Shrihari K Mangle
CSB2-74 12010517

LAB-2

2. Implement following Sorting Algorithms:

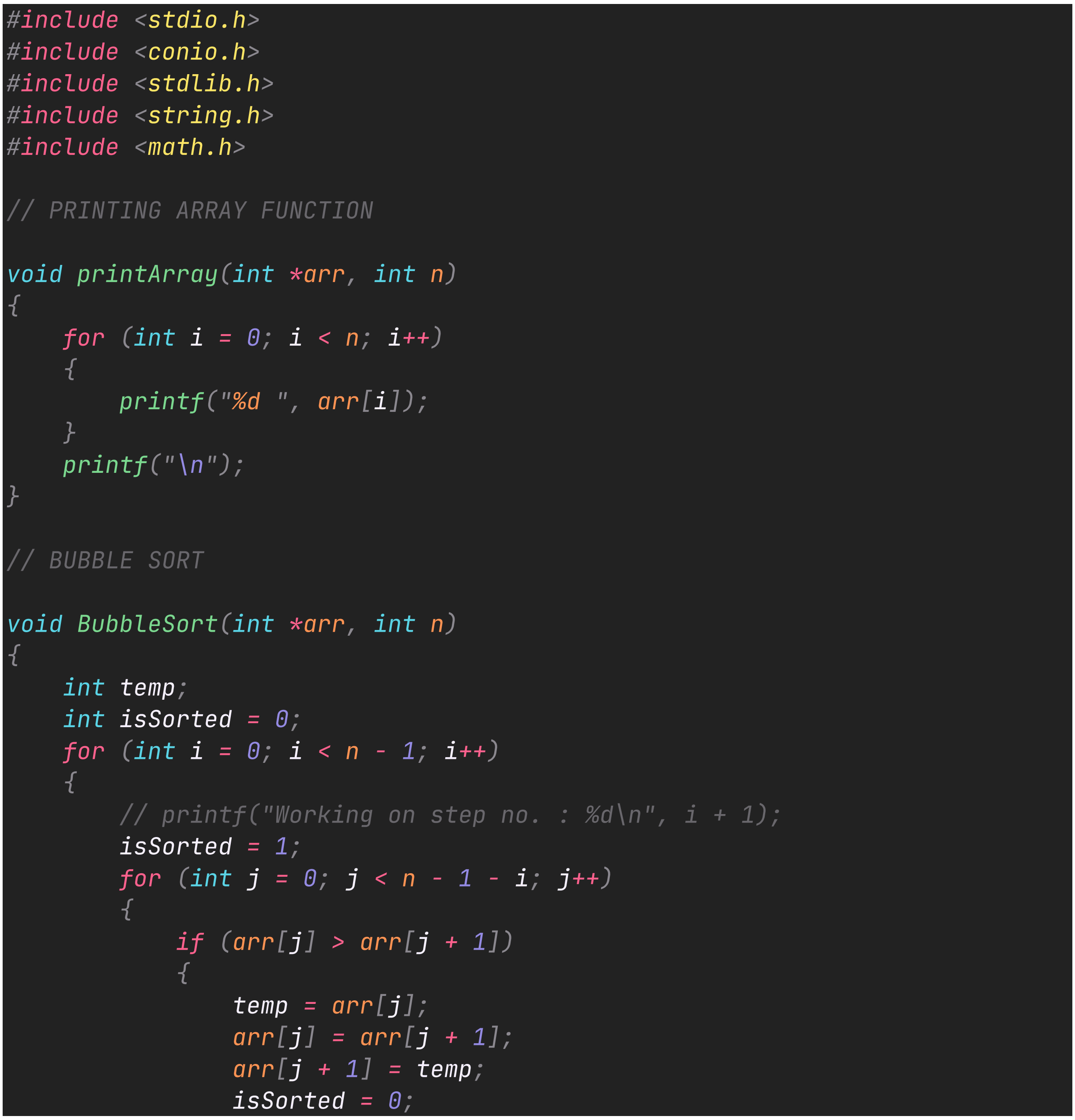
A. Bubble Sort.

