358 s  
6 Press any key to continue.
```

**4. Identify the M-th maximum number and Nth minimum number in an array and then find the sum of it and difference of it.**

**Test cases:**

**output –**

- |   |                 |
|---|-----------------|
| a. {16, 16, 16 16, 16}, M = 0, N = 1        | (illegal input) |
| b. {0, 0, 0, 0}, M = 1, N = 2               | 0               |
| c. {-12, -78, -35, -42, -85}, M = 3 , N = 3 | -7              |
| d. {15, 19, 34, 56, 12}, M = 6 , N = -3     | (illegal input) |
| e. {85, 45, 65, 75, 95}, M = 5 , N = 2      | -20             |

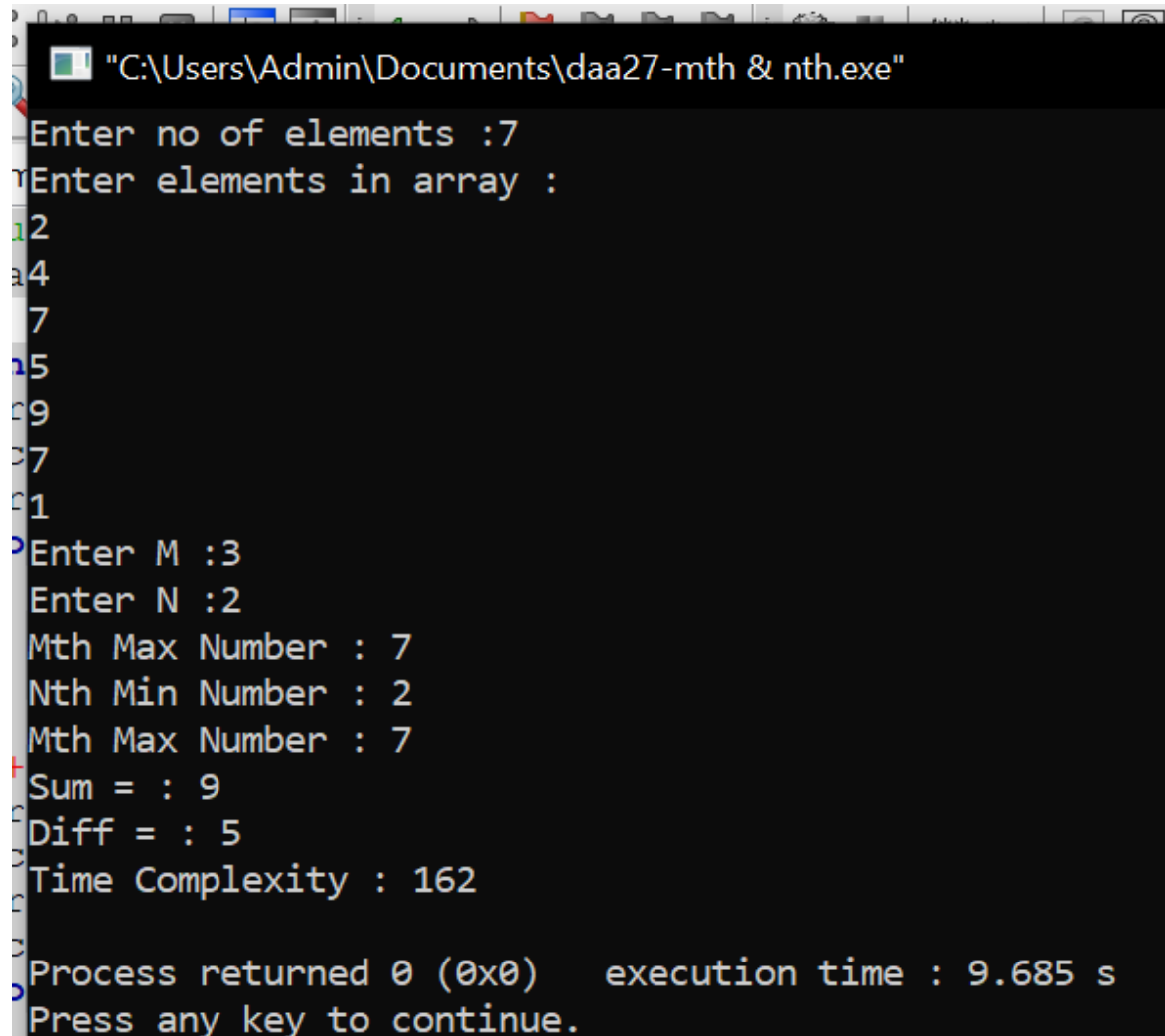
**Program:**

