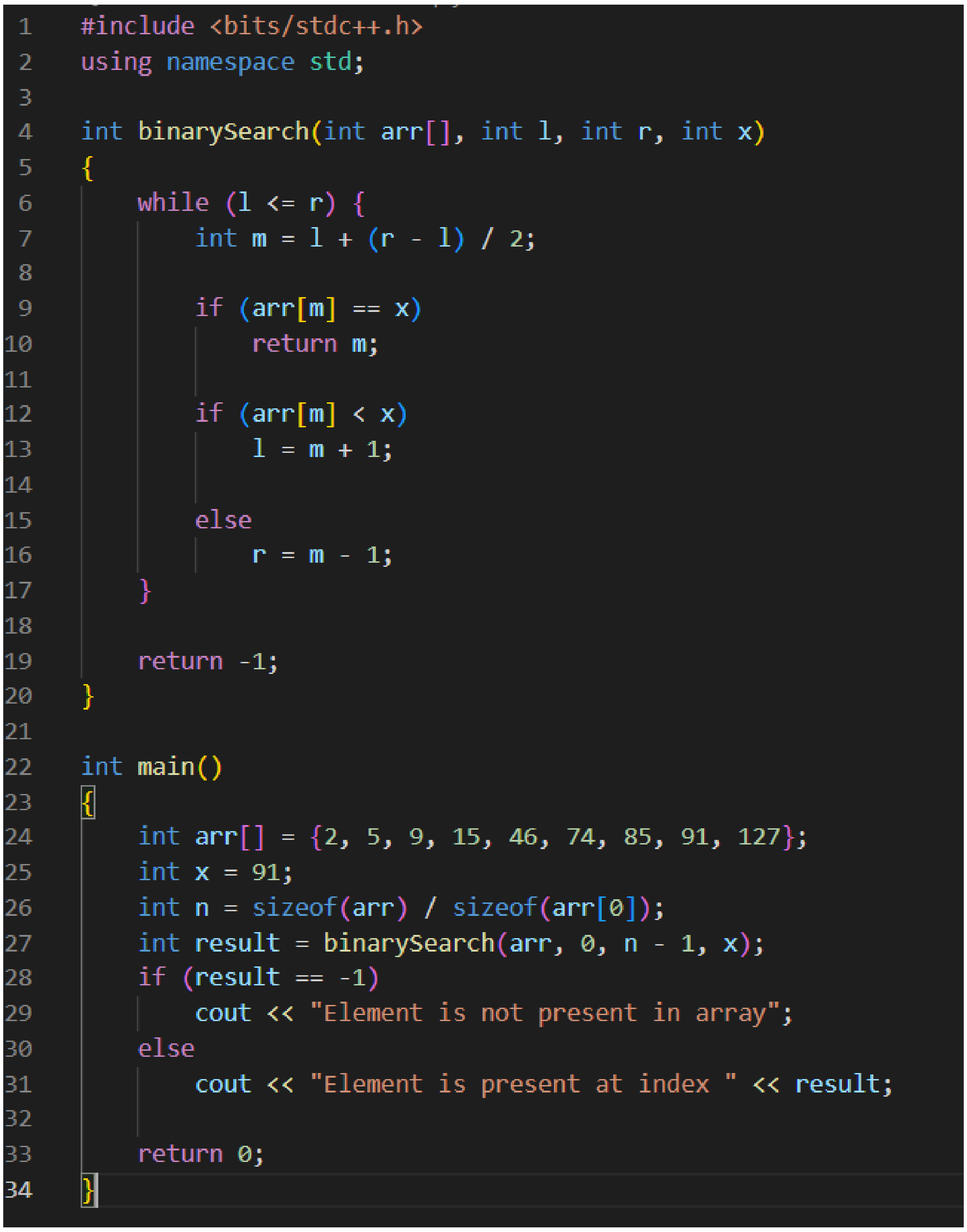
Name: Chandranshu Bhardwaj

Roll No: 102203797

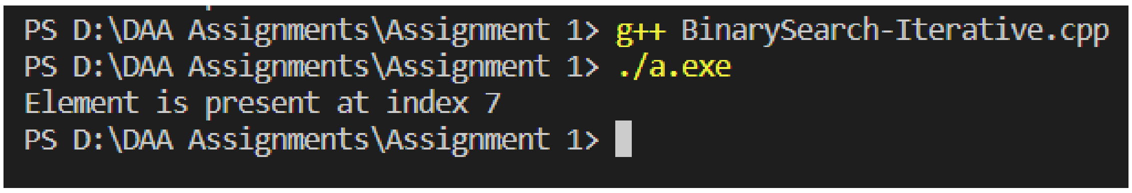
Group: 2CO-17

DAA Lab Assignment 1

Binary Search – Iterative

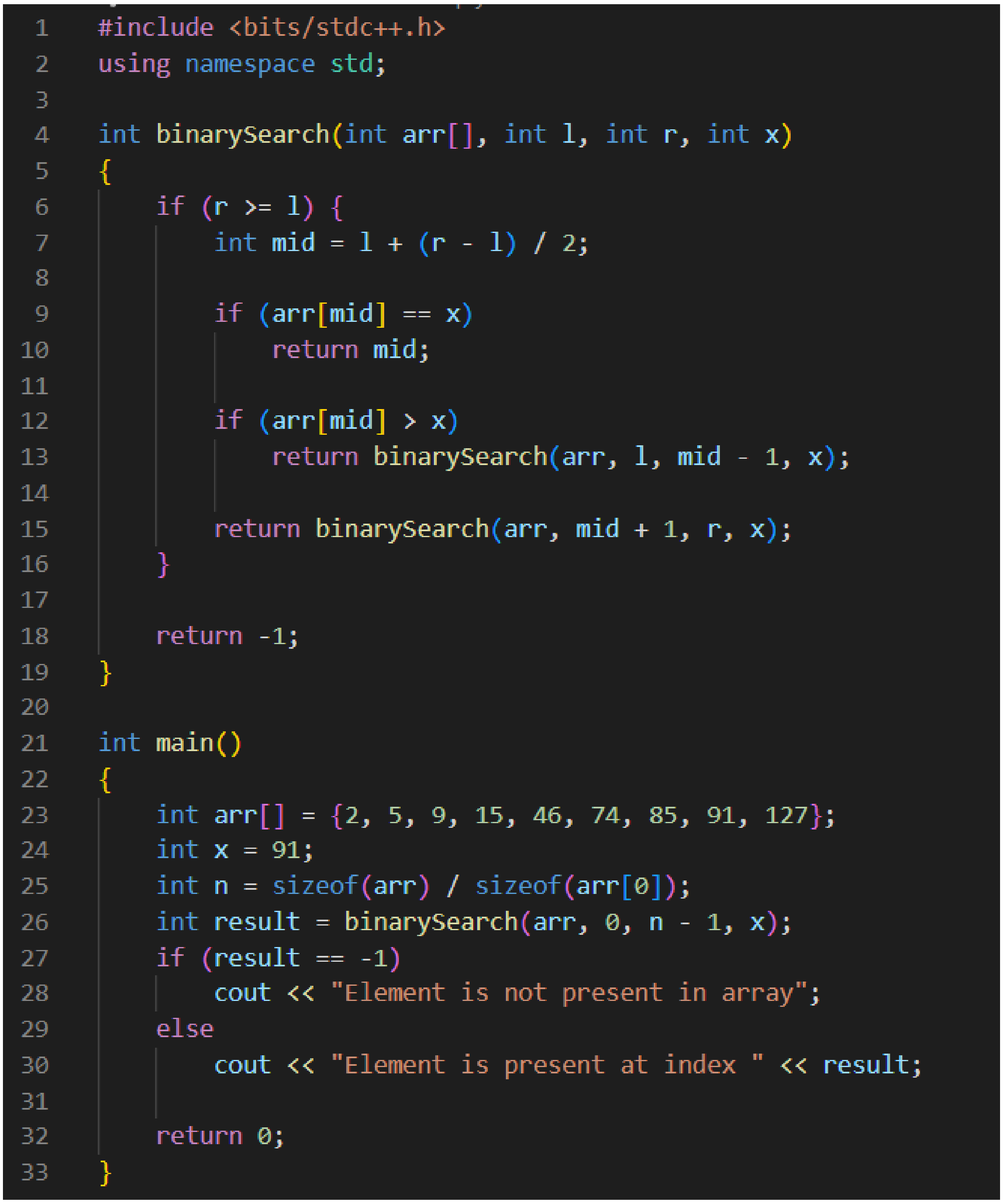


```cpp
#include <bits/stdc++.h>
using namespace std;

int binarySearch(int arr[], int l, int r, int x)
{
    while (l <= r) {
        int m = l + (r - l) / 2;

        if (arr[m] == x)
            return m;

        if (arr[m] < x)
            l = m + 1;

        else
            r = m - 1;
    }

    return -1;
}

int main()
{
    int arr[] = {2, 5, 9, 15, 46, 74, 85, 91, 127};
    int x = 91;
    int n = sizeof(arr) / sizeof(arr[0]);
    int result = binarySearch(arr, 0, n - 1, x);
    if (result == -1)
        cout << "Element is not present in array";
    else
        cout << "Element is present at index " << result;

    return 0;
}
```

