Shaheed Zulfikar Ali Bhutto Institute of Science & Technology

COMPUTER SCIENCE DEPARTMENT

Total Marks: 7.5

Obtained Marks: ________

DATA STRUCTURE AND ALGORITHM

Lab Report # 09

Submitted To: **Mam Tehreen**

Submitted By: _____**Hammad Qureshi** ___________________.

Reg. Numbers: **2112114**

Question no 1:

a. **Write a function to sort array elements using quick sort**
b. **Write a function to sort array elements using merge sort**
c. **Write a function to sort array elements using Insertion Sort**

Code:

Part(a)

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

void swap(int arr[] , int pos1, int pos2){
	int temp;
	temp = arr[pos1];
	arr[pos1] = arr[pos2];
	arr[pos2] = temp;
}

int partition(int arr[], int low, int high, int pivot){
	int i = low;
	int j = low;
	while( i <= high){
		if(arr[i] > pivot){
			i++;
		}
		else{
			swap(arr,i,j);
			i++;
			j++;
		}
	}
	return j-1;
}

void quickSort(int arr[], int low, int high){
	if(low < high){
	int pivot = arr[high];
	int pos = partition(arr, low, high, pivot);

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

int main()
{
	int n ;
	cout <<"Enter the size of array ";
	cin>>n;
	int arr[n];
	for( int i = 0 ; i < n; i++){
		cin>> arr[i];
	}
	quickSort(arr, 0 , n-1);
```

```
        cout<<"The sorted array is: ";
        for( int i = 0 ; i < n; i++){
                cout<< arr[i]<<" ";
        }

}
```

Part(c)

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

// Function to sort an array using
// insertion sort
void insertionSort(int arr[], int n)
{
    int i, key, j;
    for (i = 1; i < n; i++)
    {
        key = arr[i];
        j = i - 1;

        // Move elements of arr[0..i-1],
        // that are greater than key, to one
        // position ahead of their
        // current position
        while (j >= 0 && arr[j] > key)
        {
            arr[j + 1] = arr[j];
            j = j - 1;
        }
        arr[j + 1] = key;
    }
}

// A utility function to print an array
// of size n
void printArray(int arr[], int n)
{
    int i;
    for (i = 0; i < n; i++)
        cout << arr[i] << " ";
    cout << endl;
}

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

    insertionSort(arr, N);
    printArray(arr, N);

    return 0;
}
```

Part(c)

