AMRITA VISHWA VIDYAPEETHAM
DEEMED TO BE UNIVERSITY UNDER SECTION 3 OF UGC ACT, 1956
SCHOOL OF COMPUTING

DESIGN AND ANALYSIS OF ALGORITHMS

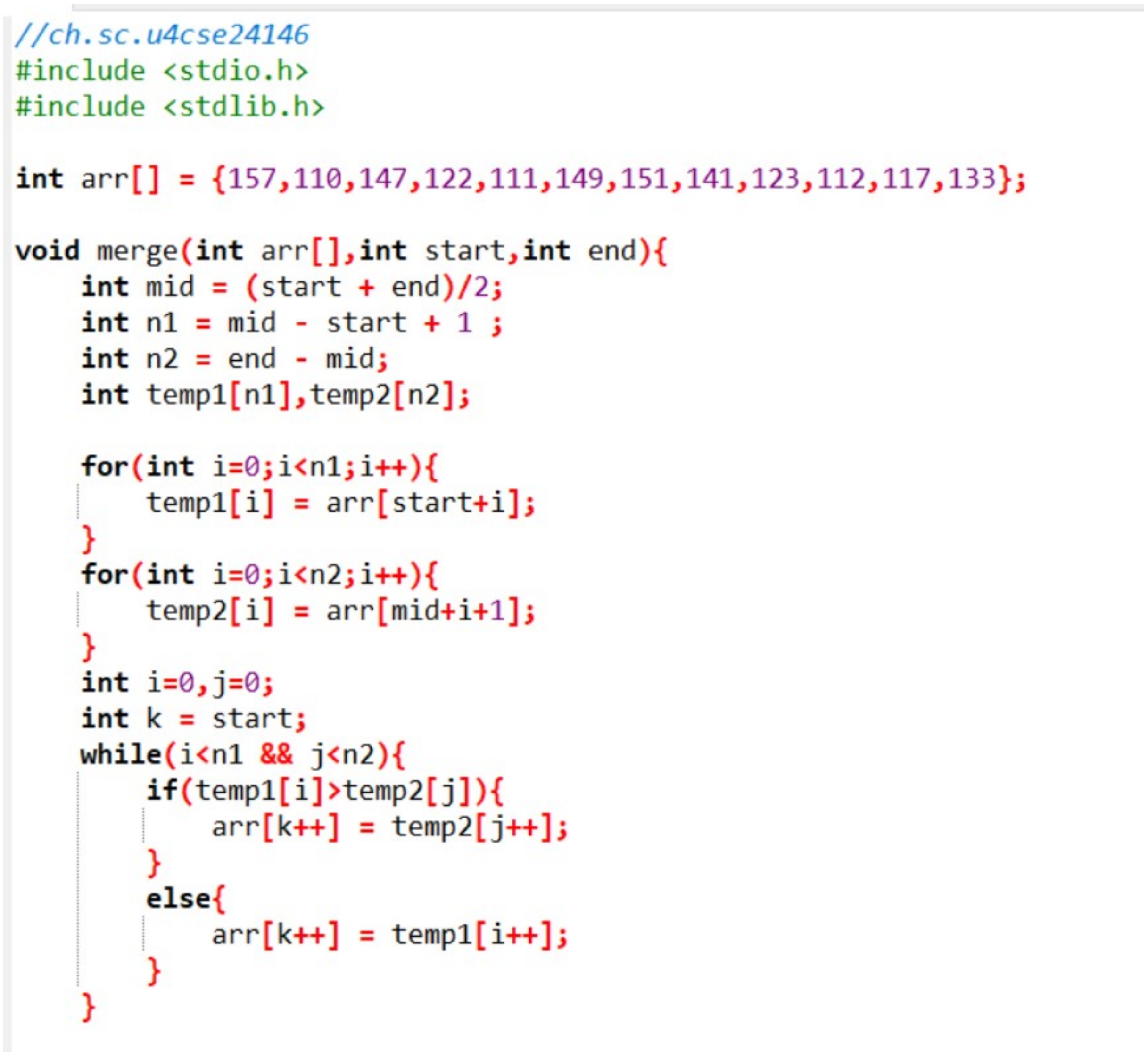
LAB WORKBOOK

WEEK - 4