B. Quick Sort.

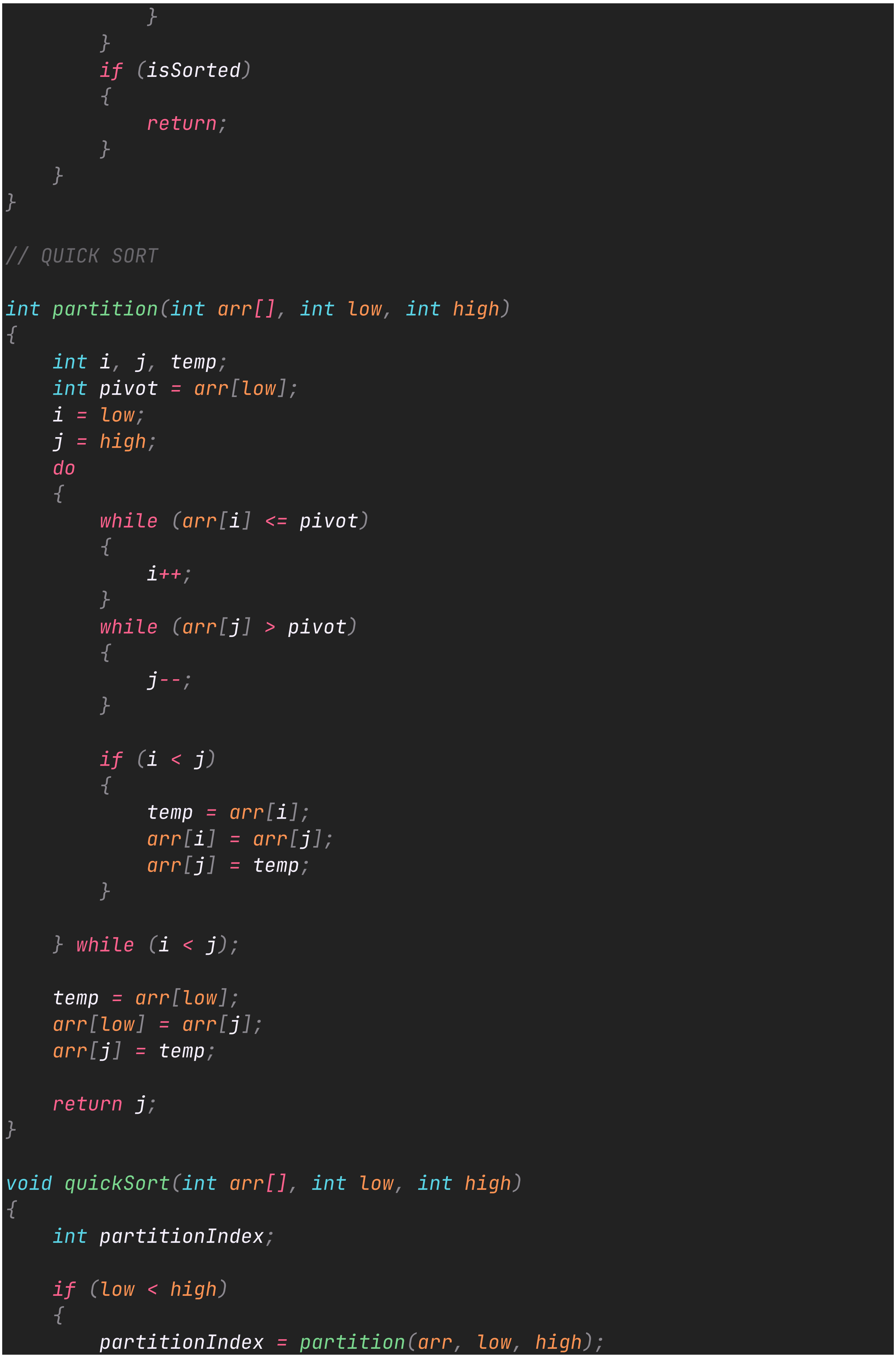
C. Heap Sort.

For sorted output; apply Binary Search for searching an Element and also mention number of comparisons required to search the element.