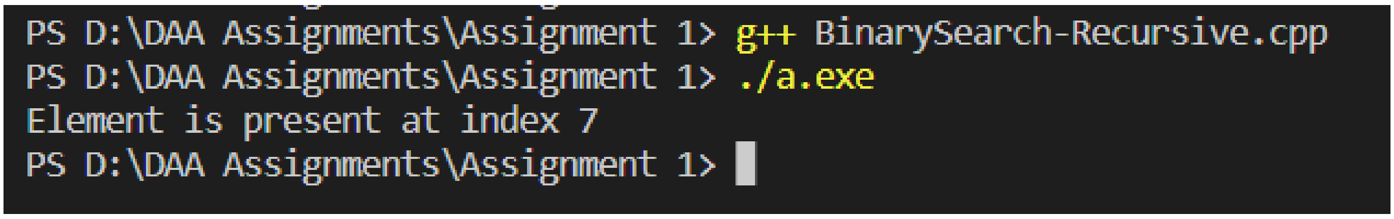


```
PS D:\DAA Assignments\Assignment 1> g++ BinarySearch-Iterative.cpp
PS D:\DAA Assignments\Assignment 1> ./a.exe
Element is present at index 7
PS D:\DAA Assignments\Assignment 1> 
```

Binary search – Recursive

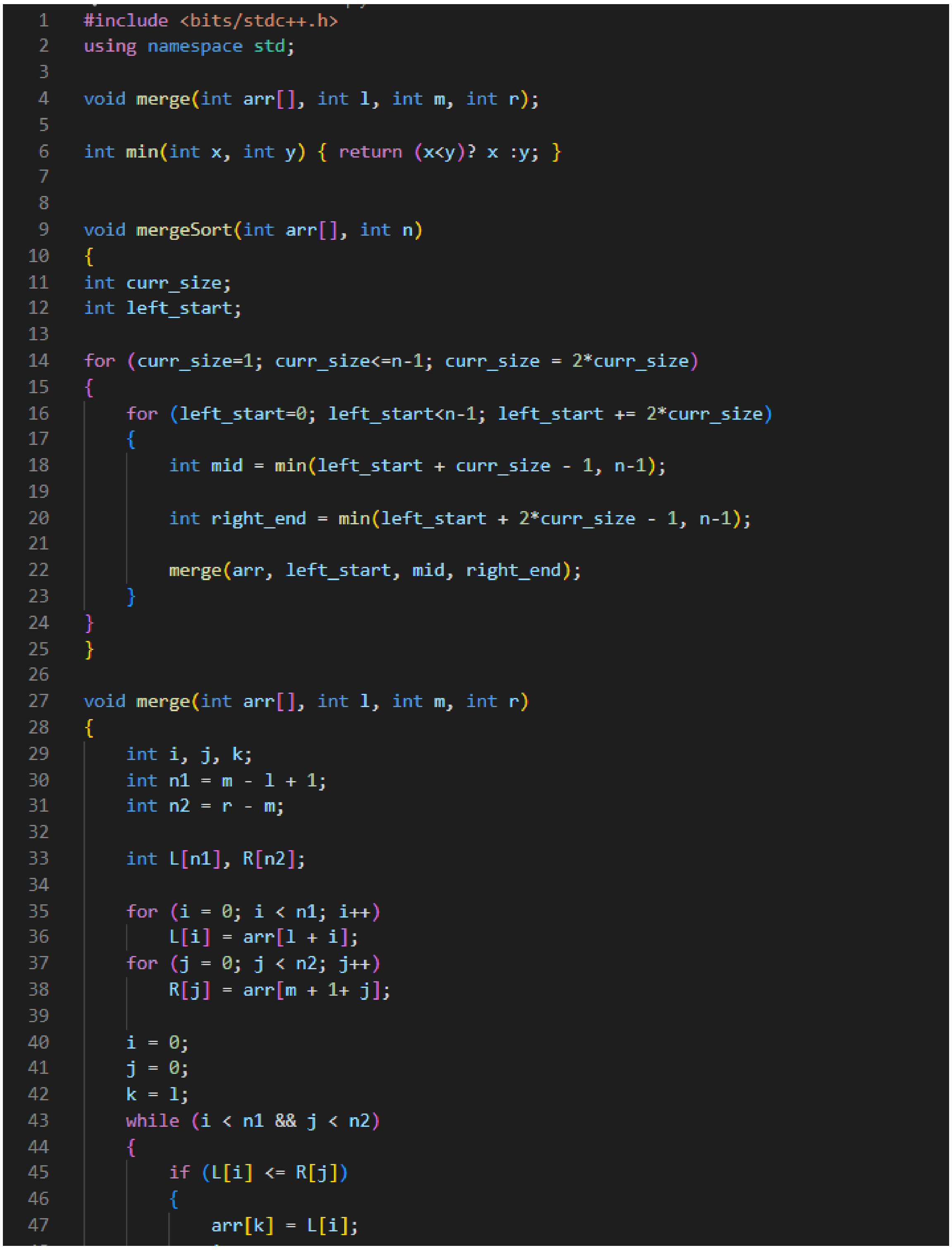


```cpp
#include <bits/stdc++.h>
using namespace std;

int binarySearch(int arr[], int l, int r, int x)
{
    if (r >= l) {
        int mid = l + (r - l) / 2;

        if (arr[mid] == x)
            return mid;

        if (arr[mid] > x)
            return binarySearch(arr, l, mid - 1, x);

        return binarySearch(arr, mid + 1, r, x);
    }

    return -1;
}

int main()
{
    int arr[] = {2, 5, 9, 15, 46, 74, 85, 91, 127};
    int x = 91;
    int n = sizeof(arr) / sizeof(arr[0]);
    int result = binarySearch(arr, 0, n - 1, x);
    if (result == -1)
        cout << "Element is not present in array";
    else
        cout << "Element is present at index " << result;

    return 0;
}
```