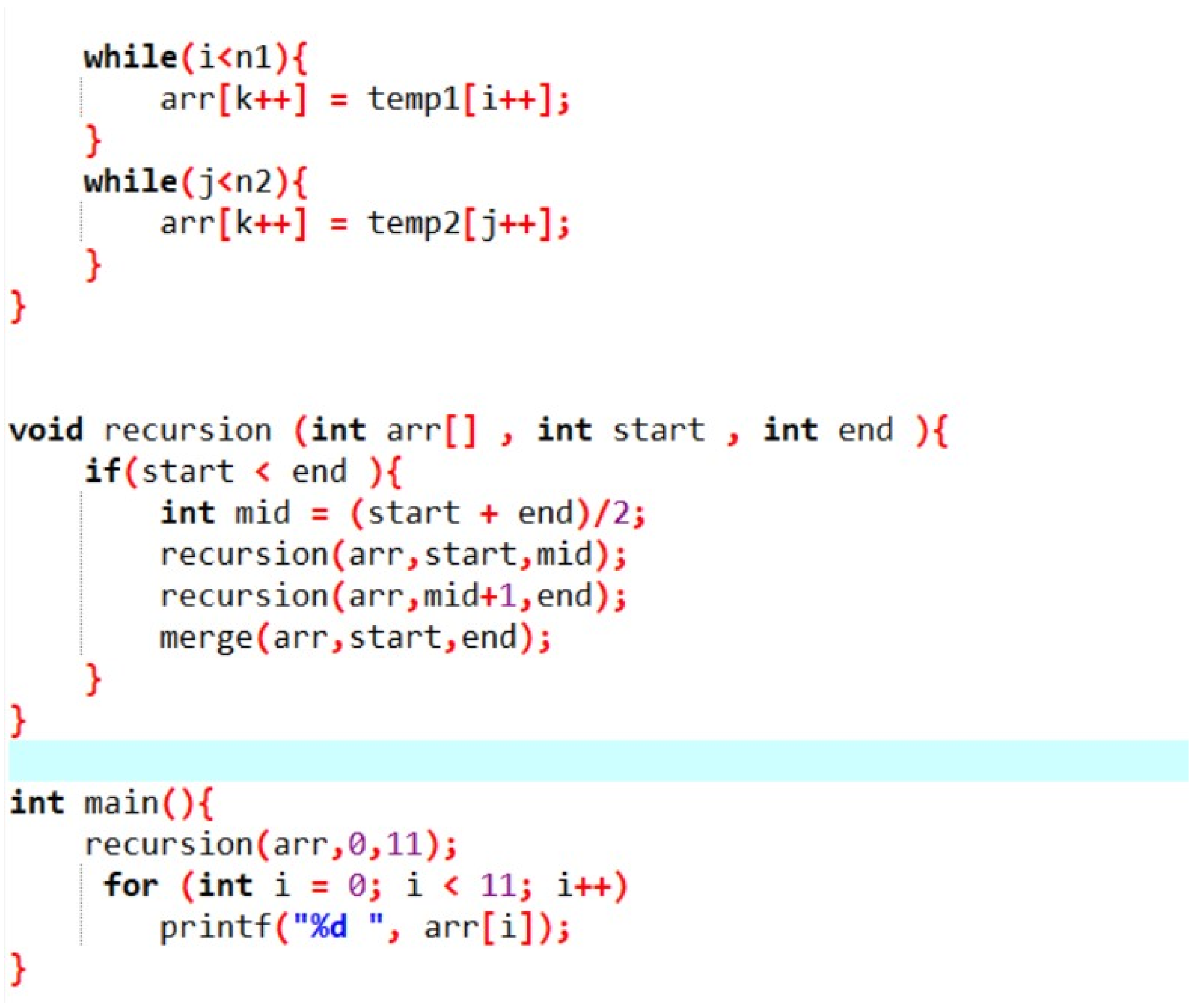
NAME	**: SRINIVAS R**
ROLL NUMBER	**: CH.SC.U4CSE24146**
CLASS	**: CSE-B**

Question 1: Write a C program that reads N integers from the user and sorts it using Merge Sort.

