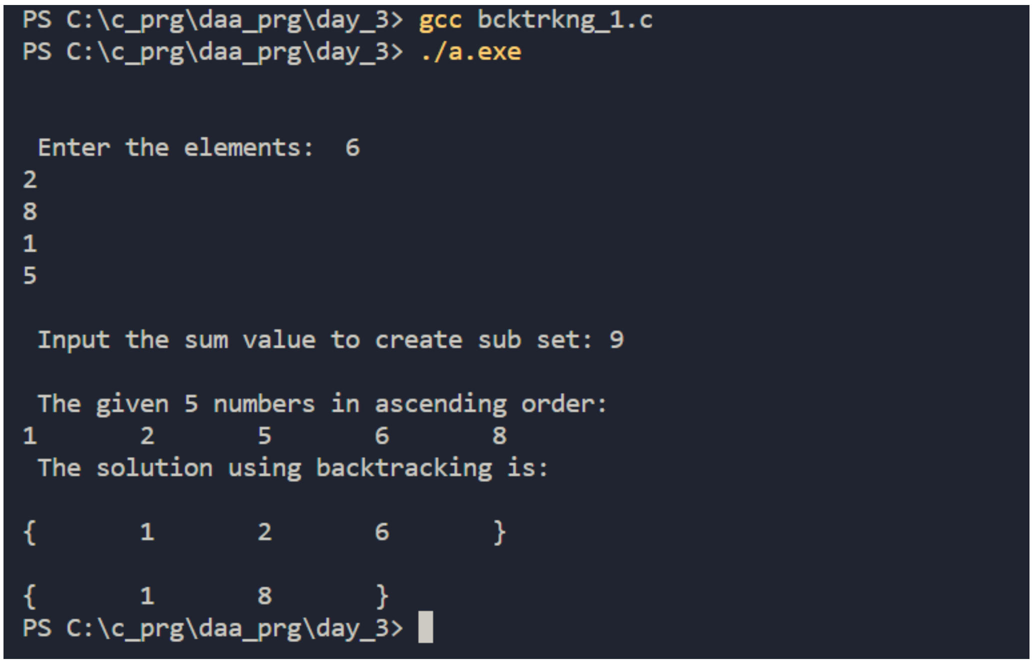
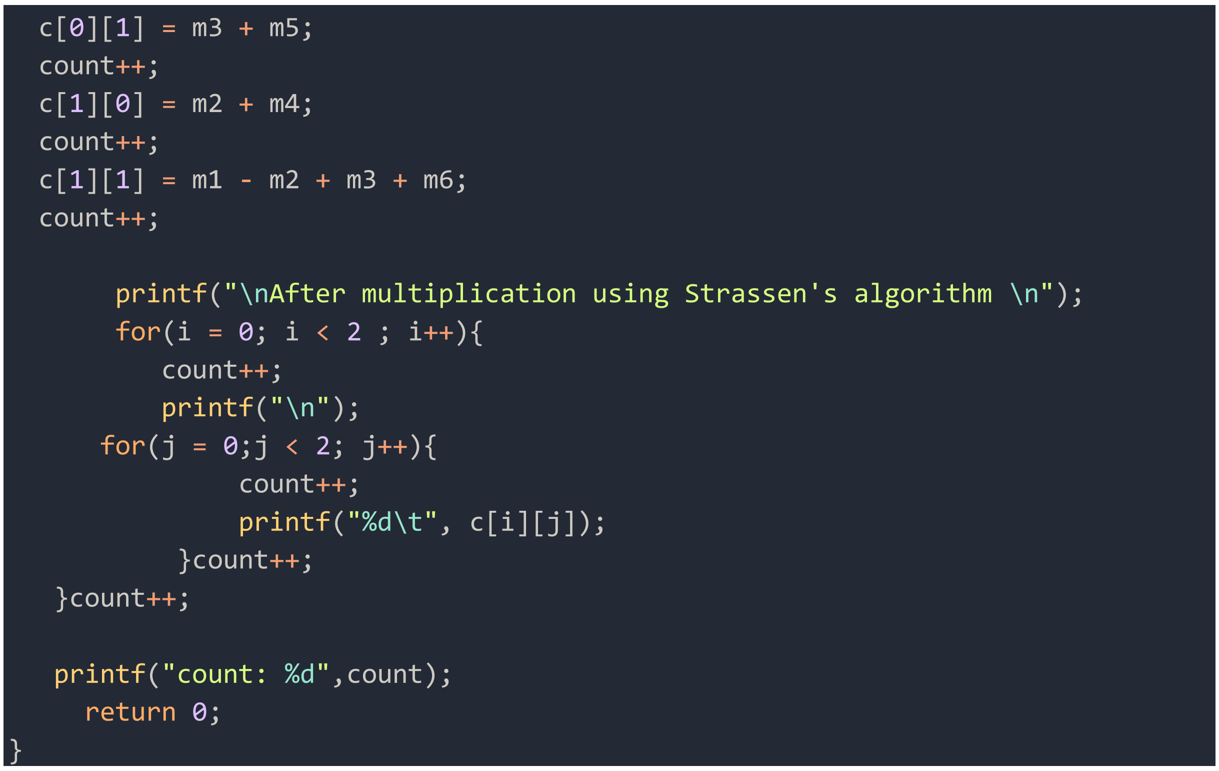
```c
//Program -1

#include <stdio.h>
#define TRUE 1
#define FALSE 0
int inc[50],w[50],sum,n;
void sumset(int i,int wt,int total);
int promising(int i,int wt,int total) {
    return(((wt+total)>=sum)&&((wt==sum)||(wt+w[i+1]<=sum)));
}
void main() {
    int i,j,n,temp,total=0;
    printf("\n Enter tot value: ");
    scanf("%d",&n);
    printf("\n Enter the elements:  ");
    for (i=0;i<n;i++) {
        scanf("%d",&w[i]);
        total+=w[i];
    }
    printf("\n Input the sum value to create sub set: ");
    scanf("%d",&sum);
    for (i=0;i<=n;i++)
      for (j=0;j<n-1;j++)
       if(w[j]>w[j+1]) {
        temp=w[j];
        w[j]=w[j+1];
        w[j+1]=temp;
    }
    printf("\n The given %d numbers in ascending order:\n",n);
    for (i=0;i<n;i++)
      printf("%d \t",w[i]);
    if((total<sum))
      printf("\n Subset construction is not possible"); else {
        for (i=0;i<n;i++)
           inc[i]=0;
        printf("\n The solution using backtracking is:\n");
        sumset(-1,0,total);
    }

}
void sumset(int i,int wt,int total) {
    int j;
    if(promising(i,wt,total)) {
        if(wt==sum) {
            printf("\n{\t");
            for (j=0;j<=i;j++)
                if(inc[j])
                 printf("%d\t",w[j]);