```
#include<stdio.h>

int main()
{
    int a,ar[100],m,n,i,j,sum,diff,k,c=0;
    printf("Enter no of elements :");
    scanf("%d",&a);
    printf("Enter elements in array :\n");
    for(i=0;i<a;i++)
    {
        c++;
        scanf("%d",&ar[i]);
    }
    c++;
    printf("Enter M :");
    scanf("%d",&m);
    printf("Enter N :");
```

```
scanf("%d",&n);
for(i=0;i<a;i++)
{
    c++;
    for(j=0;j<a;j++)
    {
        c++;
        c++;
        if(ar[i]<ar[j])
        {
            k=ar[i];
            c++;
            ar[i]=ar[j];
            c++;
            ar[j]=k;
            c++;
        }
    }
    c++;
}c++;
printf("Mth Max Number : %d\n",ar[a-m]);
printf("Nth Min Number : %d\n",ar[n-1]);
printf("Mth Max Number : %d\n",ar[a-m]);
printf("Sum = : %d\n",ar[n-1]+ar[a-m]);
c++;
printf("Diff = : %d\n",ar[a-m]-ar[n-1]);
```

```
c++;  
printf("Time Complexity : %d\n",c);  
}
```



```
"C:\Users\Admin\Documents\daa27-mth & nth.exe"  
Enter no of elements :7  
Enter elements in array :  
1  
2  
4  
5  
7  
9  
1  
Enter M :3  
Enter N :2  
Mth Max Number : 7  
Nth Min Number : 2  
Mth Max Number : 7  
Sum = : 9  
Diff = : 5  
Time Complexity : 162  
Process returned 0 (0x0) execution time : 9.685 s  
Press any key to continue.
```

**5. Write a program to perform Knapsack problem for the following set of object values.,**

**Knapsack weight 100**

<b>item</b>	<b>Weight</b>	<b>Profit</b>
<b>1</b>	<b>40</b>	<b>80</b>
<b>2</b>	<b>30</b>	<b>70</b>
<b>3</b>	<b>20</b>	<b>50</b>
<b>4</b>	<b>30</b>	<b>80</b>

**Program:**

```
#include <stdio.h>
```

```
#include <stdlib.h>
```

```
#define MAX_ITEMS 100
```

```
#define MAX_WEIGHT 100
```

```
int weight[MAX_ITEMS];
```

```
int value[MAX_ITEMS];
```

```
int dp[MAX_ITEMS][MAX_WEIGHT];
```

```
int max(int a, int b) {
```

```
    return (a > b) ? a : b;
```

```
}
```

```
int knapsack(int n, int w) {
```

```
    int i, j;
```

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

```

for (j = 0; j <= w; j++) {
    if (i == 0 || j == 0) {
        dp[i][j] = 0;
    } else if (weight[i-1] <= j) {
        dp[i][j] = max(value[i-1] + dp[i-1][j-weight[i-1]], dp[i-1][j]);
    } else {
        dp[i][j] = dp[i-1][j];
    }
}
}
return dp[n][w];
}

