SCHOOL OF COMPUTER SCIENCE AND ENGINEERING
Lab Assignment - I, Fall Semester 2021-22

Name : Arya Dubey   Reg no:20BCE0908

Faculty: Professor Srivani A

Course Code : CSE2012   Slot : L3 + L4

Course Name: Design and Analysis of Algorithm

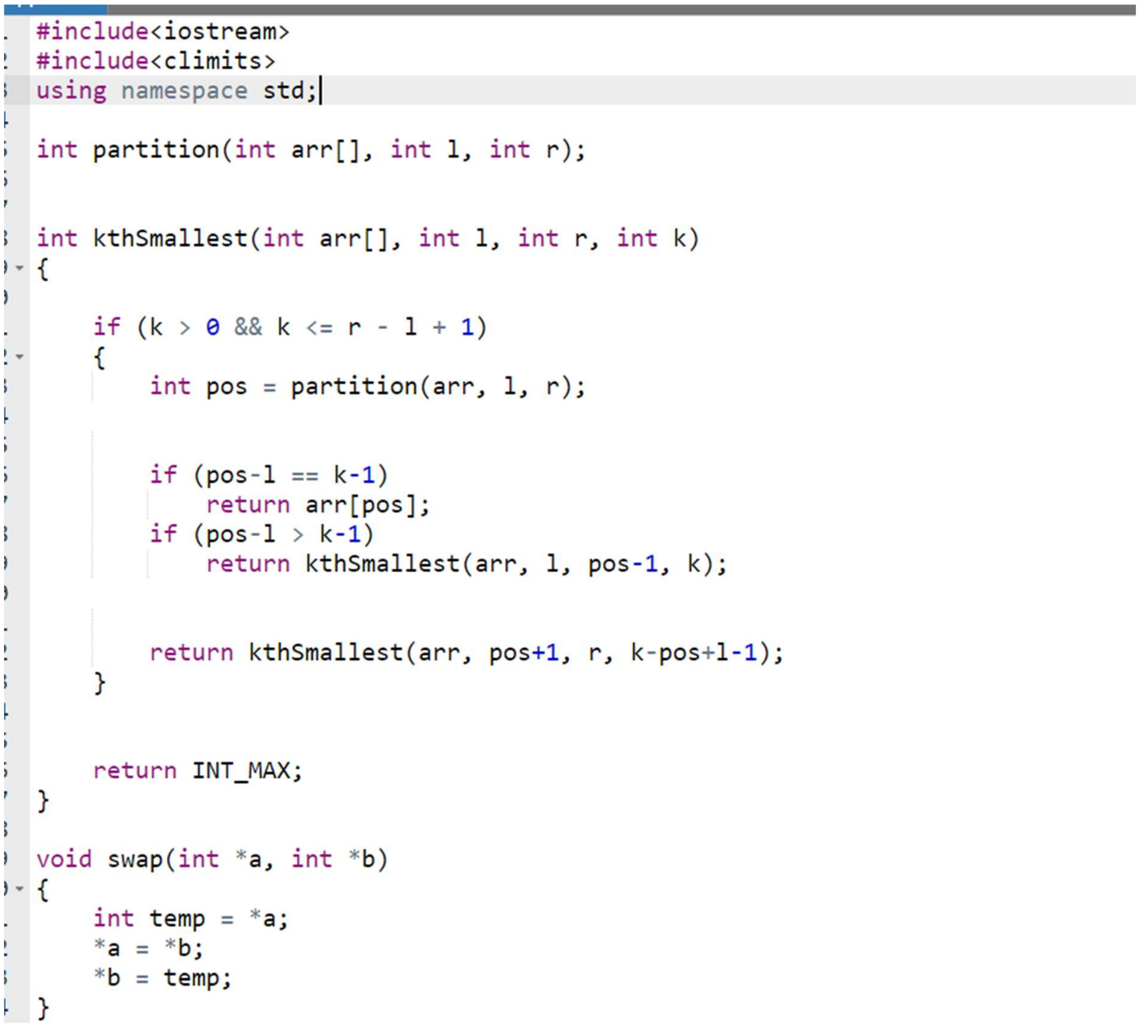
---

1. To find the kth smallest or kth largest element in a list of elements without sorting the elements.

```
#include<iostream>
#include<climits>
using namespace std;

int partition(int arr[], int l, int r);



int kthSmallest(int arr[], int l, int r, int k)
{


    if (k > 0 && k <= r - l + 1)
    {
        int pos = partition(arr, l, r);



        if (pos-l == k-1)
```

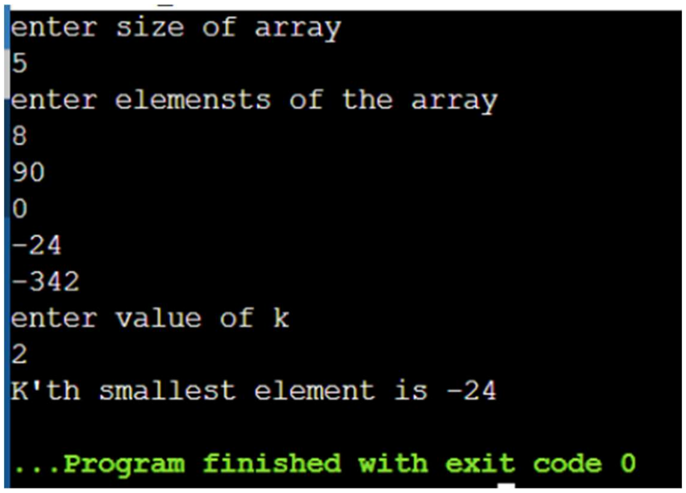
```
            return arr[pos];
        if (pos-l > k-1)
            return kthSmallest(arr, l, pos-1, k);



        return kthSmallest(arr, pos+1, r, k-pos+l-1);
    }



    return INT_MAX;
}

void swap(int *a, int *b)
{
    int temp = *a;
    *a = *b;
    *b = temp;
}



int partition(int arr[], int l, int r)
{
    int x = arr[r], i = l;
    for (int j = l; j <= r - 1; j++)
    {
        if (arr[j] <= x)
        {
            swap(&arr[i], &arr[j]);
            i++;
        }
    }
    swap(&arr[i], &arr[r]);
    return i;
}
int main()
{
    int n;
    cout<<"enter size of array"<<endl;
    cin>>n;
    int arr[n];
    cout<<"enter elemensts of the array"<<endl;
    for(int i=0;i<n;i++)
    cin>>arr[i];
    int k;
    cout<<"enter value of k"<<endl;
    cin>>k;
    cout << "K'th smallest element is " << kthSmallest(arr, 0, n-1, k);
    return 0;
}
```

Code snapshot:



```
#include<iostream>
#include<climits>
using namespace std;

int partition(int arr[], int l, int r);


int kthSmallest(int arr[], int l, int r, int k)
{

    if (k > 0 && k <= r - l + 1)
    {
        int pos = partition(arr, l, r);


        if (pos-l == k-1)
            return arr[pos];
        if (pos-l > k-1)
            return kthSmallest(arr, l, pos-1, k);


        return kthSmallest(arr, pos+1, r, k-pos+l-1);
    }


    return INT_MAX;
}

void swap(int *a, int *b)
{
    int temp = *a;
    *a = *b;
    *b = temp;
}
```

```
int partition(int arr[], int l, int r)
{
    int x = arr[r], i = l;
    for (int j = l; j <= r - 1; j++)
    {
        if (arr[j] <= x)
        {
            swap(&arr[i], &arr[j]);
            i++;
        }
    }
    swap(&arr[i], &arr[r]);
    return i;
}
int main()
{
    int n;
    cout<<"enter size of array"<<endl;
    cin>>n;
    int arr[n];
    cout<<"enter elemensts of the array"<<endl;
    for(int i=0;i<n;i++)
    cin>>arr[i];
    int k;
    cout<<"enter value of k"<<endl;
    cin>>k;
    cout << "K'th smallest element is " << kthSmallest(arr, 0, n-1, k);
    return 0;
}
```