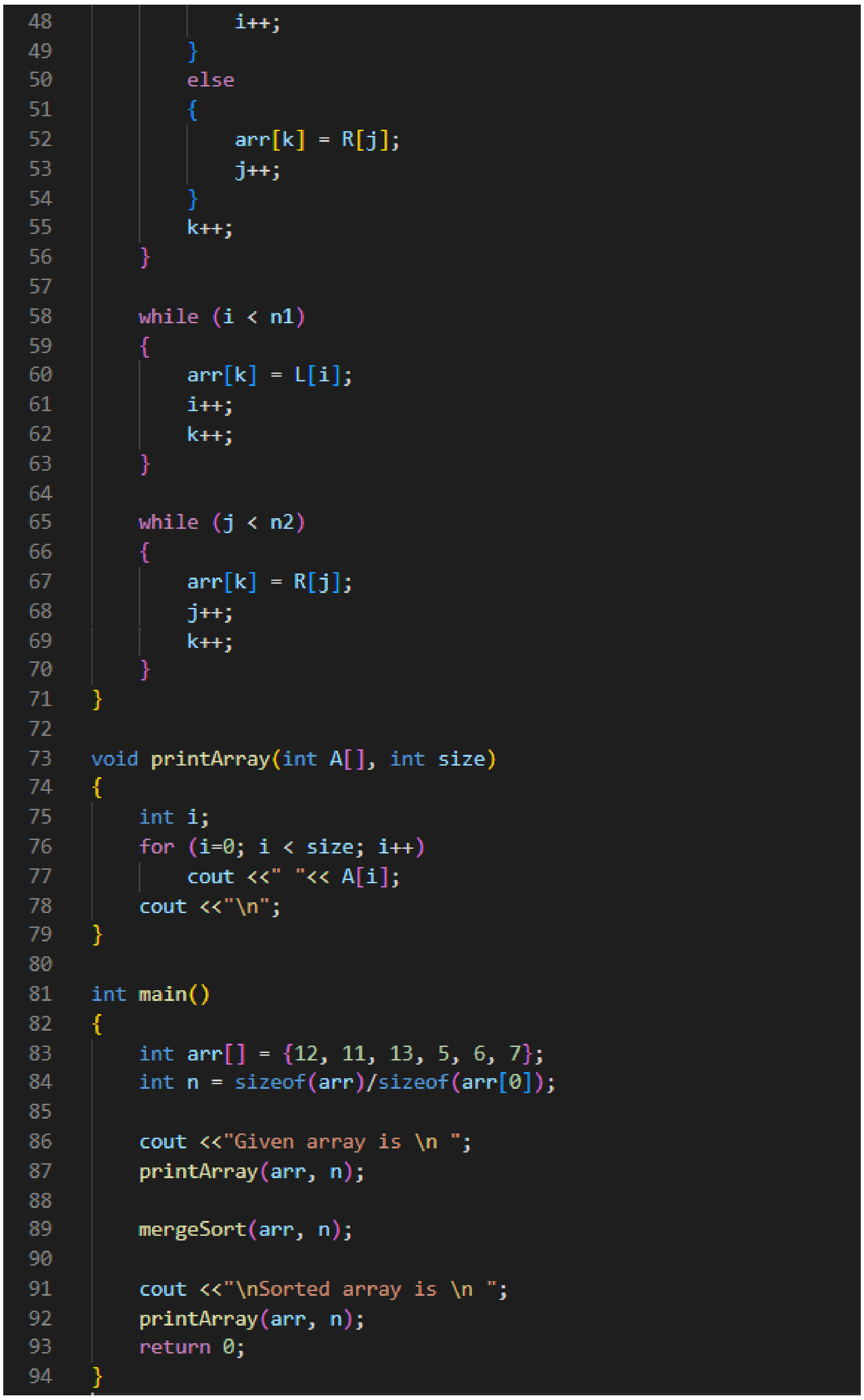


```
PS D:\DAA Assignments\Assignment 1> g++ BinarySearch-Recursive.cpp
PS D:\DAA Assignments\Assignment 1> ./a.exe
Element is present at index 7
PS D:\DAA Assignments\Assignment 1> 
```

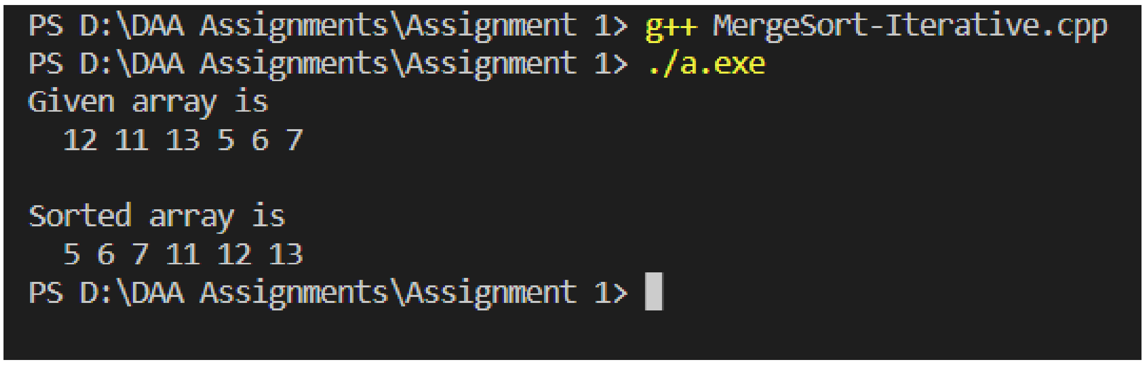
Merge Sort – Iterative



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

void merge(int arr[], int l, int m, int r);

int min(int x, int y) { return (x<y)? x :y; }


void mergeSort(int arr[], int n)
{
int curr_size;
int left_start;

for (curr_size=1; curr_size<=n-1; curr_size = 2*curr_size)
{
    for (left_start=0; left_start<n-1; left_start += 2*curr_size)
    {
        int mid = min(left_start + curr_size - 1, n-1);

        int right_end = min(left_start + 2*curr_size - 1, n-1);

        merge(arr, left_start, mid, right_end);
    }
}
}

void merge(int arr[], int l, int m, int r)
{
    int i, j, k;
    int n1 = m - l + 1;
    int n2 = r - m;

    int L[n1], R[n2];

    for (i = 0; i < n1; i++)
        L[i] = arr[l + i];
    for (j = 0; j < n2; j++)
        R[j] = arr[m + 1+ j];