CODE:

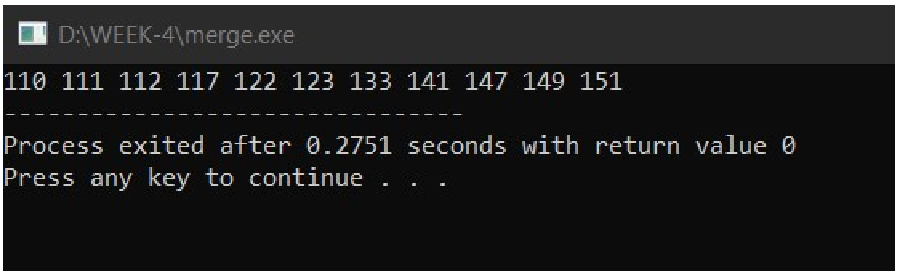


```c
//ch.sc.u4cse24146
#include <stdio.h>
#include <stdlib.h>

int arr[] = {157,110,147,122,111,149,151,141,123,112,117,133};

void merge(int arr[],int start,int end){
    int mid = (start + end)/2;
    int n1 = mid - start + 1 ;
    int n2 = end - mid;
    int temp1[n1],temp2[n2];

    for(int i=0;i<n1;i++){
        temp1[i] = arr[start+i];
    }
    for(int i=0;i<n2;i++){
        temp2[i] = arr[mid+i+1];
    }
    int i=0,j=0;
    int k = start;
    while(i<n1 && j<n2){
        if(temp1[i]>temp2[j]){
            arr[k++] = temp2[j++];
        }
        else{
            arr[k++] = temp1[i++];
        }
    }
```



```c
    while(i<n1){
        arr[k++] = temp1[i++];
    }
    while(j<n2){
        arr[k++] = temp2[j++];
    }
}


void recursion (int arr[] , int start , int end ){
    if(start < end ){
        int mid = (start + end)/2;
        recursion(arr,start,mid);
        recursion(arr,mid+1,end);
        merge(arr,start,end);
    }
}

int main(){
    recursion(arr,0,11);
     for (int i = 0; i < 11; i++)
        printf("%d ", arr[i]);
}
```

OUTPUT:



D:\WEEK-4\merge.exe

```
110 111 112 117 122 123 133 141 147 149 151
--------------------------------
Process exited after 0.2751 seconds with return value 0
Press any key to continue . . .
```

Space Complexity: O(N)

Time Complexity: O(N log N)