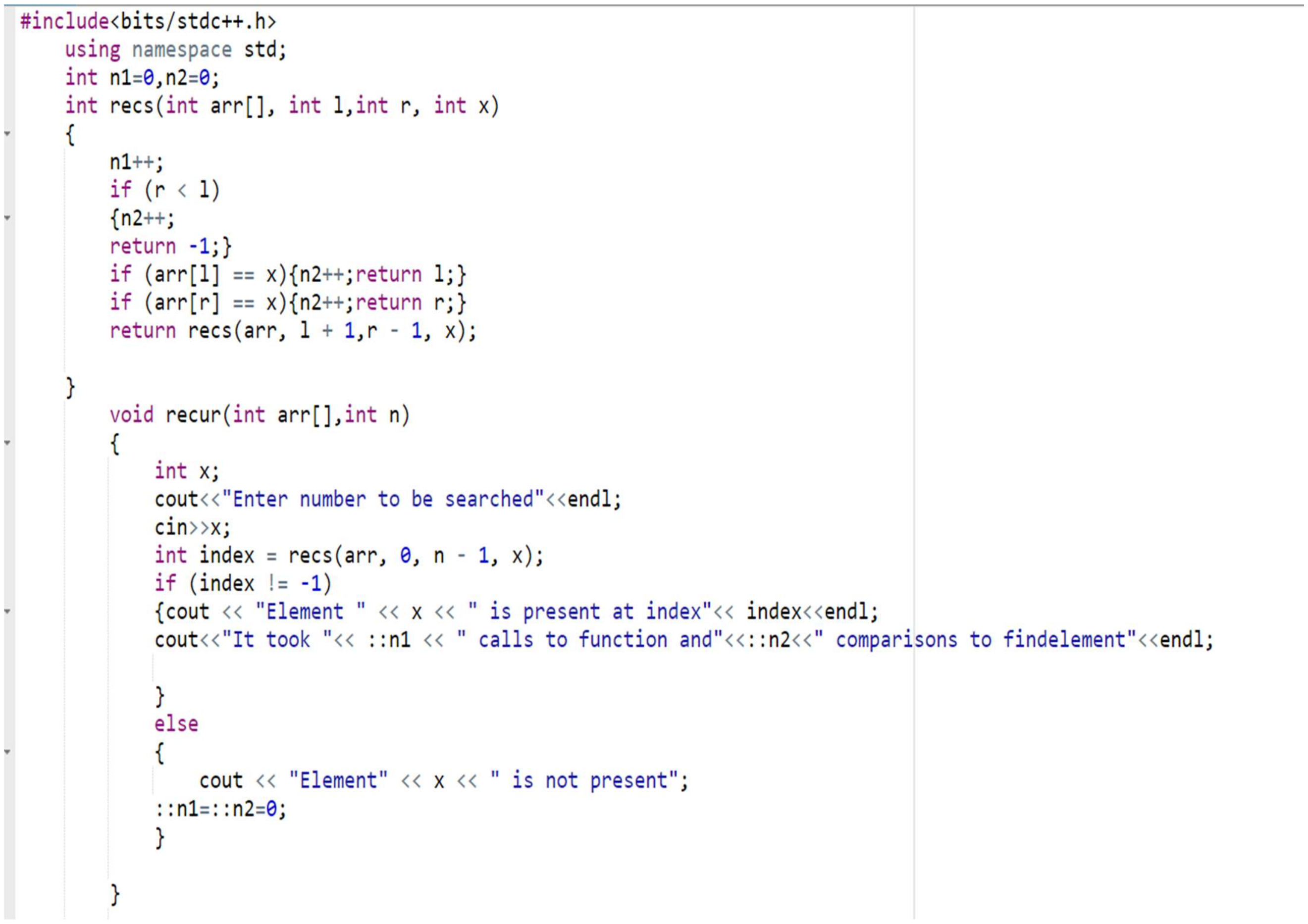
```
enter size of array
5
enter elemensts of the array
8
90
0
-24
-342
enter value of k
2
K'th smallest element is -24

...Program finished with exit code 0
```

2. To search for an element using recursive linear or binary search based on the option given by the user. The number of comparisons taken to find the element also must be found.

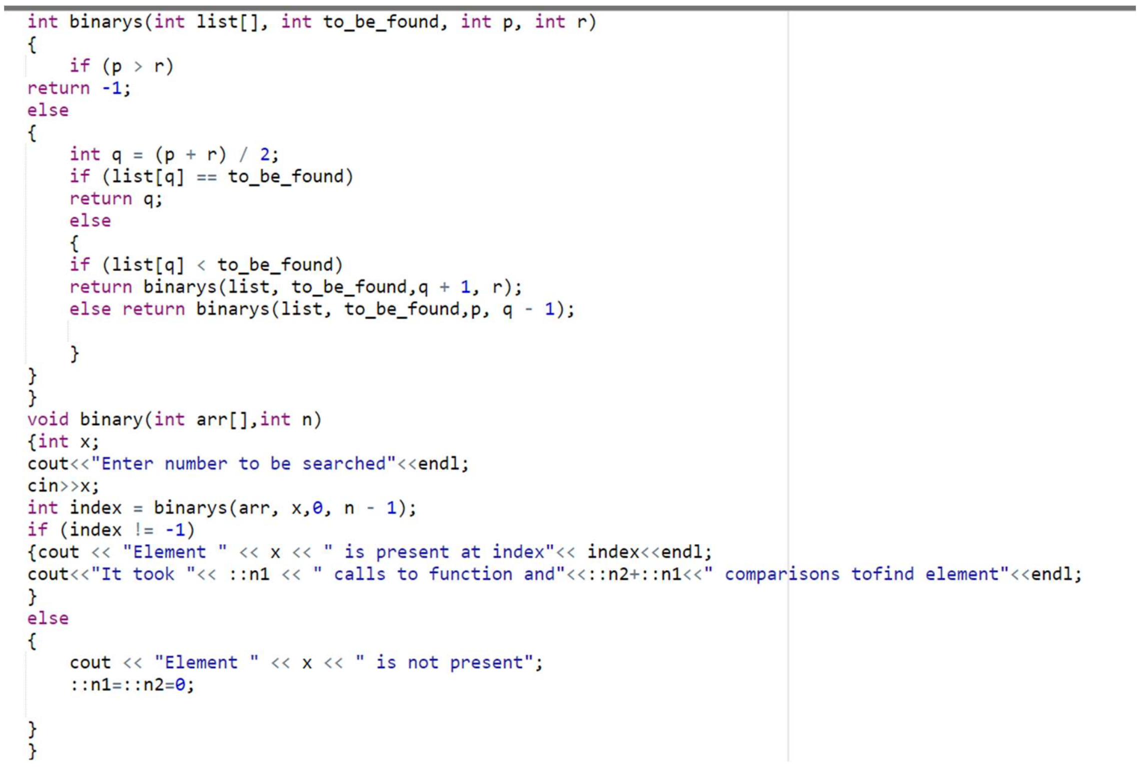
```
#include<bits/stdc++.h>
    using namespace std;
    int n1=0,n2=0;
    int recs(int arr[], int l,int r, int x)
    {
        n1++;
        if (r < l)
        {n2++;
        return -1;}
        if (arr[l] == x){n2++;return l;}
        if (arr[r] == x){n2++;return r;}
        return recs(arr, l + 1,r - 1, x);

    }
        void recur(int arr[],int n)
        {
            int x;
            cout<<"Enter number to be searched"<<endl;
            cin>>x;
            int index = recs(arr, 0, n - 1, x);
            if (index != -1)
            {cout << "Element " << x << " is present at index"<< index<<endl;
            cout<<"It took "<< ::n1 << " calls to function and"<<::n2<<" comparisons to
findelement"<<endl;


            }
            else
            {
                cout << "Element" << x << " is not present";
            ::n1=::n2=0;
            }


        }




        int binarys(int list[], int to_be_found, int p, int r)
        {
            if (p > r)
        return -1;
        else
        {
            int q = (p + r) / 2;
            if (list[q] == to_be_found)
            return q;
            else
            {
            if (list[q] < to_be_found)
            return binarys(list, to_be_found,q + 1, r);
            else return binarys(list, to_be_found,p, q - 1);


            }
        }
        }
        void binary(int arr[],int n)
```

```
{int x;
cout<<"Enter number to be searched"<<endl;
cin>>x;
int index = binarys(arr, x,0, n - 1);
if (index != -1)
{cout << "Element " << x << " is present at index"<< index<<endl;
cout<<"It took "<< ::n1 << " calls to function and"<<::n2+::n1<<" comparisons
tofind element"<<endl;
}
else
{
    cout << "Element " << x << " is not present";
    ::n1=::n2=0;


}
}
int main()
{
    int arr[30];int n,ch;
cout <<"\n Enter size of array ";
cin >> n;
cout <<"Enter the Elements: ";
for(int i=0; i<n; i++)
cin >> arr[i];
do
{cout<<"Options:"<<endl;
cout<<"1. Recursive Linear"<<endl;
cout<<"2. Binary"<<endl;
cout<<"2. Exit"<<endl;
cout<<"Enter your Choice"<<endl;
cin>>ch;
switch(ch)
{
     case 1: recur(arr,n);
     break;
     case 2: binary(arr,n);
     break;
     case 3:exit(0);


}


}while(1);
return 0;
}
```