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

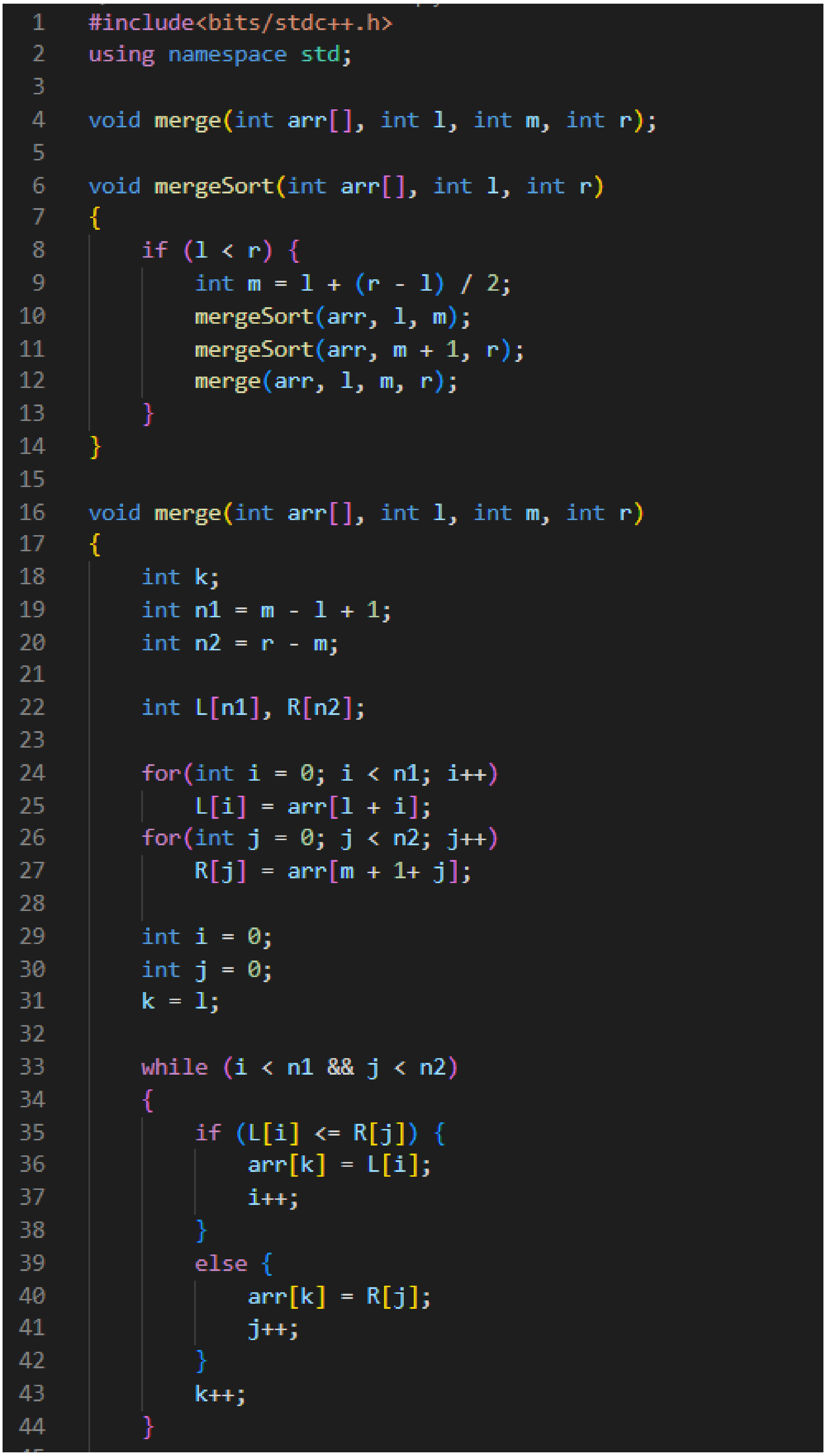


```cpp
                i++;
            }
            else
            {
                arr[k] = R[j];
                j++;
            }
            k++;
        }

        while (i < n1)
        {
            arr[k] = L[i];
            i++;
            k++;
        }

        while (j < n2)
        {
            arr[k] = R[j];
            j++;
            k++;
        }
}

void printArray(int A[], int size)
{
    int i;
    for (i=0; i < size; i++)
        cout <<" "<< A[i];
    cout <<"\n";
}

int main()
{
    int arr[] = {12, 11, 13, 5, 6, 7};
    int n = sizeof(arr)/sizeof(arr[0]);

    cout <<"Given array is \n ";
    printArray(arr, n);

    mergeSort(arr, n);

    cout <<"\nSorted array is \n ";
    printArray(arr, n);
    return 0;
}
```

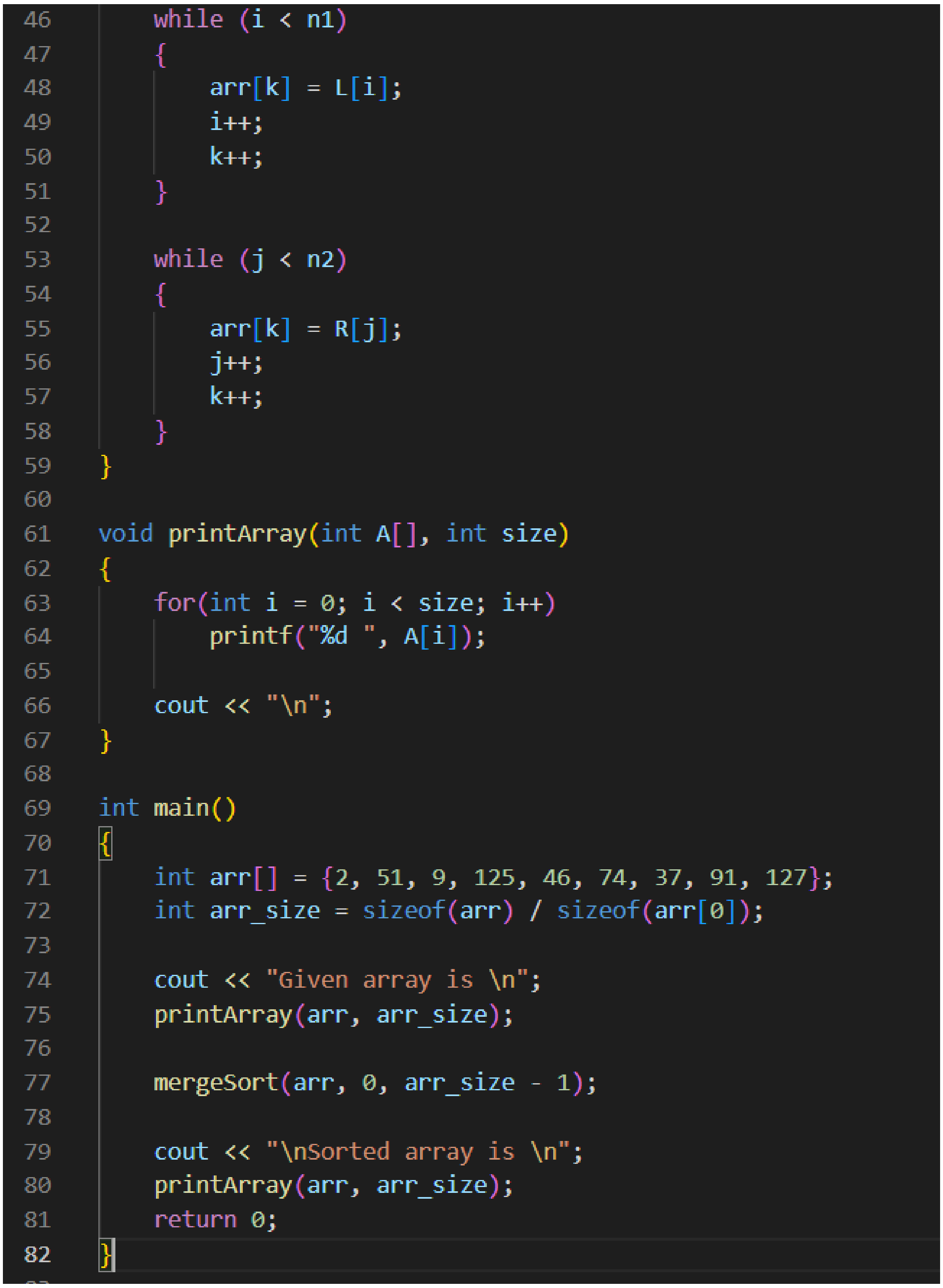


```
PS D:\DAA Assignments\Assignment 1> g++ MergeSort-Iterative.cpp
PS D:\DAA Assignments\Assignment 1> ./a.exe
Given array is
  12 11 13 5 6 7

Sorted array is
  5 6 7 11 12 13
PS D:\DAA Assignments\Assignment 1> 
```