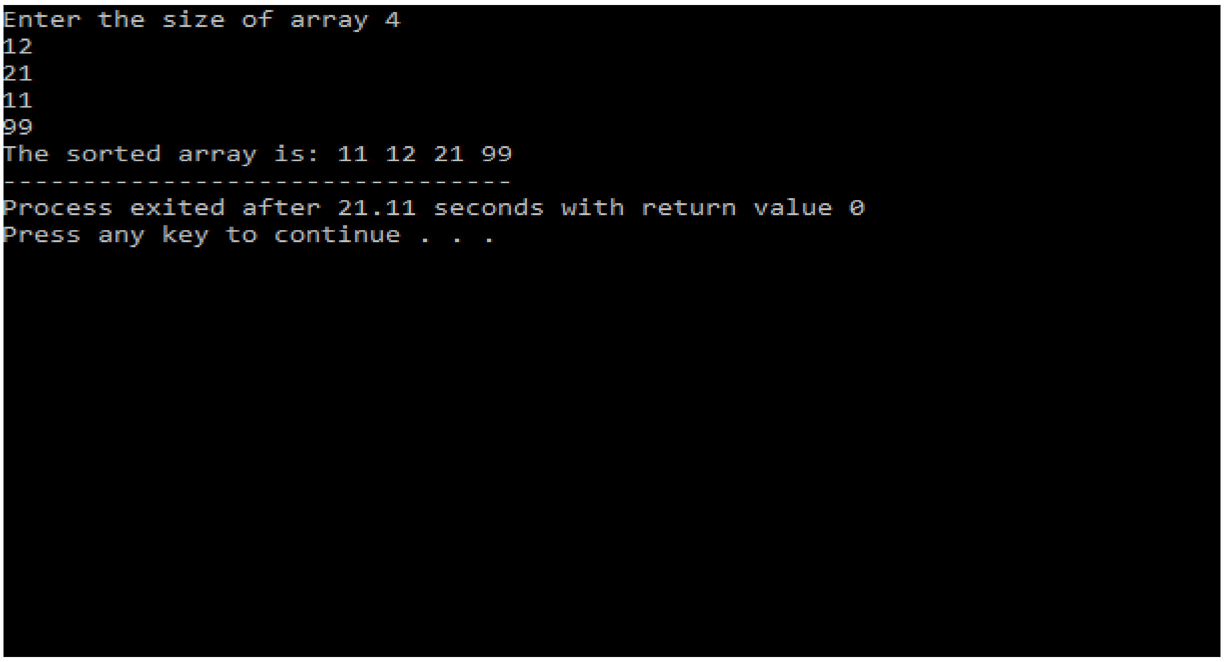
```
#include <iostream>
using namespace std;
```

```
// Merges two subarrays of array[].
// First subarray is arr[begin..mid]
// Second subarray is arr[mid+1..end]
void merge(int array[], int const left, int const mid,
         int const right)
{
   auto const subArrayOne = mid - left + 1;
   auto const subArrayTwo = right - mid;

   // Create temp arrays
   auto *leftArray = new int[subArrayOne],
        *rightArray = new int[subArrayTwo];

   // Copy data to temp arrays leftArray[] and rightArray[]
   for (auto i = 0; i < subArrayOne; i++)
      leftArray[i] = array[left + i];
   for (auto j = 0; j < subArrayTwo; j++)
      rightArray[j] = array[mid + 1 + j];

   auto indexOfSubArrayOne
      = 0, // Initial index of first sub-array
      indexOfSubArrayTwo
      = 0; // Initial index of second sub-array
   int indexOfMergedArray
      = left; // Initial index of merged array

   // Merge the temp arrays back into array[left..right]
   while (indexOfSubArrayOne < subArrayOne
          && indexOfSubArrayTwo < subArrayTwo) {
      if (leftArray[indexOfSubArrayOne]
          <= rightArray[indexOfSubArrayTwo]) {
          array[indexOfMergedArray]
              = leftArray[indexOfSubArrayOne];
          indexOfSubArrayOne++;
      }
      else {
          array[indexOfMergedArray]
              = rightArray[indexOfSubArrayTwo];
          indexOfSubArrayTwo++;
      }
      indexOfMergedArray++;
   }
   // Copy the remaining elements of
   // left[], if there are any
   while (indexOfSubArrayOne < subArrayOne) {
      array[indexOfMergedArray]
          = leftArray[indexOfSubArrayOne];
      indexOfSubArrayOne++;
      indexOfMergedArray++;
   }
   // Copy the remaining elements of
   // right[], if there are any
   while (indexOfSubArrayTwo < subArrayTwo) {
      array[indexOfMergedArray]
          = rightArray[indexOfSubArrayTwo];
      indexOfSubArrayTwo++;
      indexOfMergedArray++;
    }
    delete[] leftArray;
    delete[] rightArray;
}

// begin is for left index and end is
// right index of the sub-array
// of arr to be sorted */
void mergeSort(int array[], int const begin, int const end)
{
    if (begin >= end)
        return; // Returns recursively

    auto mid = begin + (end - begin) / 2;
    mergeSort(array, begin, mid);
    mergeSort(array, mid + 1, end);
    merge(array, begin, mid, end);
}

// UTILITY FUNCTIONS
// Function to print an array
void printArray(int A[], int size)
{
    for (auto i = 0; i < size; i++)
        cout << A[i] << " ";
}

// Driver code
int main()
```

```
{
    int arr[] = { 12, 11, 13, 5, 6, 7 };
    auto 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;
}
```