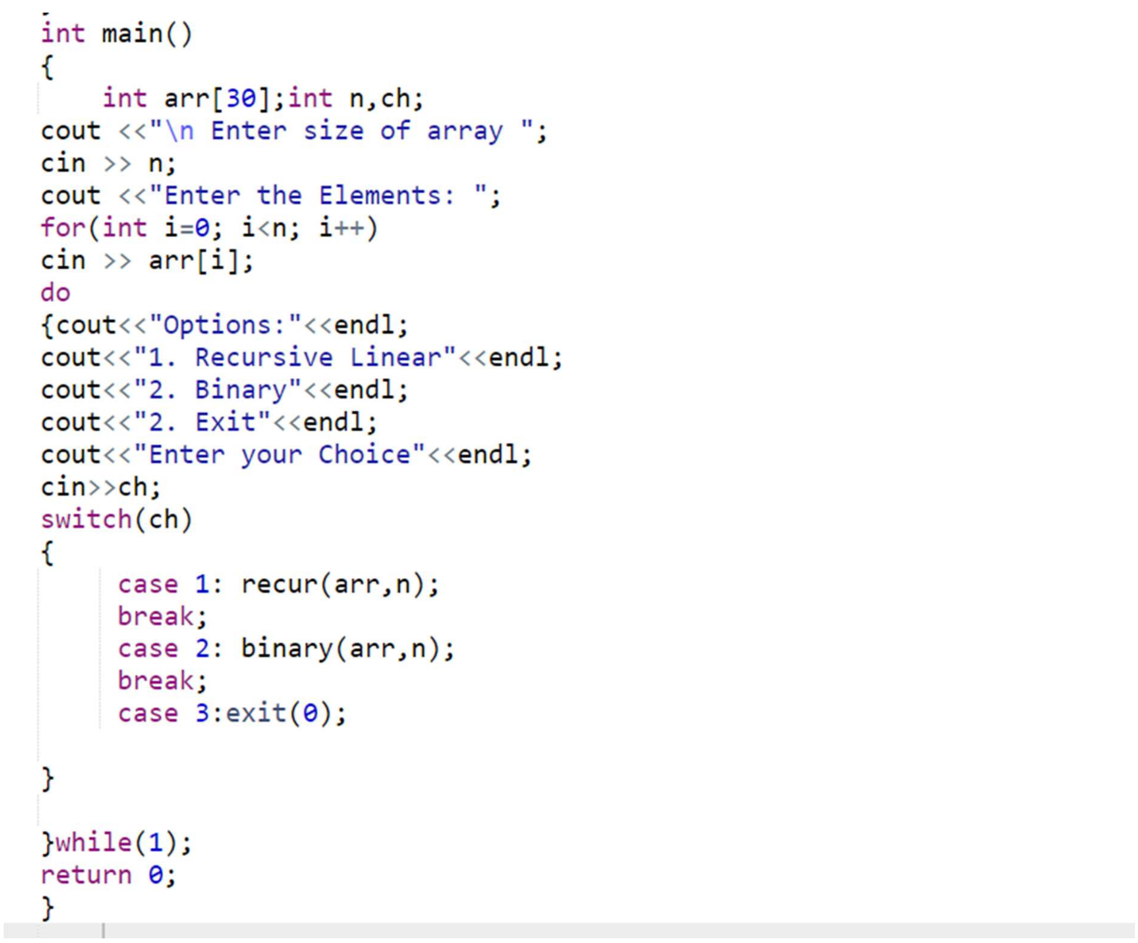
CODE SNAPSHOT:



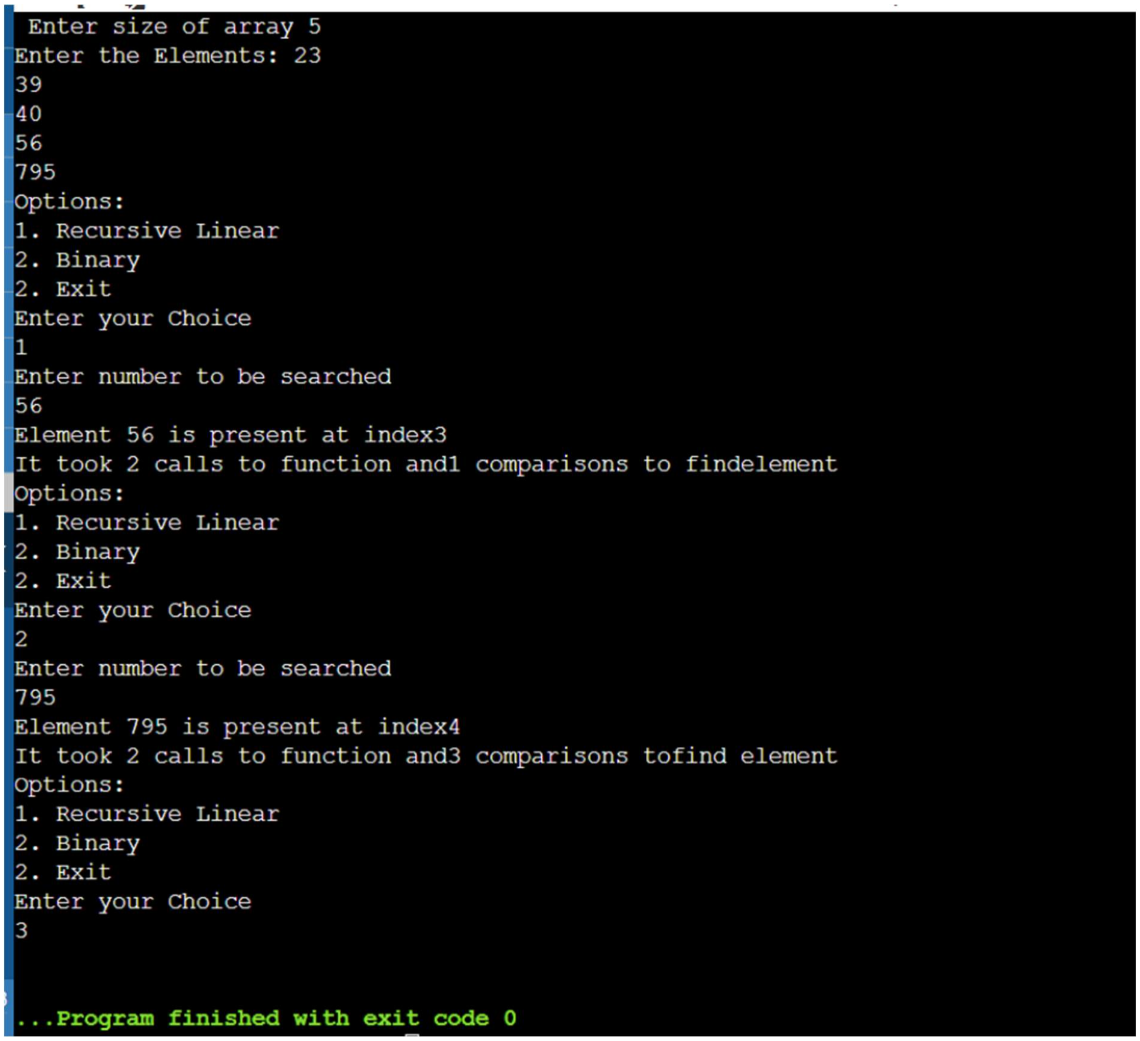
```
#include<bits/stdc++.h>
    using namespace std;
    int n1=0,n2=0;
    int recs(int arr[], int l,int r, int x)
    {
        n1++;
        if (r < l)
        {n2++;
        return -1;}
        if (arr[l] == x){n2++;return l;}
        if (arr[r] == x){n2++;return r;}
        return recs(arr, l + 1,r - 1, x);

    }
        void recur(int arr[],int n)
        {
            int x;
            cout<<"Enter number to be searched"<<endl;
            cin>>x;
            int index = recs(arr, 0, n - 1, x);
            if (index != -1)
            {cout << "Element " << x << " is present at index"<< index<<endl;
            cout<<"It took "<< ::n1 << " calls to function and"<<::n2<<" comparisons to findelement"<<endl;

            }
            else
            {
                cout << "Element" << x << " is not present";
            ::n1=::n2=0;
            }

        }
```



```
int binarys(int list[], int to_be_found, int p, int r)
{
    if (p > r)
return -1;
else
{
    int q = (p + r) / 2;
    if (list[q] == to_be_found)
    return q;
    else
    {
    if (list[q] < to_be_found)
    return binarys(list, to_be_found,q + 1, r);
    else return binarys(list, to_be_found,p, q - 1);

    }
}
}
void binary(int arr[],int n)
{int x;
cout<<"Enter number to be searched"<<endl;
cin>>x;
int index = binarys(arr, x,0, n - 1);
if (index != -1)
{cout << "Element " << x << " is present at index"<< index<<endl;
cout<<"It took "<< ::n1 << " calls to function and"<<::n2+::n1<<" comparisons tofind element"<<endl;
}
else
{
    cout << "Element " << x << " is not present";
    ::n1=::n2=0;

}
}
```



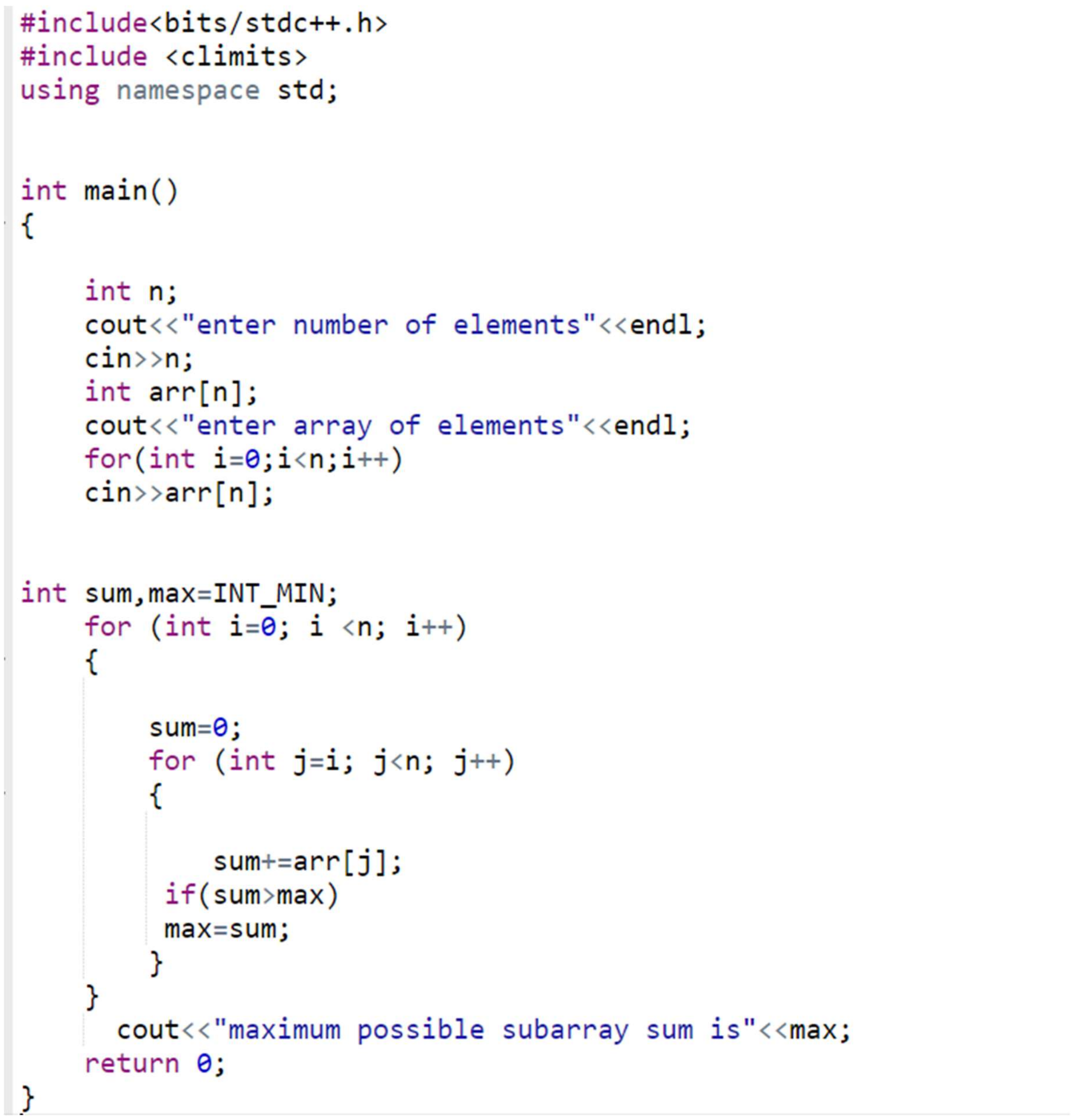
```
int main()
{
    int arr[30];int n,ch;
cout <<"\n Enter size of array ";
cin >> n;
cout <<"Enter the Elements: ";
for(int i=0; i<n; i++)
cin >> arr[i];
do
{cout<<"Options:"<<endl;
cout<<"1. Recursive Linear"<<endl;
cout<<"2. Binary"<<endl;
cout<<"2. Exit"<<endl;
cout<<"Enter your Choice"<<endl;
cin>>ch;
switch(ch)
{
     case 1: recur(arr,n);
     break;
     case 2: binary(arr,n);
     break;
     case 3:exit(0);

}

}while(1);
return 0;
}
```



```
 Enter size of array 5
Enter the Elements: 23
39
40
56
795
Options:
1. Recursive Linear
2. Binary
2. Exit
Enter your Choice
1
Enter number to be searched
56
Element 56 is present at index3
It took 2 calls to function and1 comparisons to findelement
Options:
1. Recursive Linear
2. Binary
2. Exit
Enter your Choice
2
Enter number to be searched
795
Element 795 is present at index4
It took 2 calls to function and3 comparisons tofind element
Options:
1. Recursive Linear
2. Binary
2. Exit
Enter your Choice
3


...Program finished with exit code 0
```