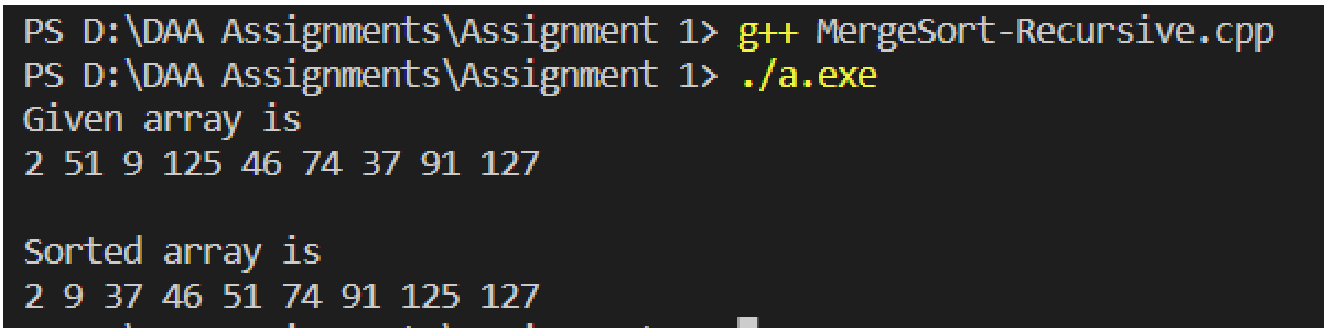
Merge Sort – Recursive



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

void merge(int arr[], int l, int m, int r);

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

void merge(int arr[], int l, int m, int r)
{
    int k;
    int n1 = m - l + 1;
    int n2 = r - m;

    int L[n1], R[n2];

    for(int i = 0; i < n1; i++)
        L[i] = arr[l + i];
    for(int j = 0; j < n2; j++)
        R[j] = arr[m + 1+ j];

    int i = 0;
    int j = 0;
    k = l;

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

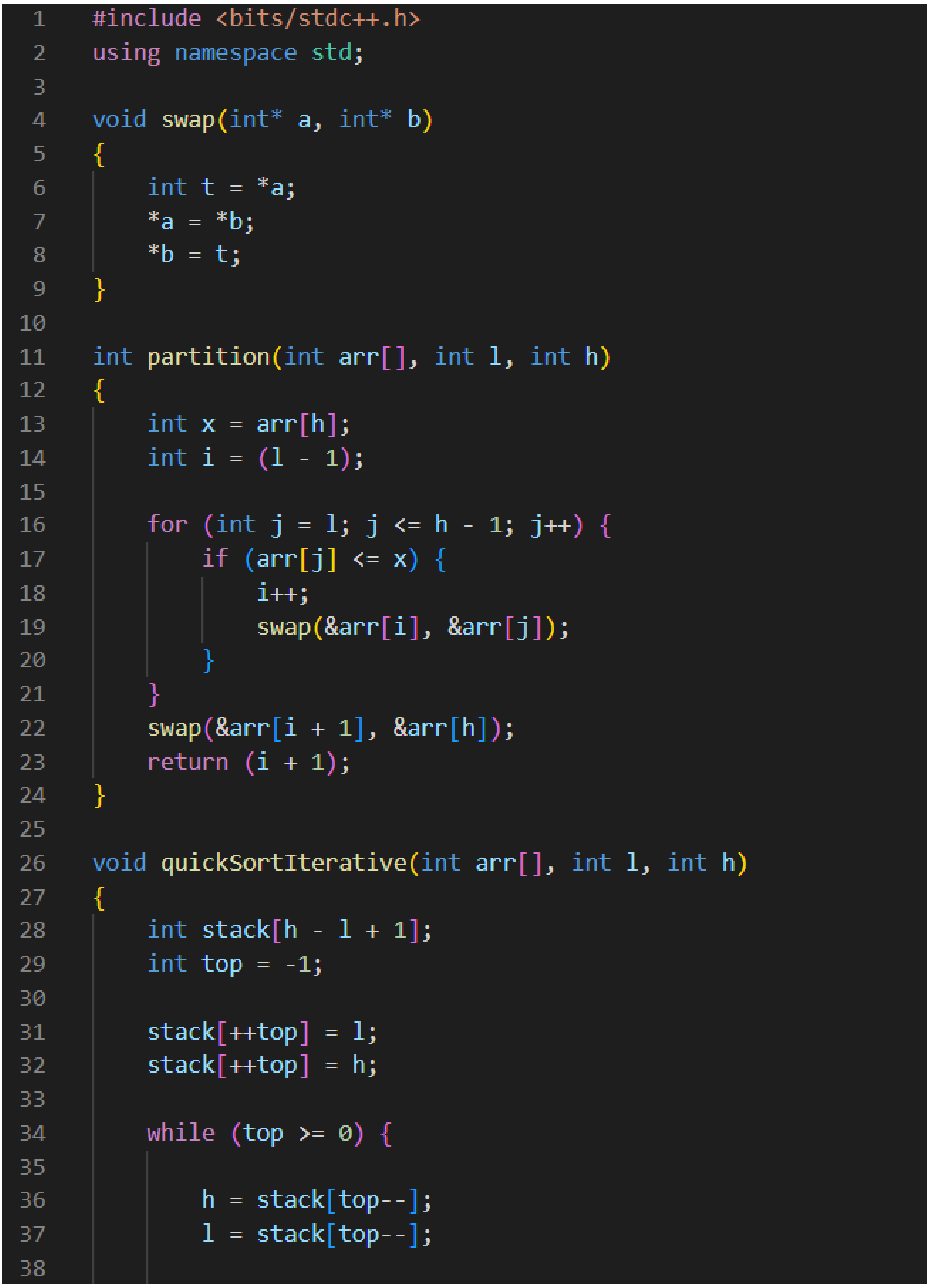


```cpp
    while (i < n1)
    {
        arr[k] = L[i];
        i++;
        k++;
    }

    while (j < n2)
    {
        arr[k] = R[j];
        j++;
        k++;
    }
}

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

    cout << "\n";
}

int main()
{
    int arr[] = {2, 51, 9, 125, 46, 74, 37, 91, 127};
    int arr_size = sizeof(arr) / sizeof(arr[0]);

    cout << "Given array is \n";
    printArray(arr, arr_size);

    mergeSort(arr, 0, arr_size - 1);

    cout << "\nSorted array is \n";
    printArray(arr, arr_size);
    return 0;
}
```