            printf("}\n");
        } else {
            inc[i+1]=TRUE;
            sumset(i+1,wt+w[i+1],total-w[i+1]);
            inc[i+1]=FALSE;
            sumset(i+1,wt,total-w[i+1]);
        }
    }
}
```

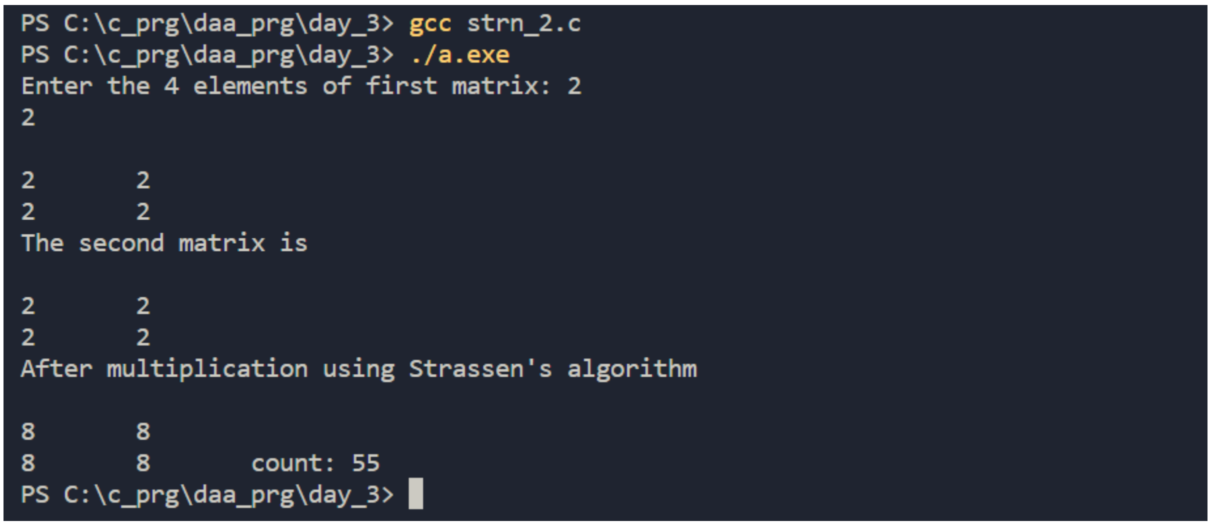


```
PS C:\c_prg\daa_prg\day_3> gcc bcktrkng_1.c
PS C:\c_prg\daa_prg\day_3> ./a.exe


 Enter the elements:  6
2
8
1
5

 Input the sum value to create sub set: 9

 The given 5 numbers in ascending order:
1       2       5       6       8
 The solution using backtracking is:

{       1       2       6       }

{       1       8       }
PS C:\c_prg\daa_prg\day_3> 
```

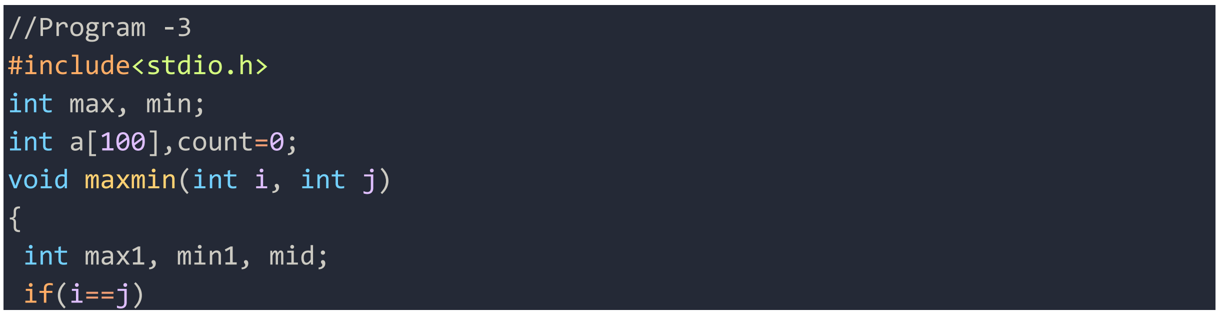
```c
//Program -2
#include<stdio.h>
int main()
{
    int a[2][2], b[2][2], c[2][2], i, j;
    int m1, m2, m3, m4 , m5, m6, m7;
    int count=0;
    count++;
    printf("Enter the 4 elements of first matrix: ");
      for(i = 0;i < 2; i++)
      {
        count++;
        for(j = 0;j < 2; j++){
               count++;
               scanf("%d", &a[i][j]);
           }count++;
      }count++;

    printf("Enter the 4 elements of second matrix: ");
      for(i = 0; i < 2; i++){
          count++;
          for(j = 0;j < 2; j++){
               count++;
               scanf("%d", &b[i][j]);
           }
       }
       count++;

      printf("\nThe first matrix is\n");