3. Brute Force Approach:
i) Any known problem

MAXIMUM SUBARRAY SUM:

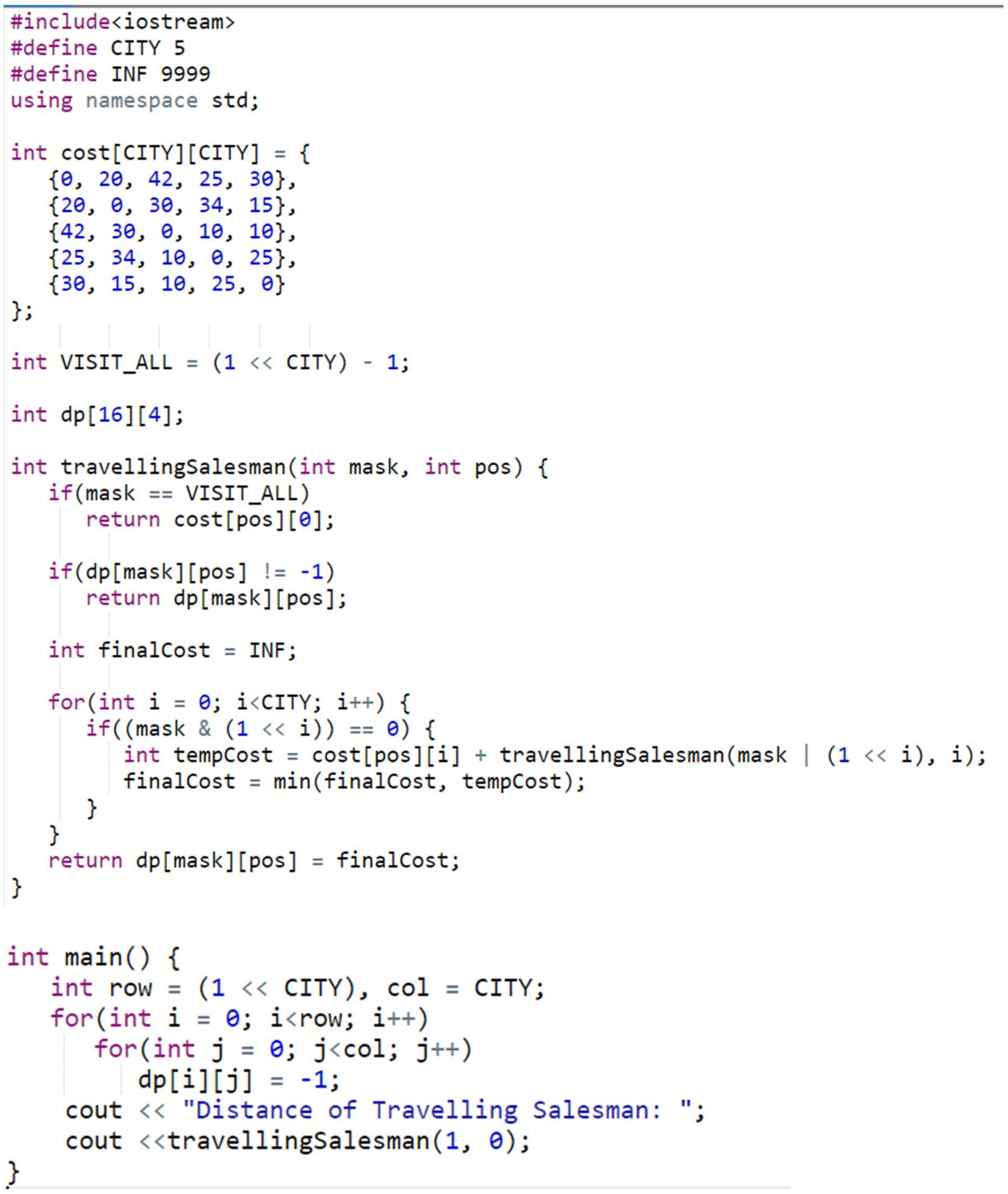


```
#include<bits/stdc++.h>
#include <climits>
using namespace std;


int main()
{

    int n;
    cout<<"enter number of elements"<<endl;
    cin>>n;
    int arr[n];
    cout<<"enter array of elements"<<endl;
    for(int i=0;i<n;i++)
    cin>>arr[n];


int sum,max=INT_MIN;
    for (int i=0; i <n; i++)
    {

        sum=0;
        for (int j=i; j<n; j++)
        {

            sum+=arr[j];
         if(sum>max)
         max=sum;
        }
    }
      cout<<"maximum possible subarray sum is"<<max;
    return 0;
}
```

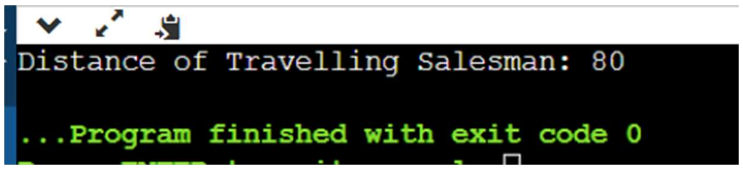
ii) Travelling Salesman Problem

```
#include<iostream>
#define CITY 5
#define INF 9999
using namespace std;

int cost[CITY][CITY] = {
   {0, 20, 42, 25, 30},
   {20, 0, 30, 34, 15},
   {42, 30, 0, 10, 10},
   {25, 34, 10, 0, 25},
   {30, 15, 10, 25, 0}
};

int VISIT_ALL = (1 << CITY) - 1;

int dp[16][4];

int travellingSalesman(int mask, int pos) {
   if(mask == VISIT_ALL)
      return cost[pos][0];

   if(dp[mask][pos] != -1)
      return dp[mask][pos];

   int finalCost = INF;

   for(int i = 0; i<CITY; i++) {
      if((mask & (1 << i)) == 0) {
         int tempCost = cost[pos][i] + travellingSalesman(mask | (1 << i), i);
         finalCost = min(finalCost, tempCost);
      }
   }
   return dp[mask][pos] = finalCost;
}

int main() {
   int row = (1 << CITY), col = CITY;
   for(int i = 0; i<row; i++)
      for(int j = 0; j<col; j++)
         dp[i][j] = -1;
    cout << "Distance of Travelling Salesman: ";
    cout <<travellingSalesman(1, 0);
}
```