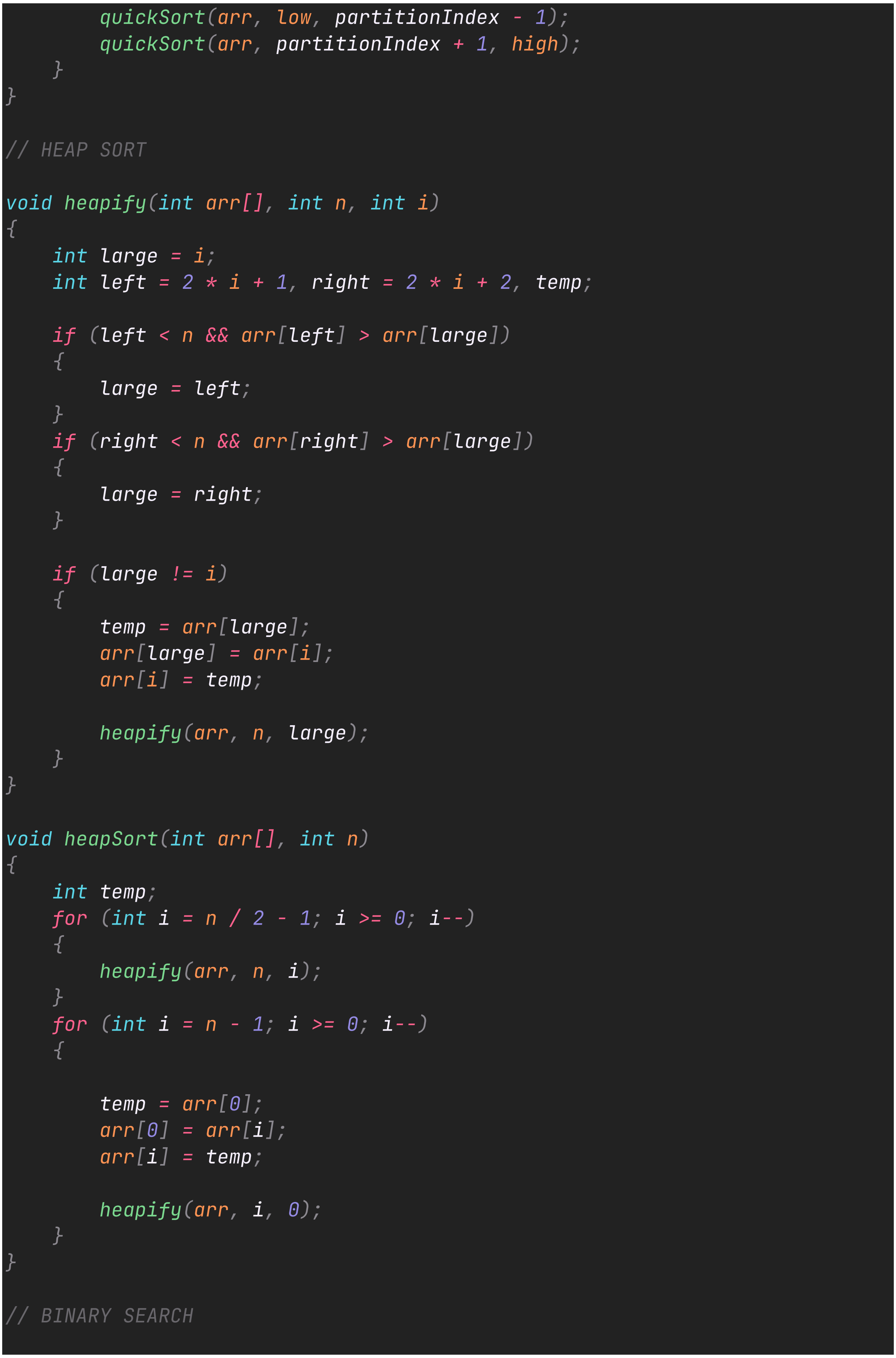
CODE :



```c
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>

// PRINTING ARRAY FUNCTION

void printArray(int *arr, int n)
{
    for (int i = 0; i < n; i++)
    {
        printf("%d ", arr[i]);
    }
    printf("\n");
}

// BUBBLE SORT

void BubbleSort(int *arr, int n)
{
    int temp;
    int isSorted = 0;
    for (int i = 0; i < n - 1; i++)
    {
        // printf("Working on step no. : %d\n", i + 1);
        isSorted = 1;
        for (int j = 0; j < n - 1 - i; j++)
        {
            if (arr[j] > arr[j + 1])
            {
                temp = arr[j];
                arr[j] = arr[j + 1];
                arr[j + 1] = temp;
                isSorted = 0;
```