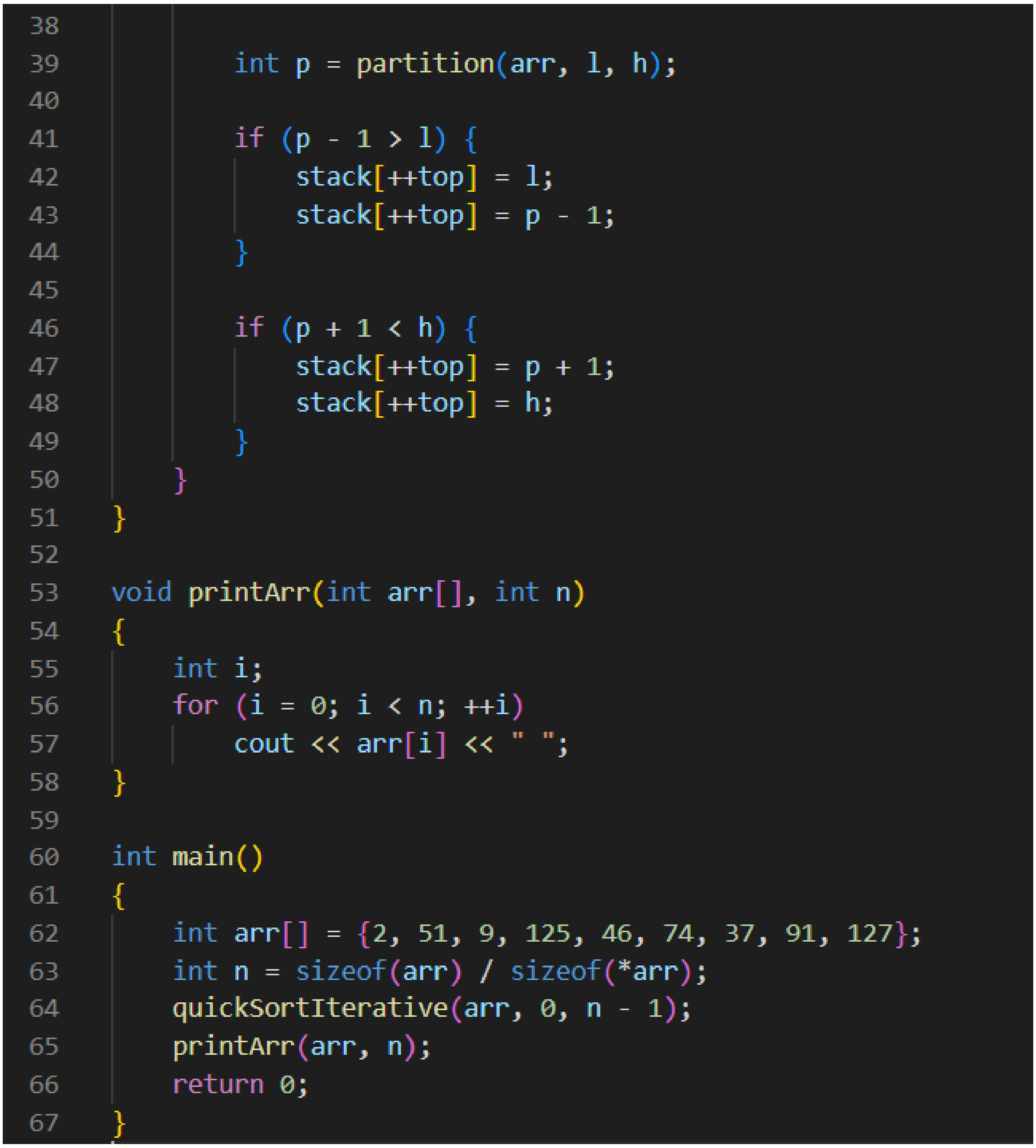


```
PS D:\DAA Assignments\Assignment 1> g++ MergeSort-Recursive.cpp
PS D:\DAA Assignments\Assignment 1> ./a.exe
Given array is
2 51 9 125 46 74 37 91 127

Sorted array is
2 9 37 46 51 74 91 125 127
```

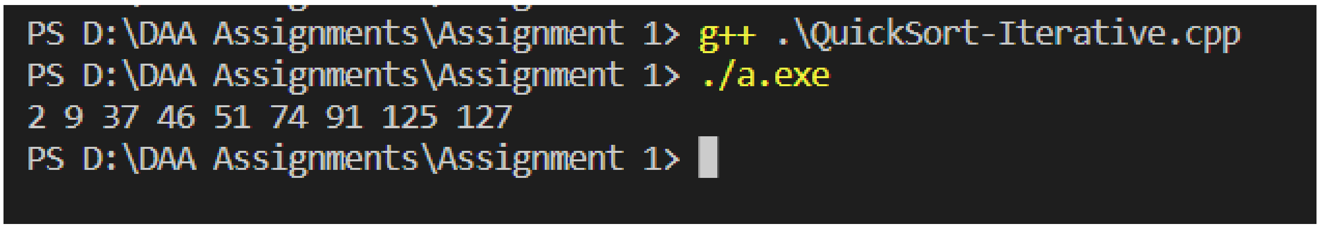
Quick Sort – Iterative



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

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

int partition(int arr[], int l, int h)
{
    int x = arr[h];
    int i = (l - 1);

    for (int j = l; j <= h - 1; j++) {
        if (arr[j] <= x) {
            i++;
            swap(&arr[i], &arr[j]);
        }
    }
    swap(&arr[i + 1], &arr[h]);
    return (i + 1);
}

void quickSortIterative(int arr[], int l, int h)
{
    int stack[h - l + 1];
    int top = -1;

    stack[++top] = l;
    stack[++top] = h;

    while (top >= 0) {

        h = stack[top--];
        l = stack[top--];

```

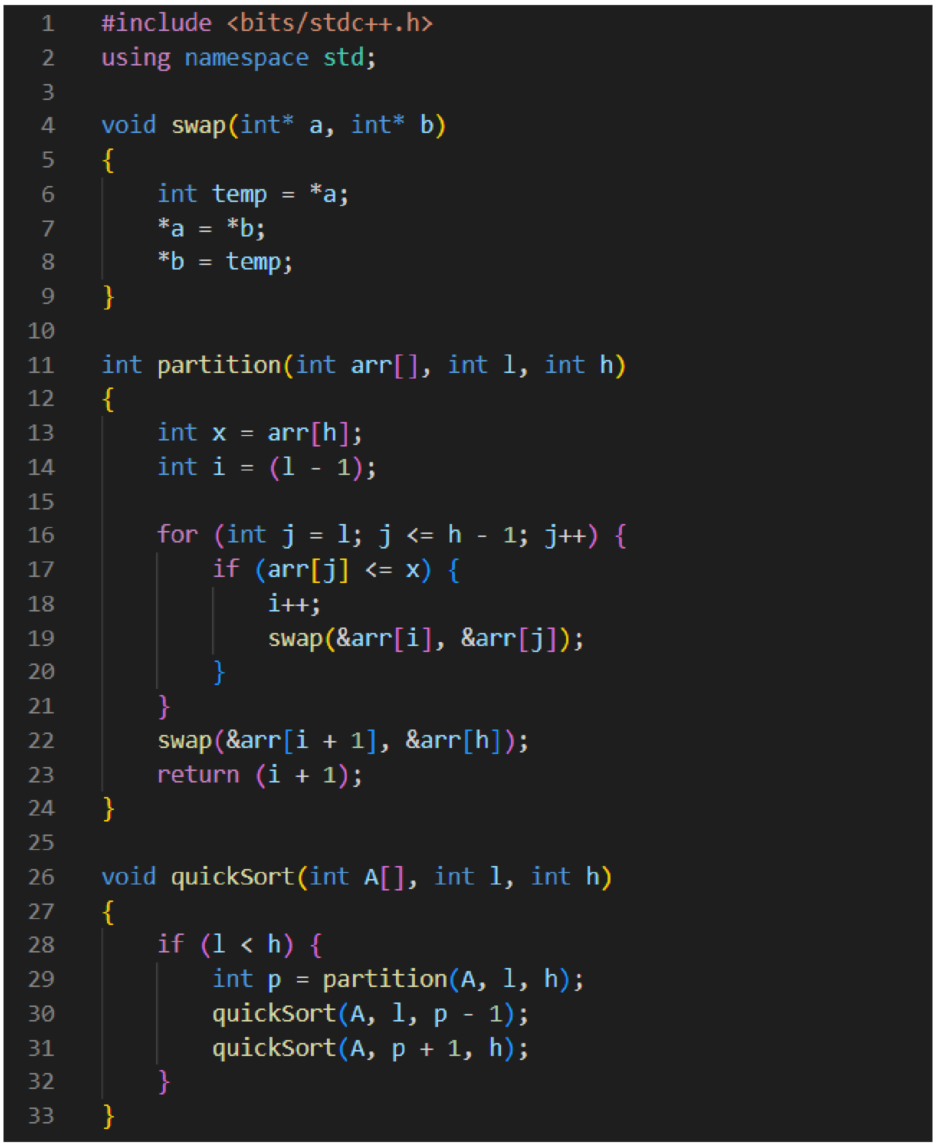