  for(i = 0; i < 2; i++){
          count++;
          printf("\n");
          for(j = 0; j < 2; j++){
               count++;
               printf("%d\t", a[i][j]);
        }count++;
      }count++;

    printf("\nThe second matrix is\n");
      for(i = 0;i < 2; i++){
          count++;
          printf("\n");
          for(j = 0;j < 2; j++){
           count++;
           printf("%d\t", b[i][j]);
       }count++;
  }count++;
  m1= (a[0][0] + a[1][1]) * (b[0][0] + b[1][1]);
  count++;
  m2= (a[1][0] + a[1][1]) * b[0][0];
  count++;
  m3= a[0][0] * (b[0][1] - b[1][1]);
  count++;
  m4= a[1][1] * (b[1][0] - b[0][0]);
  count++;
  m5= (a[0][0] + a[0][1]) * b[1][1];
  count++;
  m6= (a[1][0] - a[0][0]) * (b[0][0]+b[0][1]);
  count++;
  m7= (a[0][1] - a[1][1]) * (b[1][0]+b[1][1]);
  count++;

  c[0][0] = m1 + m4- m5 + m7;
  count++;
```



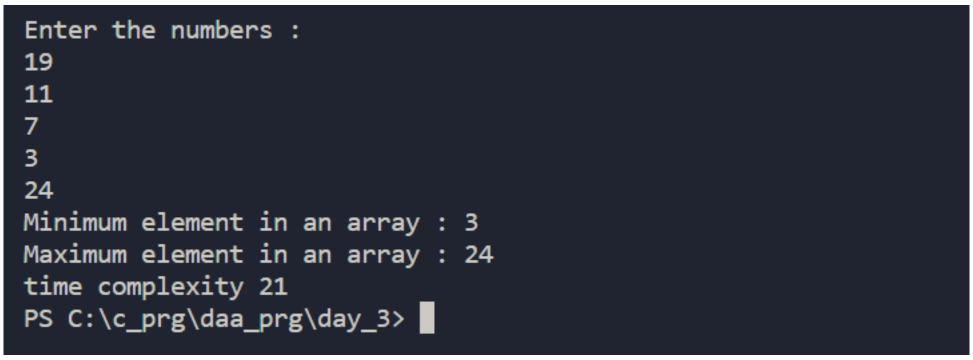
```c
  c[0][1] = m3 + m5;
  count++;
  c[1][0] = m2 + m4;
  count++;
  c[1][1] = m1 - m2 + m3 + m6;
  count++;

       printf("\nAfter multiplication using Strassen's algorithm \n");
       for(i = 0; i < 2 ; i++){
          count++;
          printf("\n");
      for(j = 0;j < 2; j++){
               count++;
               printf("%d\t", c[i][j]);
           }count++;
   }count++;

   printf("count: %d",count);
     return 0;
}