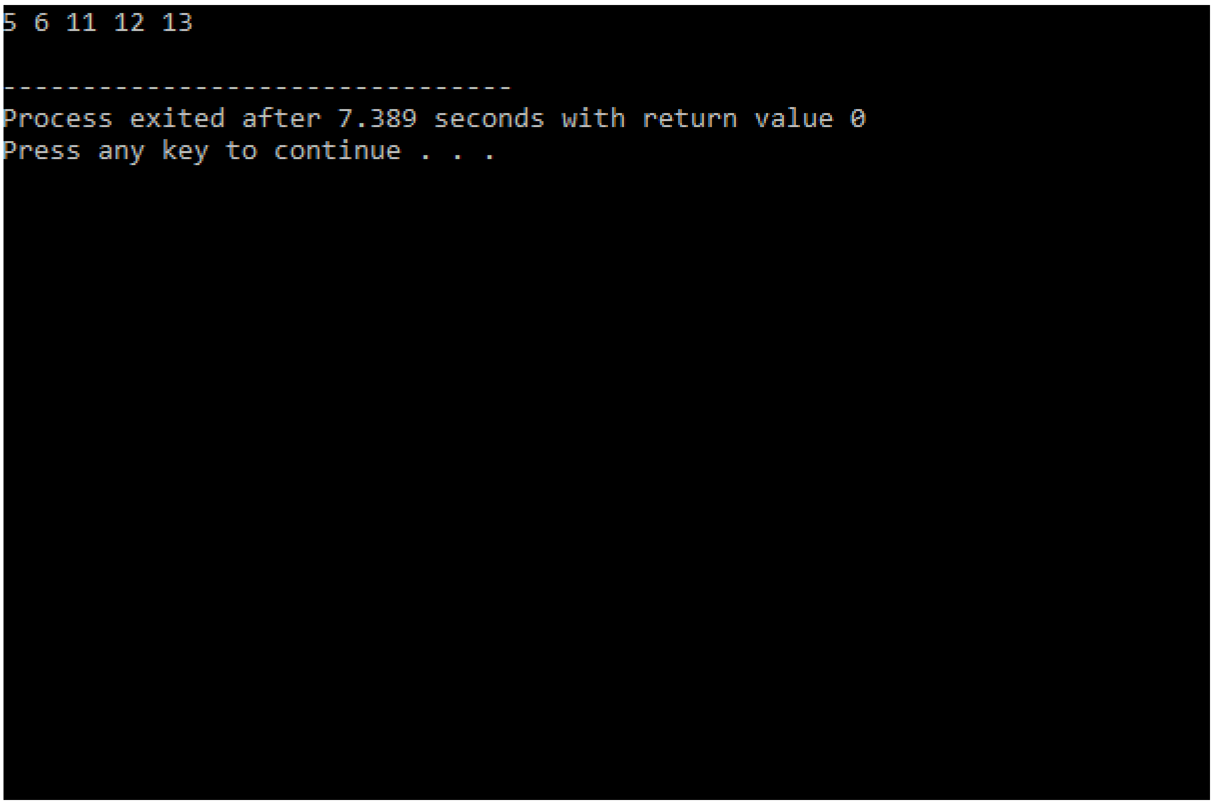
CONSOLE SCREEN:

Part(a)



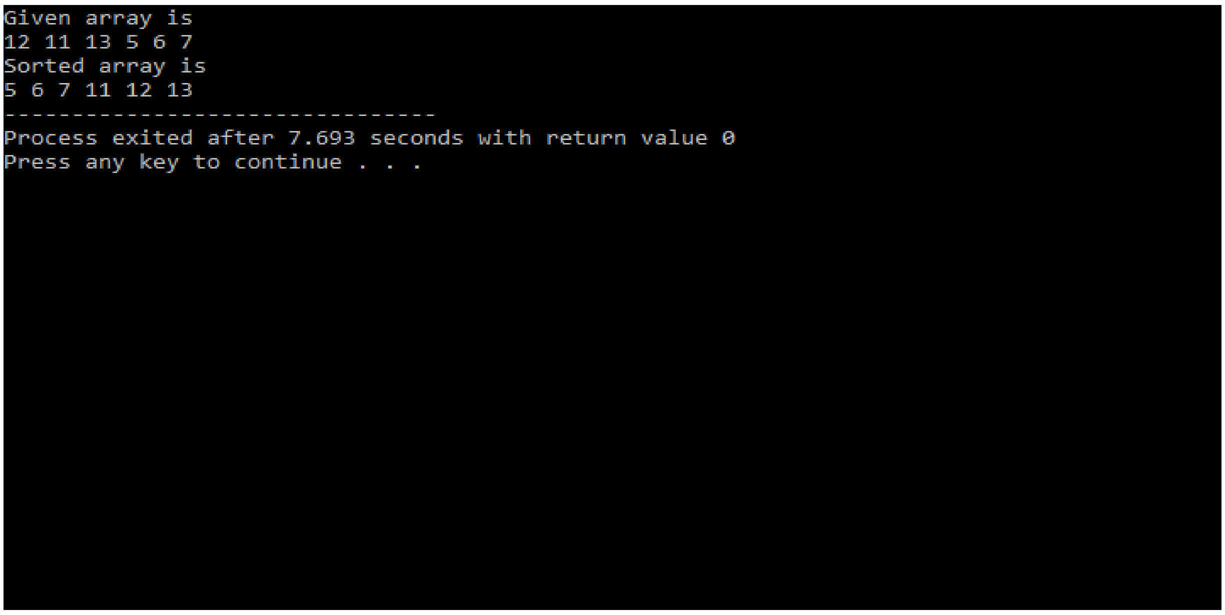
```
Enter the size of array 4
12
21
11
99
The sorted array is: 11 12 21 99
--------------------------------
Process exited after 21.11 seconds with return value 0
Press any key to continue . . .
```

Part©



```
5 6 11 12 13

--------------------------------
Process exited after 7.389 seconds with return value 0
Press any key to continue . . .
```

Part(b)



```
Given array is
12 11 13 5 6 7
Sorted array is
5 6 7 11 12 13
--------------------------------
Process exited after 7.693 seconds with return value 0
Press any key to continue . . .
```