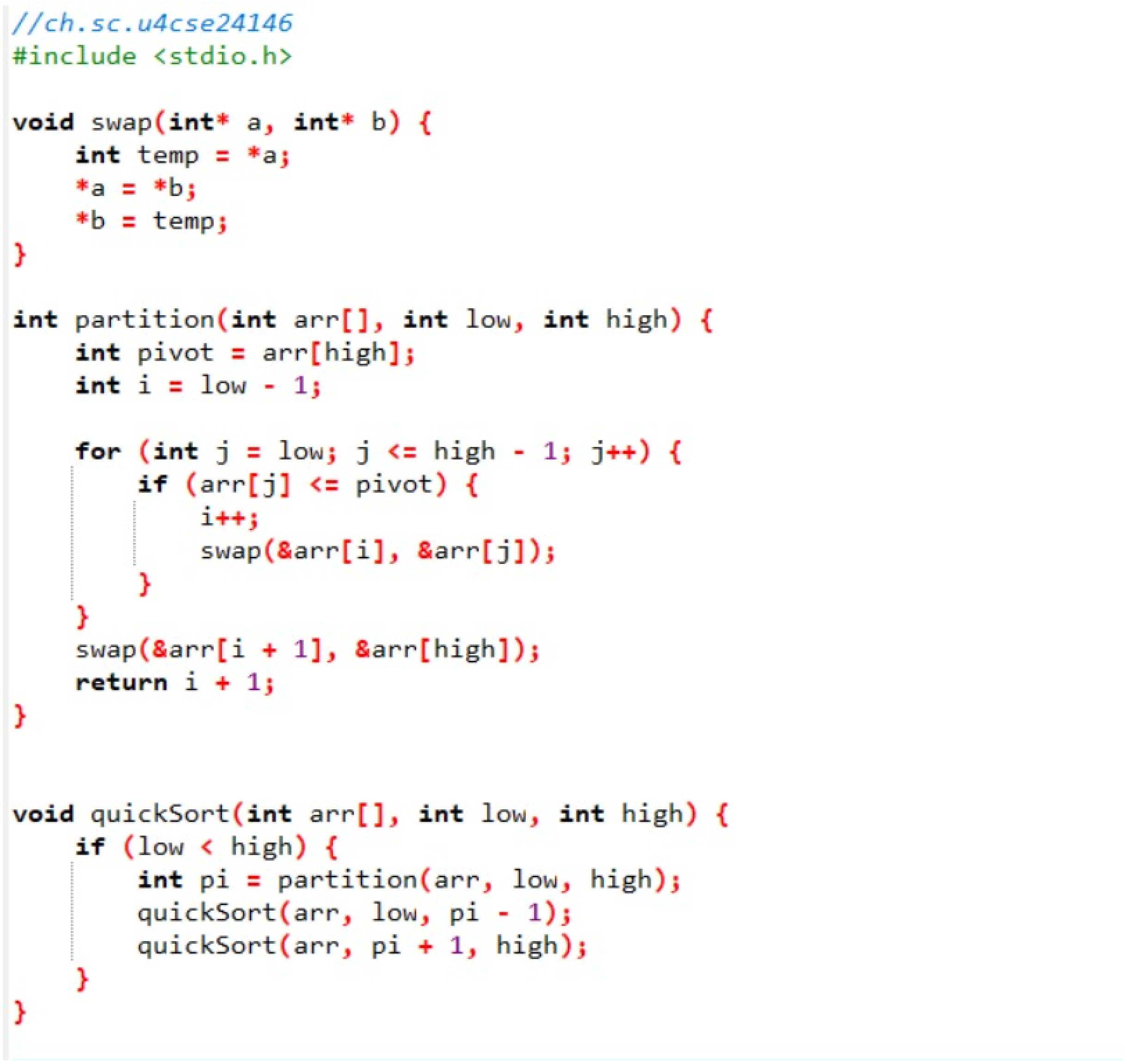
Justification:

- Merge sort divides the array into halves recursively, creating log N levels of recursion. At each level, the merge operation processes N elements, leading to a total time complexity of O(N log N)

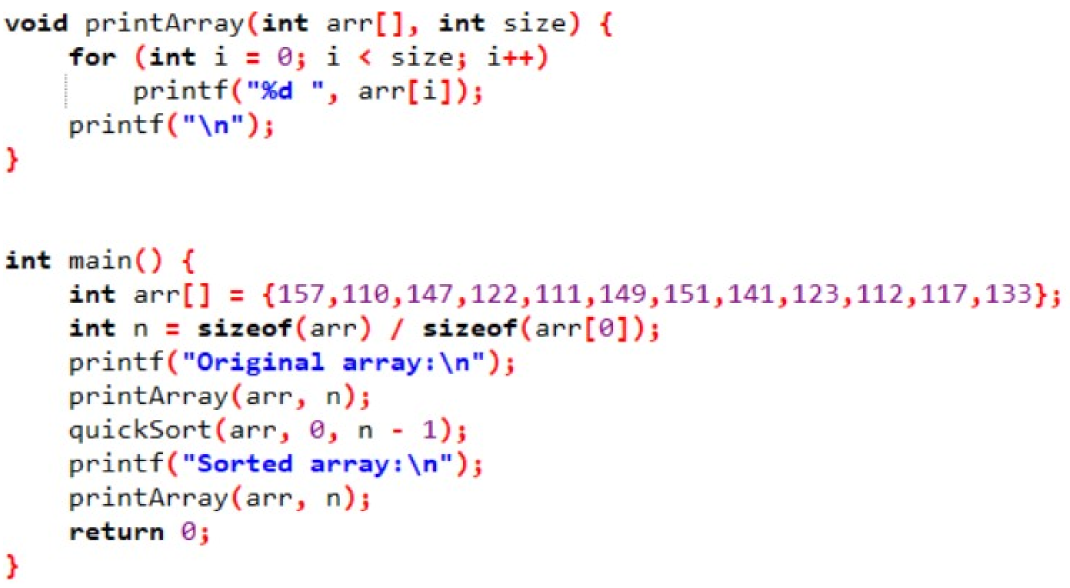
=> temporary auxiliary arrays are used to store elements, which require O(N) extra space; hence the space complexity is O(N).