```



```
PS C:\c_prg\daa_prg\day_3> gcc strn_2.c
PS C:\c_prg\daa_prg\day_3> ./a.exe
Enter the 4 elements of first matrix: 2
2

2       2
2       2
The second matrix is

2       2
2       2
After multiplication using Strassen's algorithm

8       8
8       8       count: 55
PS C:\c_prg\daa_prg\day_3> 
```



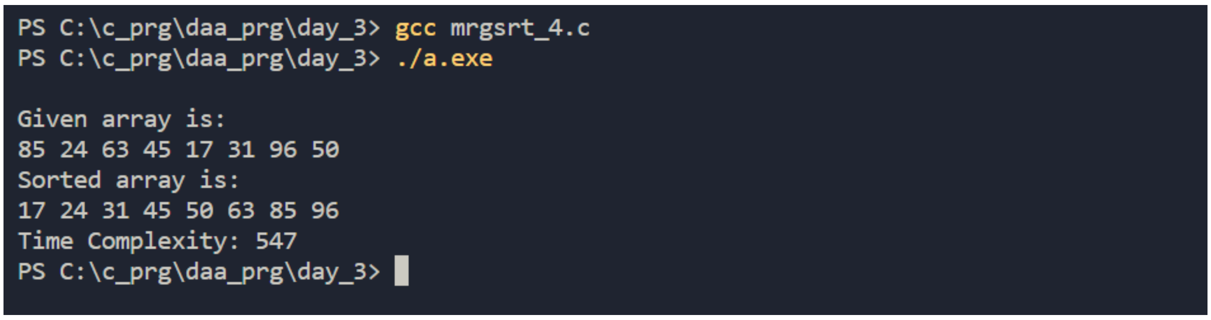
```c
//Program -3
#include<stdio.h>
int max, min;
int a[100],count=0;
void maxmin(int i, int j)
{
 int max1, min1, mid;
 if(i==j)
 {
  count++;
  max = min = a[i];
  count++;
 }
 else
 {
  if(i == j-1)
  {
   if(a[i] <a[j])
   {
    max = a[j];
    count++;
    min = a[i];
    count++;
   }
   else
   {
    max = a[i];
    count++;
    min = a[j];
    count++;
   }
  }
  else
  {
   mid = (i+j)/2;
   count++;
   maxmin(i, mid);
   count++;
   max1 = max; min1 = min;
   count++;
   maxmin(mid+1, j);
   count++;
   if(max <max1)
    max = max1;
   if(min > min1)
    min = min1;
   }
 count++;}
}

int main ()
{
 int i, num;
 printf ("\nEnter the total number of numbers : ");
 scanf ("%d",&num);
 printf ("Enter the numbers : \n");
 for (i=1;i<=num;i++)
  scanf ("%d",&a[i]);


 max = a[0];
 count++;
 min = a[0];
 count++;
 maxmin(1, num);
 count++;
 printf ("Minimum element in an array : %d\n", min);
 printf ("Maximum element in an array : %d\n", max);