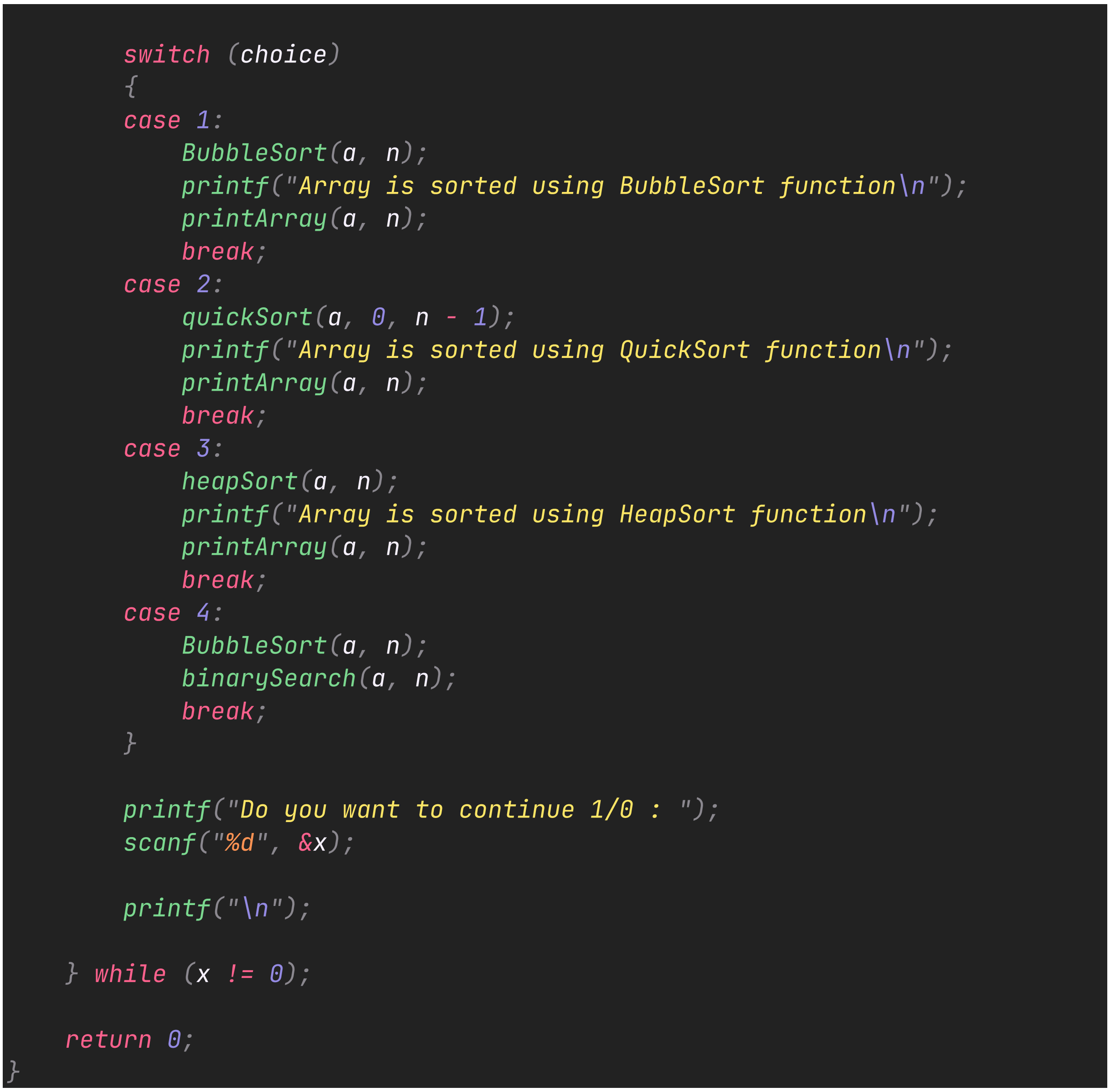


```
            }
        }
        if (isSorted)
        {
            return;
        }
    }
}

// QUICK SORT

int partition(int arr[], int low, int high)
{
    int i, j, temp;
    int pivot = arr[low];
    i = low;
    j = high;
    do
    {
        while (arr[i] <= pivot)
        {
            i++;
        }
        while (arr[j] > pivot)
        {
            j--;
        }

        if (i < j)
        {
            temp = arr[i];
            arr[i] = arr[j];
            arr[j] = temp;
        }

    } while (i < j);

    temp = arr[low];
    arr[low] = arr[j];
    arr[j] = temp;

    return j;
}

void quickSort(int arr[], int low, int high)
{
    int partitionIndex;

    if (low < high)
    {
        partitionIndex = partition(arr, low, high);
```



```
        quickSort(arr, low, partitionIndex - 1);
        quickSort(arr, partitionIndex + 1, high);
    }
}

// HEAP SORT

void heapify(int arr[], int n, int i)
{
    int large = i;
    int left = 2 * i + 1, right = 2 * i + 2, temp;

    if (left < n && arr[left] > arr[large])
    {
        large = left;
    }
    if (right < n && arr[right] > arr[large])
    {
        large = right;
    }

    if (large != i)
    {
        temp = arr[large];
        arr[large] = arr[i];
        arr[i] = temp;

        heapify(arr, n, large);
    }
}

void heapSort(int arr[], int n)
{
    int temp;
    for (int i = n / 2 - 1; i >= 0; i--)
    {
        heapify(arr, n, i);
    }
    for (int i = n - 1; i >= 0; i--)
    {

        temp = arr[0];
        arr[0] = arr[i];
        arr[i] = temp;

        heapify(arr, i, 0);
    }
}

// BINARY SEARCH
```