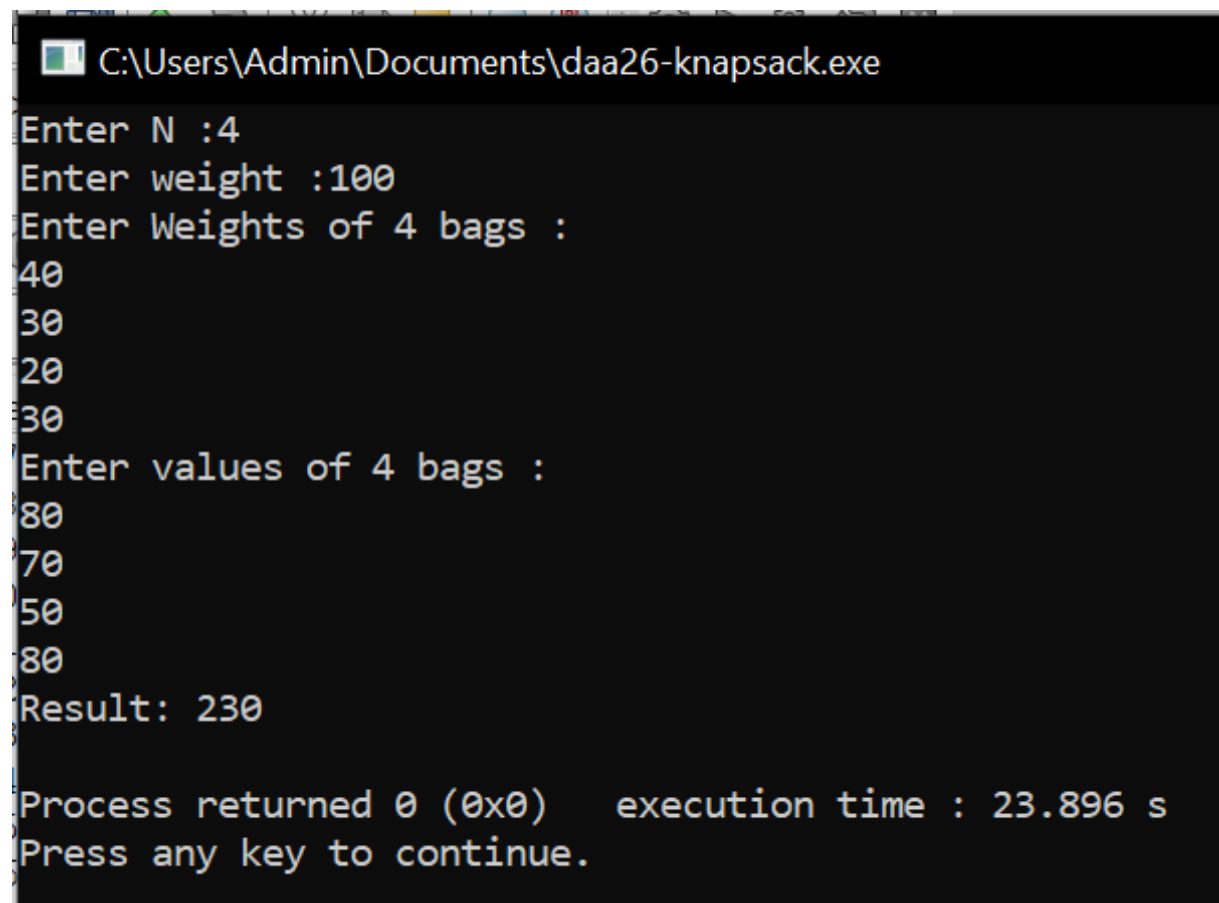
```

```

int main()
{
    int n,w,i;
    printf("Enter N :");
    scanf("%d",&n);
    printf("Enter weight :");
    scanf("%d",&w);
    printf("Enter Weights of %d bags :",n);
    for(i=0;i<n;i++)
    {
        scanf("%d",&weight[i]);
    }
    printf("Enter values of %d bags :",n);
}

```

```
for(i=0;i<n;i++)  
{  
    scanf("%d",&value[i]);  
}  
int result = knapsack(n, w);  
printf("Result: %d\n", result);  
return 0;  
}
```



The screenshot shows a Windows command prompt window titled "C:\Users\Admin\Documents\daa26-knapsack.exe". The user has entered the following inputs: N=4, weight=100, and weights of 4 bags: 40, 30, 20, 30. The program then prompts for values of 4 bags, which are entered as 80, 70, 50, and 80. The output shows "Result: 230". At the bottom, it says "Process returned 0 (0x0) execution time : 23.896 s" and "Press any key to continue."

```
C:\Users\Admin\Documents\daa26-knapsack.exe  
Enter N :4  
Enter weight :100  
Enter Weights of 4 bags :  
40  
30  
20  
30  
Enter values of 4 bags :  
80  
70  
50  
80  
Result: 230  
  
Process returned 0 (0x0)   execution time : 23.896 s  
Press any key to continue.
```

## 6. Write a program to find a minimum spanning tree using prims technique for the given graph

### Program:

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

    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];
            }
        }
    }
    count++;
    count++;
    count++;
    printf("\n Edge \t Weight\n");
    for (i = 1; i < vertices; i++)
    printf(" %d <-> %d  %d \n", parent[i], i, g[i][parent[i]]);
    count++;
    printf(" time complexity is :%d",count);

}

int main()
{
    int i,j,g[vertices][vertices];
    for (i=0;i<5;i++)
    {
        for(j=0;j<5;j++)
        {
            scanf("%d",&g[i][j]);
        }
    }
    prim(g);
    return 0;
}

```

```
C:\Users\Admin\Documents\daa25-prims.exe

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

Edge    Weight
3 <-> 1    4
0 <-> 2    3
2 <-> 3    2
3 <-> 4    1
time complexity is :8
Process returned 0 (0x0)    execution time : 82.458 s
Press any key to continue.
```