 printf("time complexity %d",count);
 return 0;
}
```



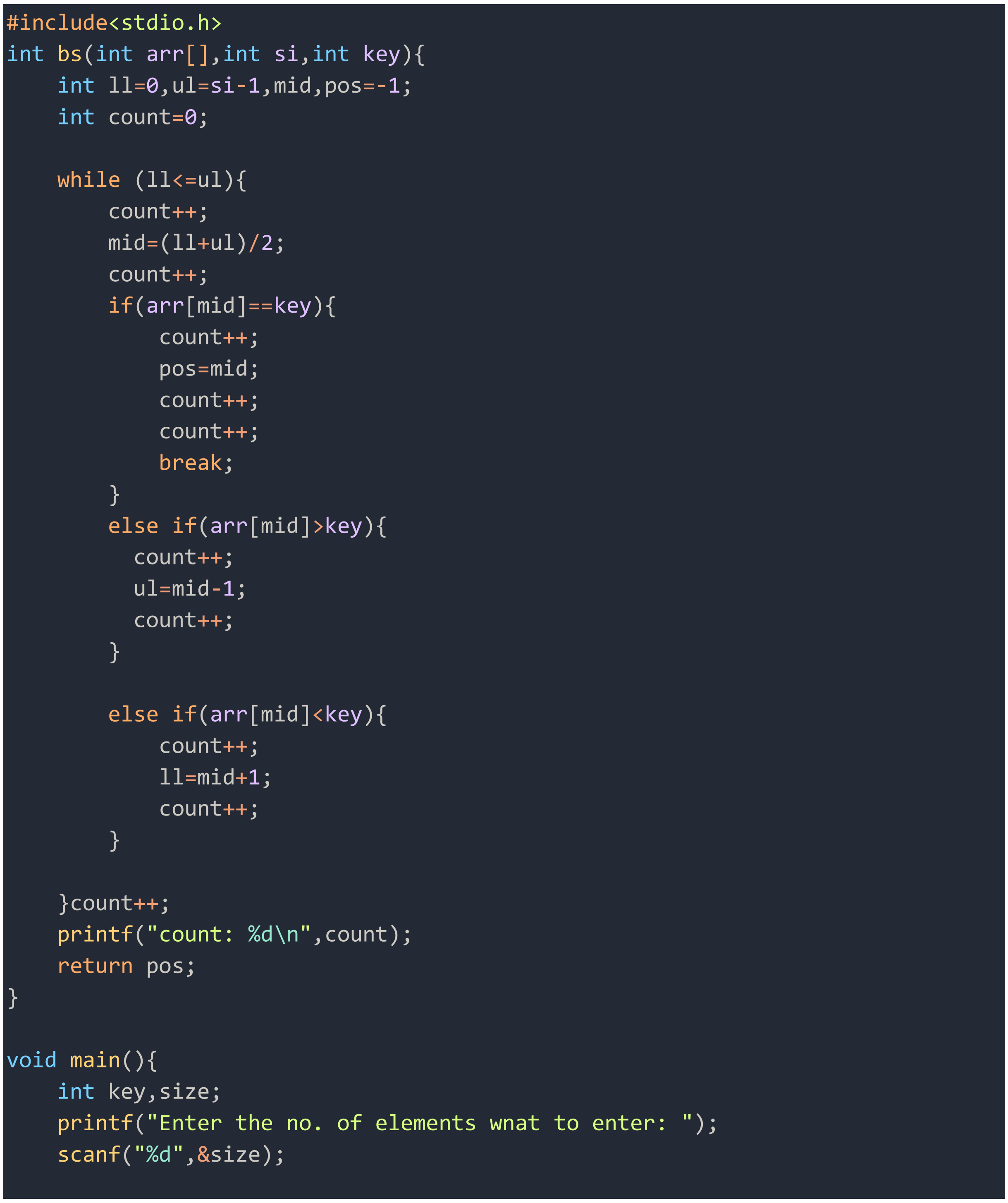
```
Enter the numbers :
19
11
7
3
24
Minimum element in an array : 3
Maximum element in an array : 24
time complexity 21
PS C:\c_prg\daa_prg\day_3>
```

```
#include<stdlib.h>
#include<stdio.h>
// Merge Function
int count=0;
int merge(int arr[], int l, int m, int r)
{
int i, j, k;
int n1 = m - l + 1;
count++;
int n2 = r - m;
count++;
int L[n1], R[n2];
for (i = 0; i < n1; i++){
    count++;
    L[i] = arr[l + i];
    count++;
}count++;

for (j = 0; j < n2; j++){
    count++;
    R[j] = arr[m + 1+ j];
    count++;
}count++;

i = 0;
count++;
j = 0;
count++;
k = l;
count++;
while (i < n1 && j < n2)
{count++;
if (L[i] <= R[j])
{count++;
arr[k] = L[i];
count++;
i++;
count++;
}
else
{count++;
arr[k] = R[j];
count++;
j++;
count++;
}
k++;
count++;
}count++;
while (i < n1)
{count++;
arr[k] = L[i];
count++;
i++;
count++;
k++;
count++;
}count++;
while (j < n2)
{count++;
arr[k] = R[j];
count++;
j++;
count++;
k++;
count++;
}count++;
```

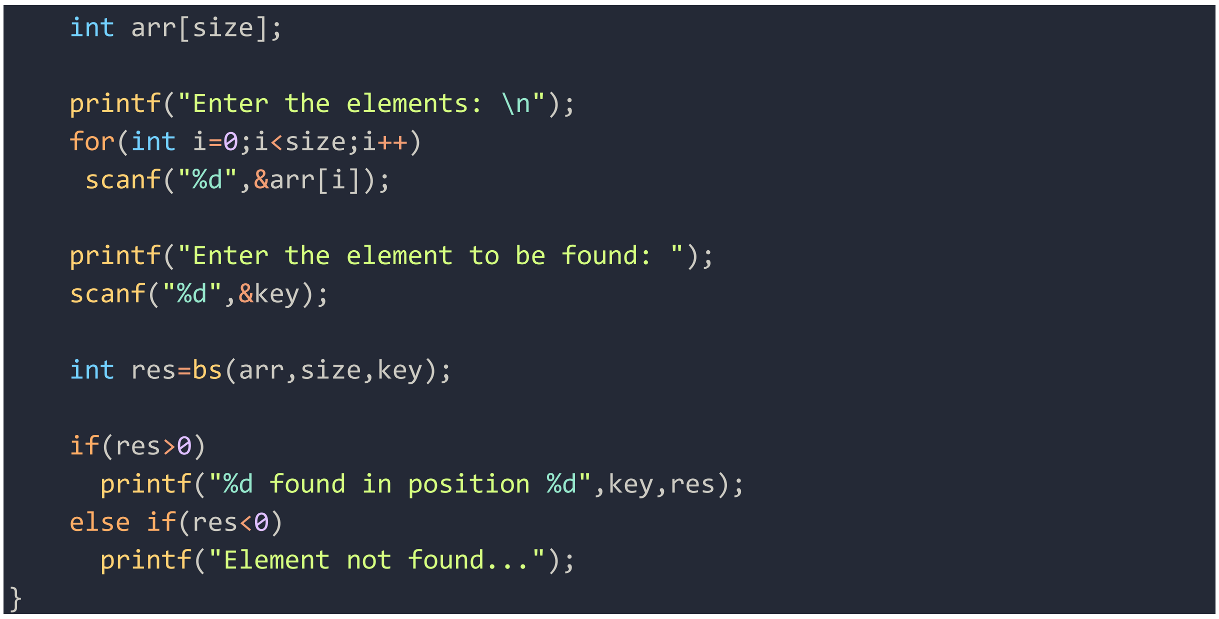
```c
return count;
}

int mergeSort(int arr[], int l, int r)
{
if (l < r)
{count++;
int m = l+(r-l)/2;
count++;
mergeSort(arr, l, m);
count++;
mergeSort(arr, m+1, r);
count++;
int res=merge(arr, l, m, r);
count++;
return res+count;
}
}

int printArray(int A[], int size)