CODE SNAPSHOT :



```cpp
#include<iostream>
#define CITY 5
#define INF 9999
using namespace std;

int cost[CITY][CITY] = {
   {0, 20, 42, 25, 30},
   {20, 0, 30, 34, 15},
   {42, 30, 0, 10, 10},
   {25, 34, 10, 0, 25},
   {30, 15, 10, 25, 0}
};

int VISIT_ALL = (1 << CITY) - 1;

int dp[16][4];

int travellingSalesman(int mask, int pos) {
   if(mask == VISIT_ALL)
      return cost[pos][0];

   if(dp[mask][pos] != -1)
      return dp[mask][pos];

   int finalCost = INF;

   for(int i = 0; i<CITY; i++) {
      if((mask & (1 << i)) == 0) {
         int tempCost = cost[pos][i] + travellingSalesman(mask | (1 << i), i);
         finalCost = min(finalCost, tempCost);
      }
   }
   return dp[mask][pos] = finalCost;
}

int main() {
   int row = (1 << CITY), col = CITY;
   for(int i = 0; i<row; i++)
      for(int j = 0; j<col; j++)
         dp[i][j] = -1;
    cout << "Distance of Travelling Salesman: ";
    cout <<travellingSalesman(1, 0);
}
```



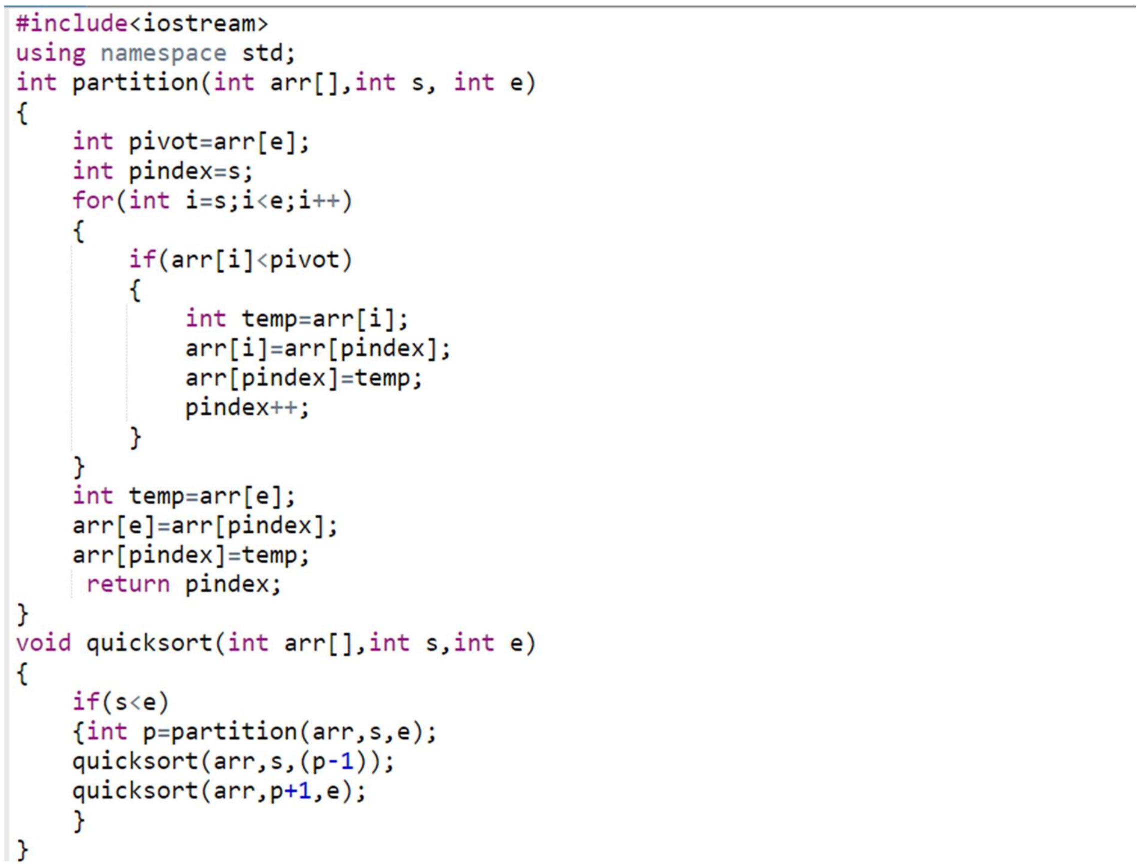
```
Distance of Travelling Salesman: 80

...Program finished with exit code 0
```

4. Divide and Conquer Approach

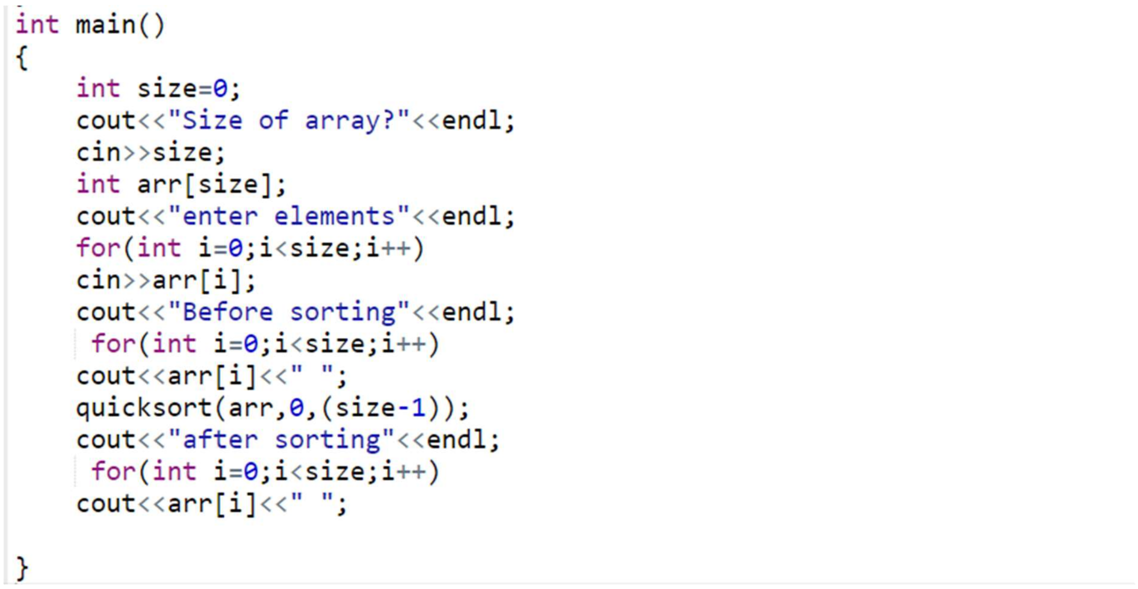
i) Quick Sort or Merge Sort

```
#include<iostream>
using namespace std;
int partition(int arr[],int s, int e)
{
    int pivot=arr[e];
    int pindex=s;
    for(int i=s;i<e;i++)
    {
        if(arr[i]<pivot)
        {
            int temp=arr[i];
            arr[i]=arr[pindex];
            arr[pindex]=temp;
            pindex++;
        }
    }
    int temp=arr[e];
    arr[e]=arr[pindex];
    arr[pindex]=temp;
     return pindex;
}
void quicksort(int arr[],int s,int e)
{
    if(s<e)
    {int p=partition(arr,s,e);
    quicksort(arr,s,(p-1));
    quicksort(arr,p+1,e);
    }
}
int main()
{
    int size=0;
    cout<<"Size of array?"<<endl;
    cin>>size;
    int arr[size];
    cout<<"enter elements"<<endl;
    for(int i=0;i<size;i++)
    cin>>arr[i];
    cout<<"Before sorting"<<endl;
     for(int i=0;i<size;i++)
    cout<<arr[i]<<" ";
    quicksort(arr,0,(size-1));
    cout<<"after sorting"<<endl;
     for(int i=0;i<size;i++)
    cout<<arr[i]<<" ";

}
```