```cpp

        int p = partition(arr, l, h);

        if (p - 1 > l) {
            stack[++top] = l;
            stack[++top] = p - 1;
        }

        if (p + 1 < h) {
            stack[++top] = p + 1;
            stack[++top] = h;
        }
    }
}

void printArr(int arr[], int n)
{
    int i;
    for (i = 0; i < n; ++i)
        cout << arr[i] << " ";
}

int main()
{
    int arr[] = {2, 51, 9, 125, 46, 74, 37, 91, 127};
    int n = sizeof(arr) / sizeof(*arr);
    quickSortIterative(arr, 0, n - 1);
    printArr(arr, n);
    return 0;
}
```

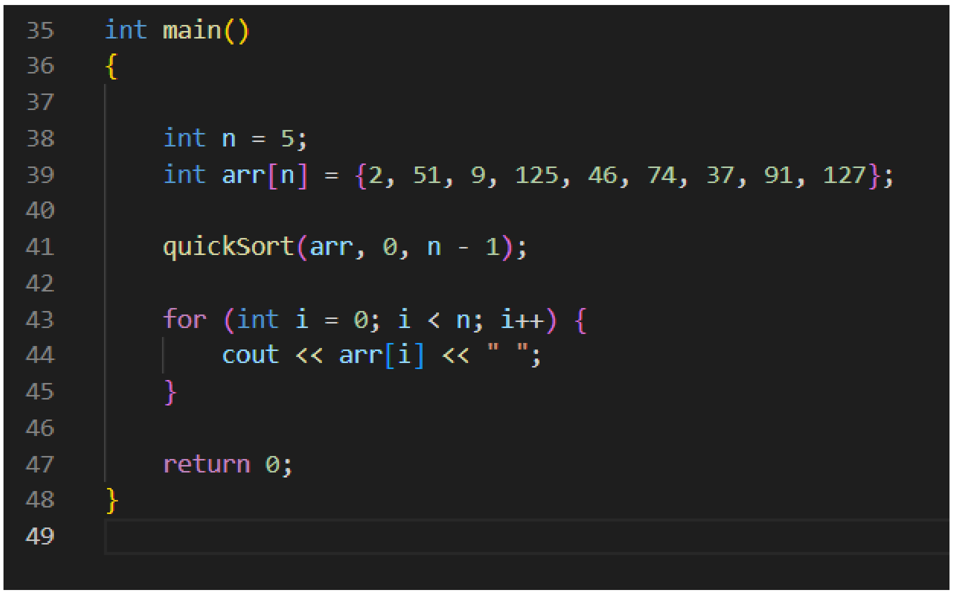


```
PS D:\DAA Assignments\Assignment 1> g++ .\QuickSort-Iterative.cpp
PS D:\DAA Assignments\Assignment 1> ./a.exe
2 9 37 46 51 74 91 125 127
PS D:\DAA Assignments\Assignment 1>
```

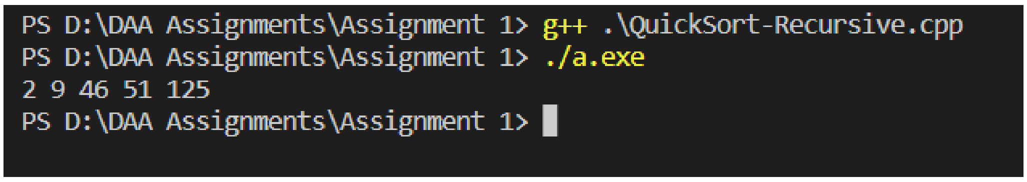
Quick Sort – Recursive



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

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

int partition(int arr[], int l, int h)
{
    int x = arr[h];
    int i = (l - 1);

    for (int j = l; j <= h - 1; j++) {
        if (arr[j] <= x) {
            i++;
            swap(&arr[i], &arr[j]);
        }
    }
    swap(&arr[i + 1], &arr[h]);
    return (i + 1);
}

void quickSort(int A[], int l, int h)
{
    if (l < h) {
        int p = partition(A, l, h);
        quickSort(A, l, p - 1);
        quickSort(A, p + 1, h);
    }
}
```