{
int i,count=0;
printf("\n");
for (i=0; i < size; i++){
    count++;
    printf("%d ", A[i]);
}count++;

return count;

}

int main()
{
int arr[] = {85, 24, 63, 45, 17, 31, 96, 50};
int arr_size = sizeof(arr)/sizeof(arr[0]);
int res;
printf("\nGiven array is: ");
res+=printArray(arr, arr_size);
res+=mergeSort(arr, 0, arr_size - 1);
printf("\nSorted array is: ");
res+=printArray(arr, arr_size);
printf("\nTime Complexity: %d",res);
return 0;
}
```



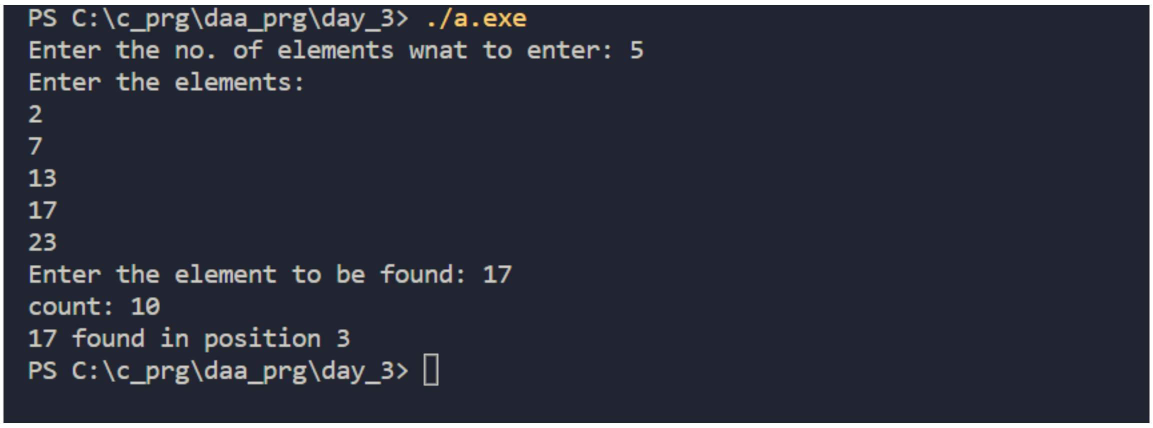
```
PS C:\c_prg\daa_prg\day_3> gcc mrgsrt_4.c
PS C:\c_prg\daa_prg\day_3> ./a.exe

Given array is:
85 24 63 45 17 31 96 50
Sorted array is:
17 24 31 45 50 63 85 96
Time Complexity: 547
PS C:\c_prg\daa_prg\day_3> 
```



```
#include<stdio.h>
int bs(int arr[],int si,int key){
    int ll=0,ul=si-1,mid,pos=-1;
    int count=0;

    while (ll<=ul){
        count++;
        mid=(ll+ul)/2;
        count++;
        if(arr[mid]==key){
            count++;
            pos=mid;
            count++;
            count++;
            break;
        }
        else if(arr[mid]>key){
          count++;
          ul=mid-1;
          count++;
        }

        else if(arr[mid]<key){
            count++;
            ll=mid+1;
            count++;
        }

    }count++;
    printf("count: %d\n",count);
    return pos;
}

void main(){