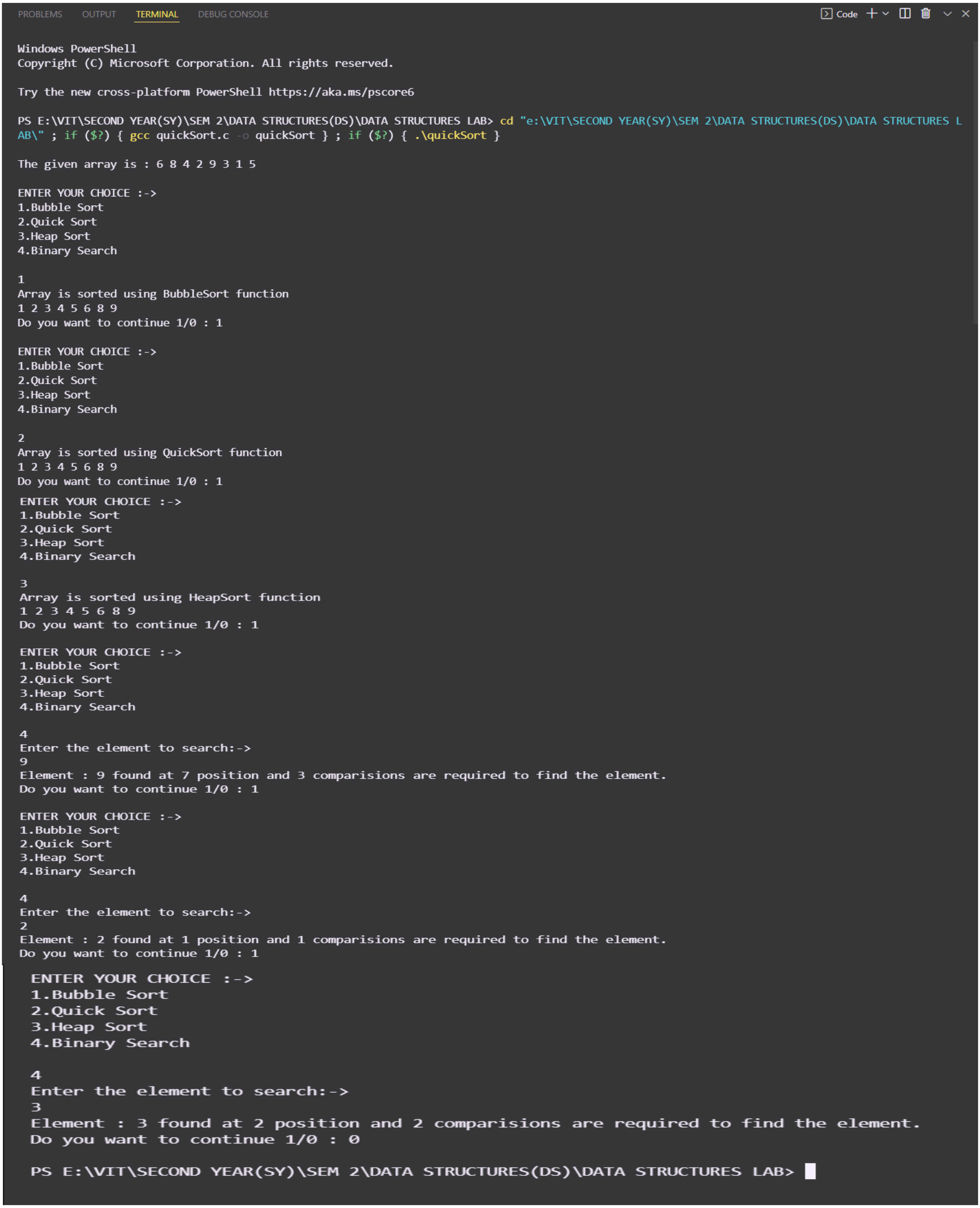
```c
void binarySearch(int arr[], int size)
{
    int element;
    printf("Enter the element to search:->\n");
    scanf("%d", &element);
    int low = 0, mid, high = size - 1, count = 0;

    while (low <= high)
    {
        mid = (low + high) / 2;
        if (arr[mid] == element)
        {
            printf("Element : %d found at %d position and %d comparisions are required to find the element.\n", arr[mid], mid, count);
        }
        if (arr[mid] < element)
        {
            low = mid + 1;
        }
        else
        {
            high = mid - 1;
        }
        count++;
    }
}



int main()
{
    printf("\n");
    int choice, element;
    int a[] = {6, 8, 4, 2, 9, 3, 1, 5};
    int n = sizeof(a) / sizeof(int);
    int x = 0;

    printf("The given array is : ");
    for (int i = 0; i < n; i++)
    {
        printf("%d ", a[i]);
    }

    printf("\n\n");


    do
    {
        printf("ENTER YOUR CHOICE :-> \n1.Bubble Sort\n2.Quick Sort\n3.Heap Sort\n4.Binary Search\n\n");
        scanf("%d", &choice);
```



```
        switch (choice)
        {
        case 1:
            BubbleSort(a, n);
            printf("Array is sorted using BubbleSort function\n");
            printArray(a, n);
            break;
        case 2:
            quickSort(a, 0, n - 1);
            printf("Array is sorted using QuickSort function\n");
            printArray(a, n);
            break;
        case 3:
            heapSort(a, n);
            printf("Array is sorted using HeapSort function\n");
            printArray(a, n);
            break;
        case 4:
            BubbleSort(a, n);
            binarySearch(a, n);
            break;
        }

        printf("Do you want to continue 1/0 : ");
        scanf("%d", &x);

        printf("\n");

    } while (x != 0);

    return 0;
}
```