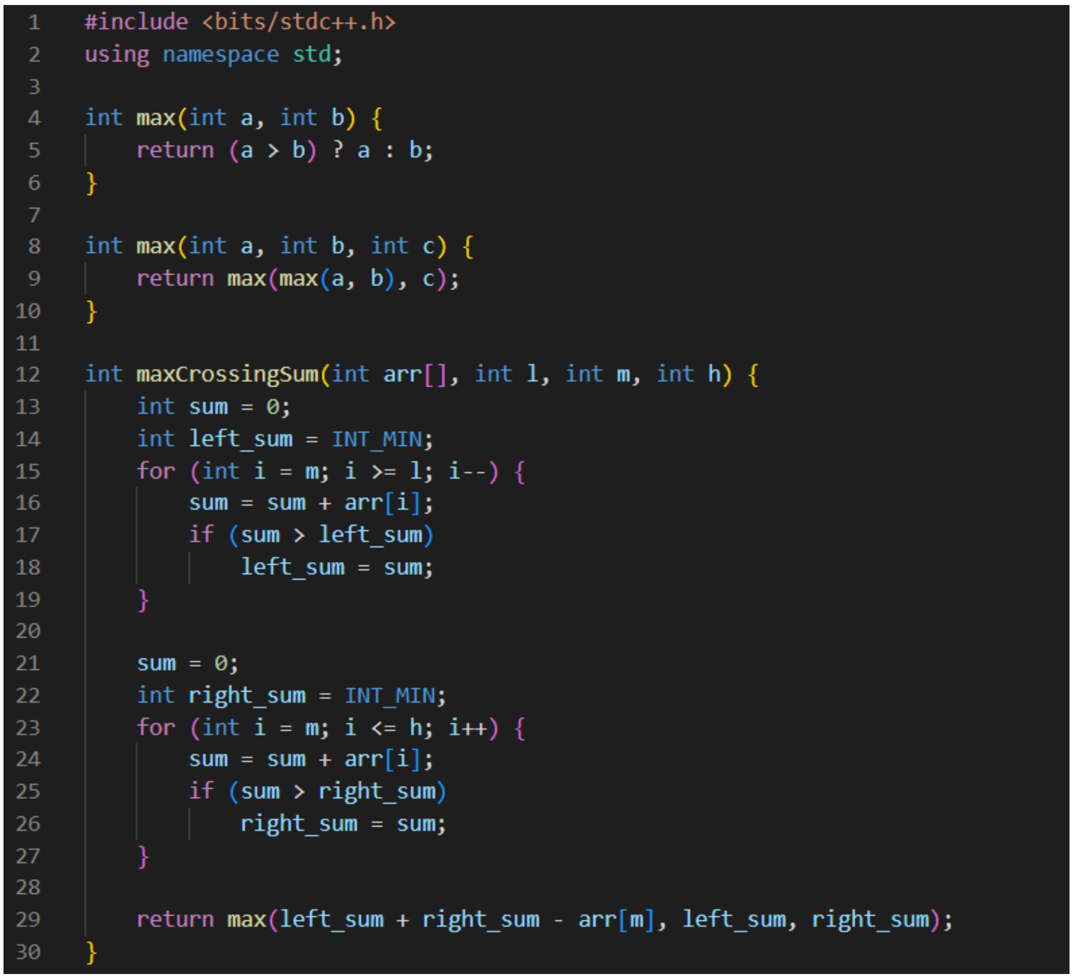


```cpp
int main()
{

    int n = 5;
    int arr[n] = {2, 51, 9, 125, 46, 74, 37, 91, 127};

    quickSort(arr, 0, n - 1);

    for (int i = 0; i < n; i++) {
        cout << arr[i] << " ";
    }

    return 0;
}

```

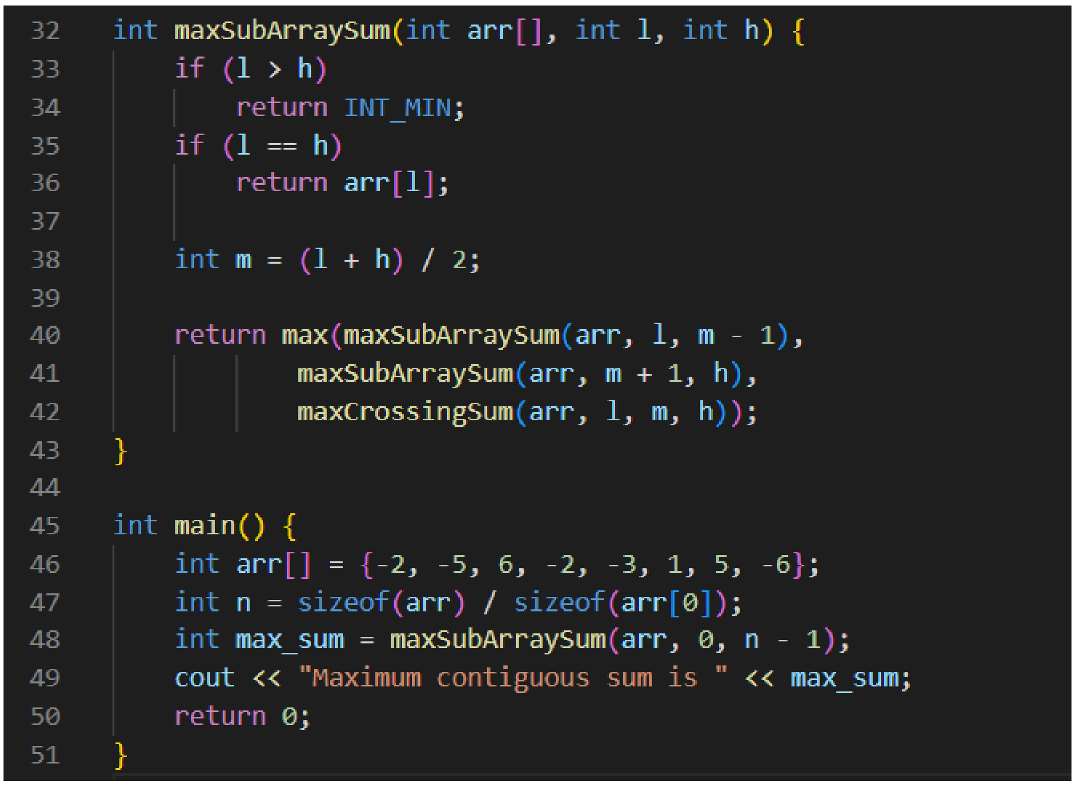


```
PS D:\DAA Assignments\Assignment 1> g++ .\QuickSort-Recursive.cpp
PS D:\DAA Assignments\Assignment 1> ./a.exe
2 9 46 51 125
PS D:\DAA Assignments\Assignment 1> █
```

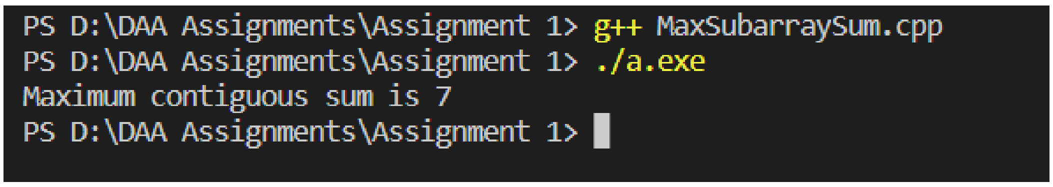
Maximum Subarray Sum



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

int max(int a, int b) {
    return (a > b) ? a : b;
}

int max(int a, int b, int c) {
    return max(max(a, b), c);
}

int maxCrossingSum(int arr[], int l, int m, int h) {
    int sum = 0;
    int left_sum = INT_MIN;
    for (int i = m; i >= l; i--) {
        sum = sum + arr[i];
        if (sum > left_sum)
            left_sum = sum;
    }

    sum = 0;
    int right_sum = INT_MIN;
    for (int i = m; i <= h; i++) {
        sum = sum + arr[i];
        if (sum > right_sum)
            right_sum = sum;
    }

    return max(left_sum + right_sum - arr[m], left_sum, right_sum);
}
```



```cpp
int maxSubArraySum(int arr[], int l, int h) {
    if (l > h)
        return INT_MIN;
    if (l == h)
        return arr[l];

    int m = (l + h) / 2;

    return max(maxSubArraySum(arr, l, m - 1),
               maxSubArraySum(arr, m + 1, h),
               maxCrossingSum(arr, l, m, h));
}

int main() {
    int arr[] = {-2, -5, 6, -2, -3, 1, 5, -6};
    int n = sizeof(arr) / sizeof(arr[0]);
    int max_sum = maxSubArraySum(arr, 0, n - 1);
    cout << "Maximum contiguous sum is " << max_sum;
    return 0;
}
```



```
PS D:\DAA Assignments\Assignment 1> g++ MaxSubarraySum.cpp
PS D:\DAA Assignments\Assignment 1> ./a.exe
Maximum contiguous sum is 7
PS D:\DAA Assignments\Assignment 1> 
```