Question 2: Create a C program that implements Quick Sort to sort an array of integers.

CODE:

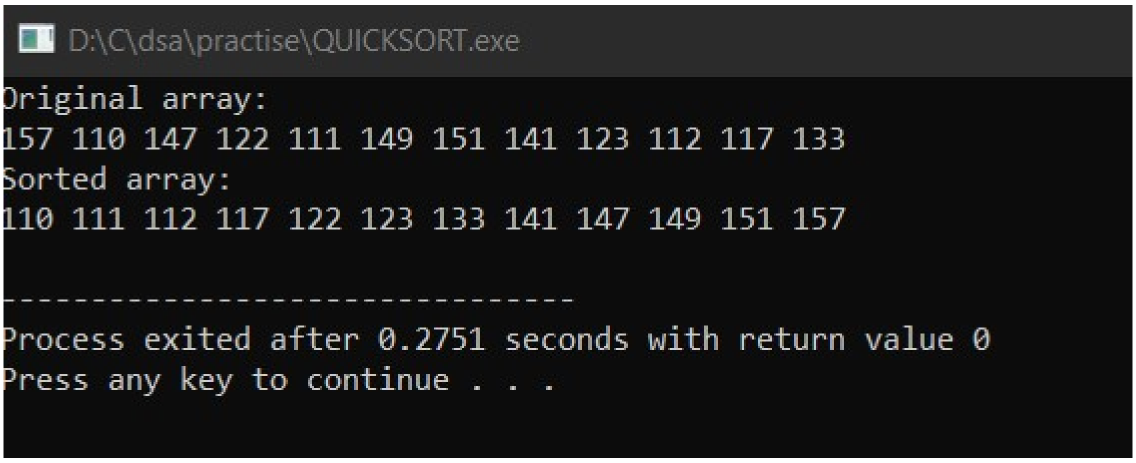


```c
//ch.sc.u4cse24146
#include <stdio.h>

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

int partition(int arr[], int low, int high) {
    int pivot = arr[high];
    int i = low - 1;

    for (int j = low; j <= high - 1; j++) {
        if (arr[j] <= pivot) {
            i++;
            swap(&arr[i], &arr[j]);
        }
    }
    swap(&arr[i + 1], &arr[high]);
    return i + 1;
}


void quickSort(int arr[], int low, int high) {
    if (low < high) {
        int pi = partition(arr, low, high);
        quickSort(arr, low, pi - 1);
        quickSort(arr, pi + 1, high);
    }
}
```



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


int main() {
    int arr[] = {157,110,147,122,111,149,151,141,123,112,117,133};
    int n = sizeof(arr) / sizeof(arr[0]);
    printf("Original array:\n");
    printArray(arr, n);
    quickSort(arr, 0, n - 1);
    printf("Sorted array:\n");
    printArray(arr, n);
    return 0;
}
```

OUTPUT:



D:\C\dsa\practise\QUICKSORT.exe

```
Original array:
157 110 147 122 111 149 151 141 123 112 117 133
Sorted array:
110 111 112 117 122 123 133 141 147 149 151 157

--------------------------------
Process exited after 0.2751 seconds with return value 0
Press any key to continue . . .
```

Space Complexity: O(N)

Time Complexity: O(N^2)

Justification:

- the recursion depth becomes n, and the recursion stack stores up to n function calls, leading to **O(N) Space Complexity**.
- This occurs when the pivot divides the array into highly unbalanced parts, such as in already sorted or reverse sorted array. Hence the **Time Complexity** is **O(N^2)**.