OUTPUT :

Shrihari K Mangle
CSB2-74 12010517



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS E:\VIT\SECOND YEAR(SY)\SEM 2\DATA STRUCTURES(DS)\DATA STRUCTURES LAB> cd "e:\VIT\SECOND YEAR(SY)\SEM 2\DATA STRUCTURES(DS)\DATA STRUCTURES L
AB\" ; if ($?) { gcc quickSort.c -o quickSort } ; if ($?) { .\quickSort }

The given array is : 6 8 4 2 9 3 1 5

ENTER YOUR CHOICE :->
1.Bubble Sort
2.Quick Sort
3.Heap Sort
4.Binary Search

1
Array is sorted using BubbleSort function
1 2 3 4 5 6 8 9
Do you want to continue 1/0 : 1

ENTER YOUR CHOICE :->
1.Bubble Sort
2.Quick Sort
3.Heap Sort
4.Binary Search

2
Array is sorted using QuickSort function
1 2 3 4 5 6 8 9
Do you want to continue 1/0 : 1
ENTER YOUR CHOICE :->
1.Bubble Sort
2.Quick Sort
3.Heap Sort
4.Binary Search

3
Array is sorted using HeapSort function
1 2 3 4 5 6 8 9
Do you want to continue 1/0 : 1

ENTER YOUR CHOICE :->
1.Bubble Sort
2.Quick Sort
3.Heap Sort
4.Binary Search

4
Enter the element to search:->
9
Element : 9 found at 7 position and 3 comparisions are required to find the element.
Do you want to continue 1/0 : 1

ENTER YOUR CHOICE :->
1.Bubble Sort
2.Quick Sort
3.Heap Sort
4.Binary Search

4
Enter the element to search:->
2
Element : 2 found at 1 position and 1 comparisions are required to find the element.
Do you want to continue 1/0 : 1
ENTER YOUR CHOICE :->
1.Bubble Sort
2.Quick Sort
3.Heap Sort
4.Binary Search

4
Enter the element to search:->
3
Element : 3 found at 2 position and 2 comparisions are required to find the element.
Do you want to continue 1/0 : 0

PS E:\VIT\SECOND YEAR(SY)\SEM 2\DATA STRUCTURES(DS)\DATA STRUCTURES LAB> █
```