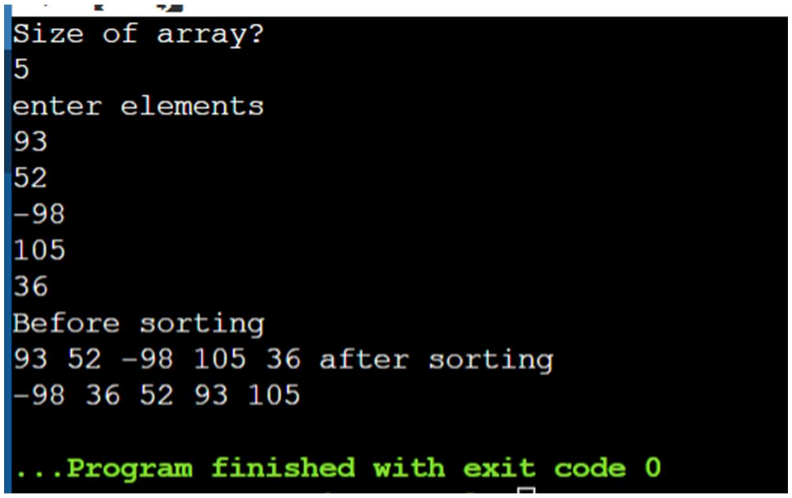
CODE SNAPSHOT:



```
#include<iostream>
using namespace std;
int partition(int arr[],int s, int e)
{
    int pivot=arr[e];
    int pindex=s;
    for(int i=s;i<e;i++)
    {
        if(arr[i]<pivot)
        {
            int temp=arr[i];
            arr[i]=arr[pindex];
            arr[pindex]=temp;
            pindex++;
        }
    }
    int temp=arr[e];
    arr[e]=arr[pindex];
    arr[pindex]=temp;
     return pindex;
}
void quicksort(int arr[],int s,int e)
{
    if(s<e)
    {int p=partition(arr,s,e);
    quicksort(arr,s,(p-1));
    quicksort(arr,p+1,e);
    }
}
```



```
int main()
{
    int size=0;
    cout<<"Size of array?"<<endl;
    cin>>size;
    int arr[size];
    cout<<"enter elements"<<endl;
    for(int i=0;i<size;i++)
    cin>>arr[i];
    cout<<"Before sorting"<<endl;
     for(int i=0;i<size;i++)
    cout<<arr[i]<<" ";
    quicksort(arr,0,(size-1));
    cout<<"after sorting"<<endl;
     for(int i=0;i<size;i++)
    cout<<arr[i]<<" ";

}
```


```
Size of array?
5
enter elements
93
52
-98
105
36
Before sorting
93 52 -98 105 36 after sorting
-98 36 52 93 105

...Program finished with exit code 0
```

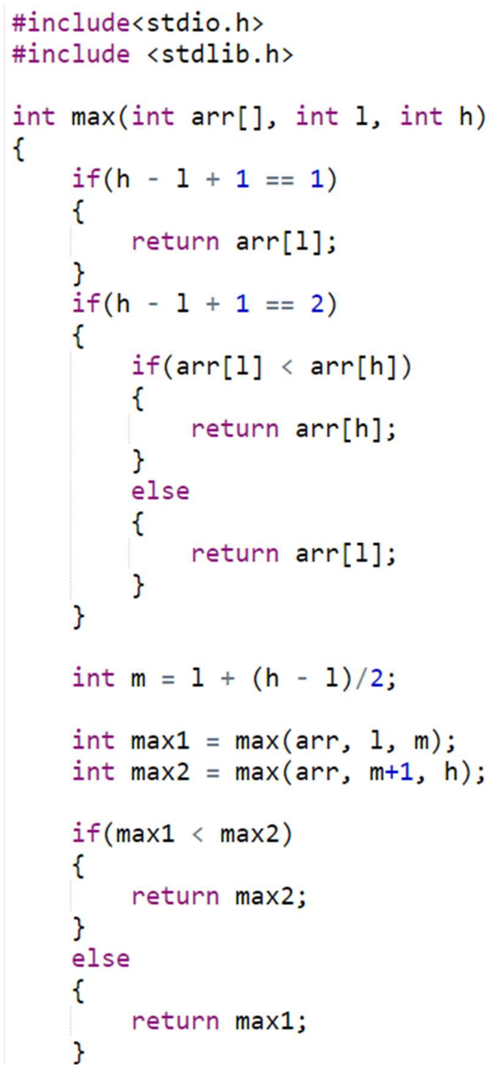
---

ii) Min and Max in an array

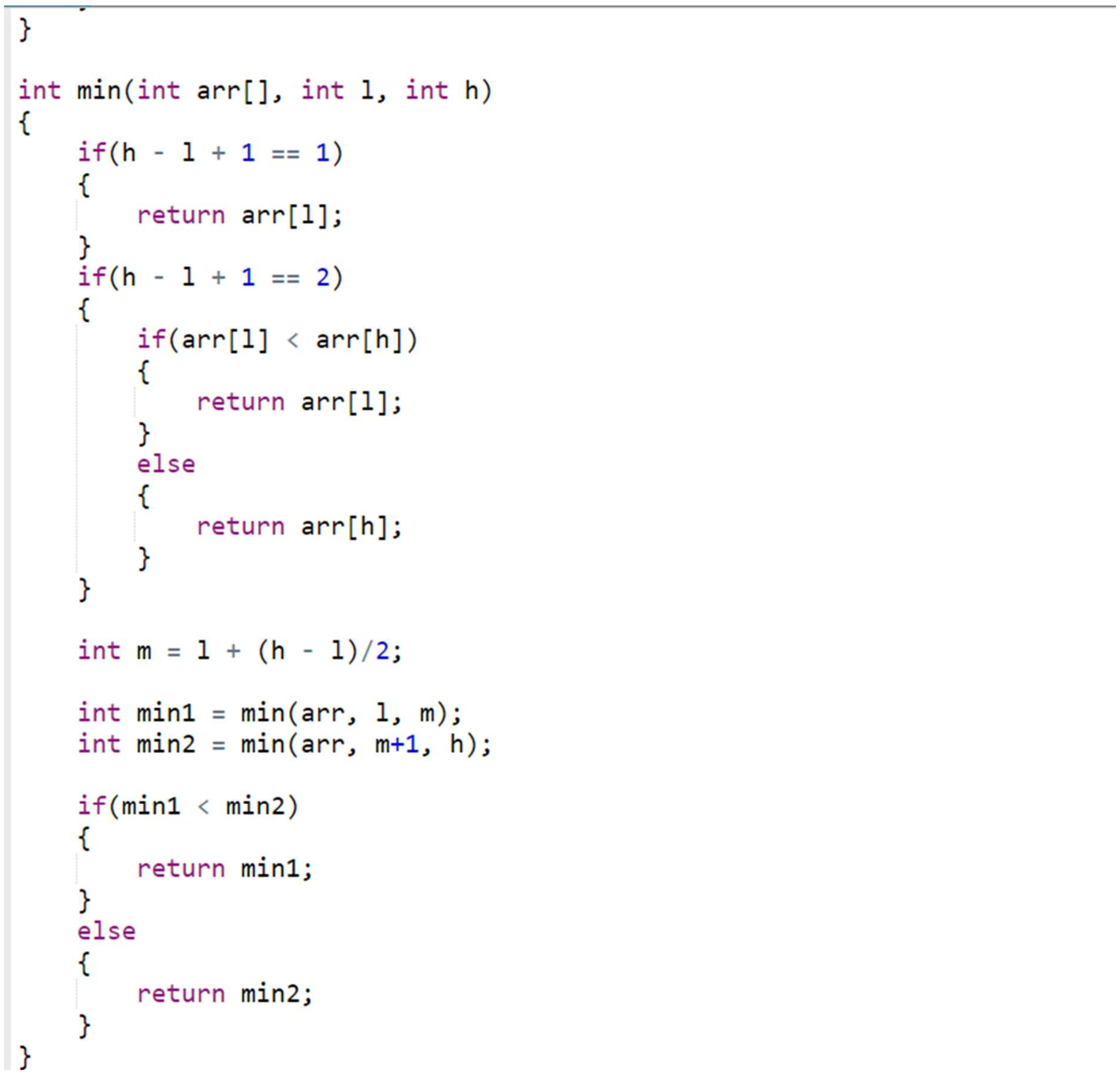
```
#include<stdio.h>
#include <stdlib.h>

int max(int arr[], int l, int h)
{
        if(h - l + 1 == 1)
        {
                return arr[l];
        }
        if(h - l + 1 == 2)
        {
                if(arr[l] < arr[h])
                {
                        return arr[h];
                }
                else
                {
                        return arr[l];
                }
        }

        int m = l + (h - l)/2;

        int max1 = max(arr, l, m);
        int max2 = max(arr, m+1, h);

        if(max1 < max2)
        {
                return max2;
        }
        else
        {
                return max1;
        }
}

int min(int arr[], int l, int h)
{
        if(h - l + 1 == 1)
        {
                return arr[l];
        }
        if(h - l + 1 == 2)
        {
                if(arr[l] < arr[h])
                {
                        return arr[l];
                }
                else
                {
                        return arr[h];
                }
        }

        int m = l + (h - l)/2;

        int min1 = min(arr, l, m);
        int min2 = min(arr, m+1, h);

        if(min1 < min2)
        {
                return min1;
        }
        else
        {
                return min2;
        }
}


int main(void) {

        printf("Enter number of elements - ");

        int n;
        scanf("%d", &n);

        int arr[n];
        for(int i = 0; i < n; i++)
```

```
        {
            printf("enter number");
                scanf("%d", &arr[i]);
        }

        printf("Maximum number - %d\n", max(arr, 0, n-1));

        printf("Minimum number - %d\n", min(arr, 0, n-1));

        return 0;
}
```