    int key,size;
    printf("Enter the no. of elements wnat to enter: ");
    scanf("%d",&size);
```



```c
    int arr[size];

    printf("Enter the elements: \n");
    for(int i=0;i<size;i++)
     scanf("%d",&arr[i]);

    printf("Enter the element to be found: ");
    scanf("%d",&key);

    int res=bs(arr,size,key);

    if(res>0)
      printf("%d found in position %d",key,res);
    else if(res<0)
      printf("Element not found...");
}
```



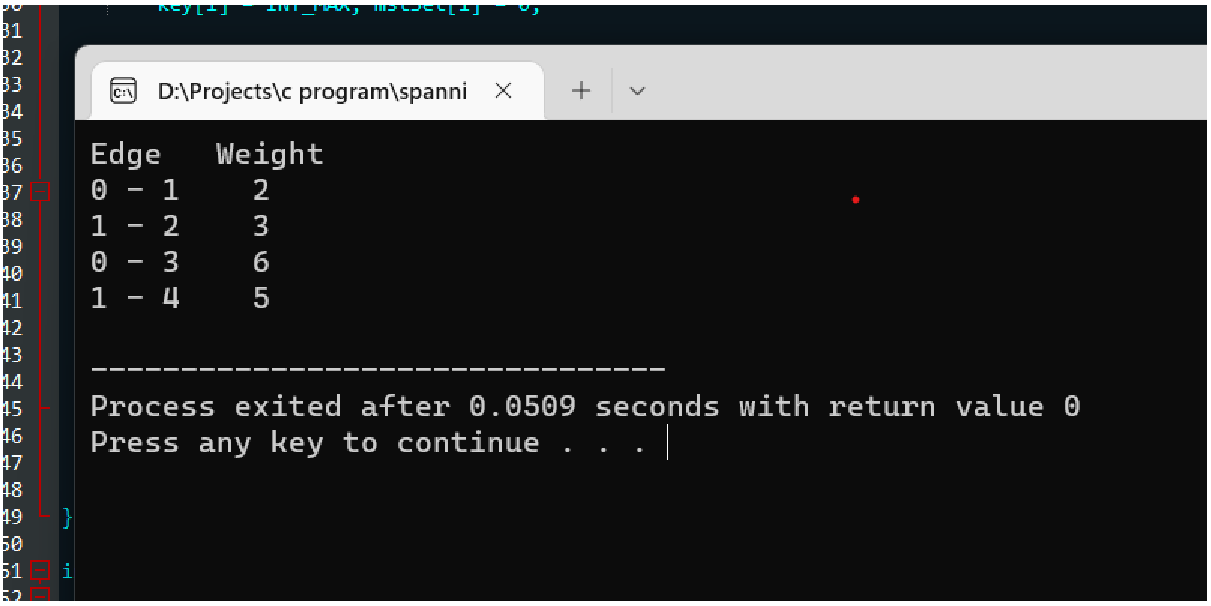
```
PS C:\c_prg\daa_prg\day_3> ./a.exe
Enter the no. of elements wnat to enter: 5
Enter the elements:
2
7
13
17
23
Enter the element to be found: 17
count: 10
17 found in position 3
PS C:\c_prg\daa_prg\day_3> 
```

PROGRAM 6

```c
#include <stdio.h>

#include <limits.h>


#define V 5


int minKey(int key[], int mstSet[]) {

   int min = INT_MAX, min_index;

   int v;
```

```
    for (v = 0; v < V; v++)
        if (mstSet[v] == 0 && key[v] < min)
            min = key[v], min_index = v;

    return min_index;
}

int printMST(int parent[], int n, int graph[V][V]) {
    int i;
    printf("Edge   Weight\n");
    for (i = 1; i < V; i++)
        printf("%d - %d    %d \n", parent[i], i, graph[i][parent[i]]);
}

void primMST(int graph[V][V]) {
    int parent[V]; // Array to store constructed MST