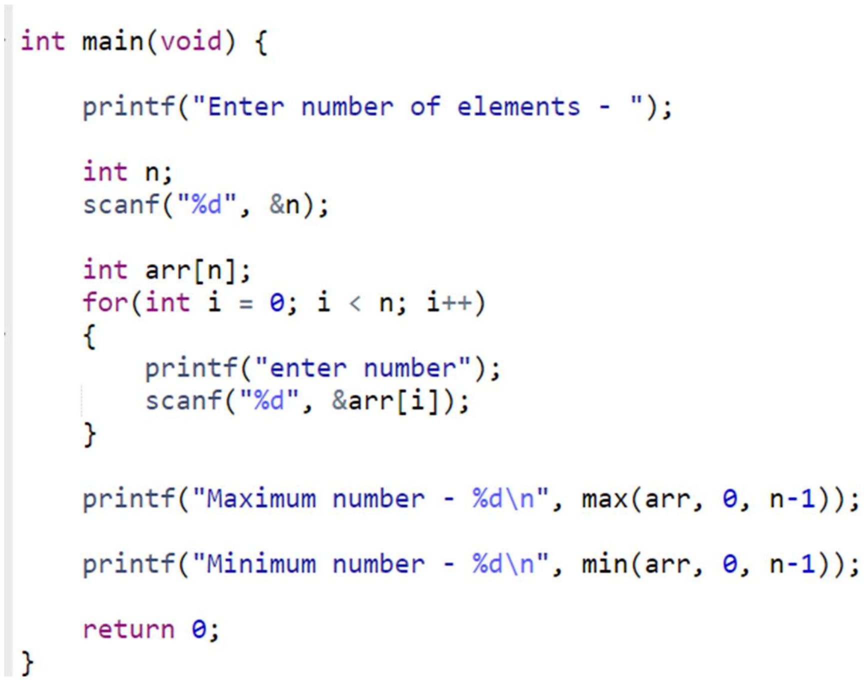
CODE SNAPSHOT:



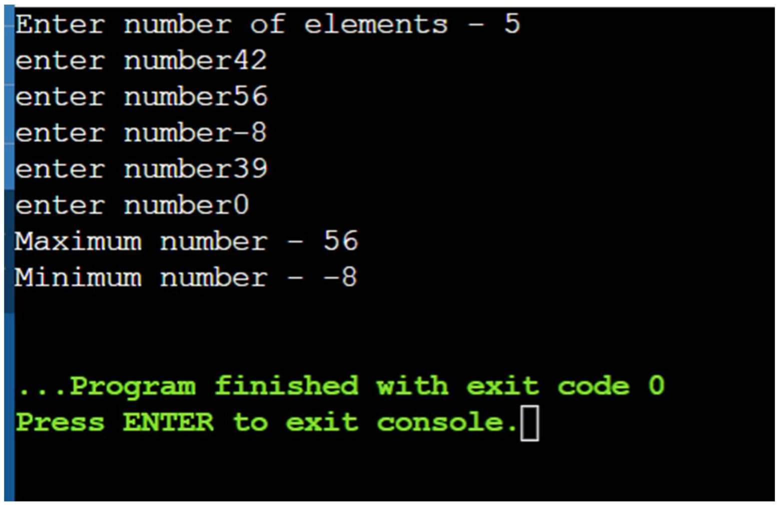
```
#include<stdio.h>
#include <stdlib.h>

int max(int arr[], int l, int h)
{
    if(h - l + 1 == 1)
    {
        return arr[l];
    }
    if(h - l + 1 == 2)
    {
        if(arr[l] < arr[h])
        {
            return arr[h];
        }
        else
        {
            return arr[l];
        }
    }

    int m = l + (h - l)/2;

    int max1 = max(arr, l, m);
    int max2 = max(arr, m+1, h);

    if(max1 < max2)
    {
        return max2;
    }
    else
    {
        return max1;
    }
```



```
}

int min(int arr[], int l, int h)
{
    if(h - l + 1 == 1)
    {
        return arr[l];
    }
    if(h - l + 1 == 2)
    {
        if(arr[l] < arr[h])
        {
            return arr[l];
        }
        else
        {
            return arr[h];
        }
    }

    int m = l + (h - l)/2;

    int min1 = min(arr, l, m);
    int min2 = min(arr, m+1, h);

    if(min1 < min2)
    {
        return min1;
    }
    else
    {
        return min2;
    }
}
```



```c
int main(void) {

    printf("Enter number of elements - ");

    int n;
    scanf("%d", &n);

    int arr[n];
    for(int i = 0; i < n; i++)
    {
        printf("enter number");
        scanf("%d", &arr[i]);
    }

    printf("Maximum number - %d\n", max(arr, 0, n-1));

    printf("Minimum number - %d\n", min(arr, 0, n-1));

    return 0;
}
```



```
Enter number of elements - 5
enter number42
enter number56
enter number-8
enter number39
enter number0
Maximum number - 56
Minimum number - -8


...Program finished with exit code 0
Press ENTER to exit console.
```