    int key[V], i, v, count; // Key values used to pick minimum weight edge in cut
    int mstSet[V]; // To represent set of vertices not yet included in MST

    // Initialize all keys as INFINITE
    for (i = 0; i < V; i++)
        key[i] = INT_MAX, mstSet[i] = 0;

    // Always include first 1st vertex in MST.
    key[0] = 0; // Make key 0 so that this vertex is picked as first vertex
    parent[0] = -1; // First node is always root of MST

    // The MST will have V vertices
    for (count = 0; count < V - 1; count++) {
        int u = minKey(key, mstSet);
        mstSet[u] = 1;
        for (v = 0; v < V; v++)

            if (graph[u][v] && mstSet[v] == 0 && graph[u][v] < key[v])
                parent[v] = u, key[v] = graph[u][v];
    }

    // print the constructed MST
    printMST(parent, V, graph);
}

int main() {
    /* Let us create the following graph
     2   3
    (0)--(1)--(2)
     |   / \   |
    6| 8/   \5 |7
     | /     \ |
    (3)-------(4)
     9         */
    int graph[V][V] = { { 0, 2, 0, 6, 0 }, { 2, 0, 3, 8, 5 },
            { 0, 3, 0, 0, 7 }, { 6, 8, 0, 0, 9 }, { 0, 5, 7, 9, 0 }, };

    primMST(graph);

    return 0;
}
```

OUTPUT



```
Edge   Weight
0 - 1    2
1 - 2    3
0 - 3    6
1 - 4    5

--------------------------------
Process exited after 0.0509 seconds with return value 0
Press any key to continue . . .
```