Week-1
Practice lab – Algorithms and Problem Solving (15B17CI471)

Q-1 - We are given an array of n distinct numbers; where n is large numbers are randomly generated.
The task is to
a) Sort the entire array using selection sort, bubble sort, insertion sort, quick sort and merge sort. Print the total number of comparisons done in each of the sorting algorithm.

Example-

Input: arr[] = {2, 1, 5, 4, 3}
Output: arr[] = {1, 2, 3, 4, 5} Sol-

// Instertion sort :

Code:

```
int insertionSort(int arr[], int n) // function to sort an array with insertion sort

{       int i, j, temp; int

        count = 0 ;

        for (i = 1; i < n; i++) { temp = arr[i];

                j = i – 1;

                while(j>=0 && temp <= arr[j])                //Move the elements greater than temp to one position
ahead from their current position

                {       count++;

                        arr[j+1] = arr[j]; j = j-1; }

                arr[j+1] = temp;}

      return count;
      }
```

Best case : 0

Avg case : 4

Worst case: 10

```
// Selection sort


int selection(int arr[], int n)

{       int count = 0; int i, j,

        small;

        for (i = 0; i < n-1; i++)                    // One by one move boundary of unsorted subarray { small = i;

                //minimum element in unsorted array

                for (j = i+1; j < n; j++)

                if (arr[j] < arr[small]){ count++; small = j;}

// Swap the minimum element with the first element int temp =

        arr[small];

        arr[small] = arr[i]; arr[i] =

        temp;

        }

        return count;

        }
```

Sol:

Best case : 0

Avg case : 3

Worst case: 6

```
// bubble sort

  int bubble(int a[], int n) // function to implement bubble sort

  {

  int i, j, temp;

  int count = 0;

      for(i = 0; i < n; i++){
        for(j = i+1; j < n; j++)
          {
              if(a[j] < a[i])
              {     count++; temp
                    = a[i]; a[i] = a[j];
                    a[j] = temp;
              }
          }
      }
      return count;
 }
```

Comparisions:

Best case : 10

Avg case : 4

Worst case: 6

Merge Sort

```
int comp=0;
void merge(int array[], int const left, int const mid, int const right)
{
     int const subArrayOne = mid - left + 1; int const
     subArrayTwo = right - mid;

     int *leftArray = new int[subArrayOne],
          *rightArray = new int[subArrayTwo];

     comp++;
     for (int i = 0; i < subArrayOne; i++)
     {
          comp++;
          leftArray[i] = array[left + i];
     }
     comp++;
```

```
        for (int j = 0; j < subArrayTwo; j++)
        {
                comp++;
                rightArray[j] = array[mid + 1 + j];
        }
        int indexOfSubArrayOne = 0,
                indexOfSubArrayTwo = 0;
        int indexOfMergedArray = left; comp++;
        while (indexOfSubArrayOne < subArrayOne && indexOfSubArrayTwo < subArrayTwo)
        {
                comp++;
                if (leftArray[indexOfSubArrayOne] <= rightArray[indexOfSubArrayTwo])
                {
                        comp++;
                        array[indexOfMergedArray] = leftArray[indexOfSubArrayOne];
                        indexOfSubArrayOne++;
                }
                else
                {
                        array[indexOfMergedArray] = rightArray[indexOfSubArrayTwo];
                        indexOfSubArrayTwo++;
                }
                indexOfMergedArray++;
                comp++;
        }

        comp++;
        while (indexOfSubArrayOne < subArrayOne)
        {
                comp++;
                array[indexOfMergedArray] = leftArray[indexOfSubArrayOne]; indexOfSubArrayOne++;
                indexOfMergedArray++;
        }

        comp++;
        while (indexOfSubArrayTwo < subArrayTwo)
        {
                comp++;
                array[indexOfMergedArray] = rightArray[indexOfSubArrayTwo];
                indexOfSubArrayTwo++;
                indexOfMergedArray++;
        }
        delete[] leftArray; delete[]
        rightArray;
}
```

```
void mergeSort(int array[], int const begin, int const end)
{
        comp++;
        if (begin >= end) return;
        int mid = begin + (end - begin) / 2;
        mergeSort(array, begin, mid);
        mergeSort(array, mid + 1, end); merge(array,
        begin, mid, end);
}
void mergeSort1(int *input, int size)
{
        mergeSort(input, 0, size - 1); int comp
        = 0;
        cout << "Total number of comp are:" << comp << endl;
}
```

Comparisions:

Best case : 67

Avg case : 65

Worst case: 58

Quick sort

```
int comparisions = 0;

int partition(int input[], int start, int end, int flag)

{

        if (flag == 0)

        {

                int b = input[start]; int c = 0;

                comparisions++;

                for (int i = start + 1; i <= end; i++)

                {

                        comparisions++;   if

                        (input[i] <= b)

                        {

                                comparisions++; c++;
                comparisions++;

        }

        c = start + c;

        int     temp     =     input[start];

        input[start] = input[c]; input[c]

        = temp;

        int i = start; int j =

        end;

        comparisions++;

        while (i < c && j > c)

        {
        comparisions++;

if (input[i] <= b)

                {
          comparisions++; i++;

                }
                                        else if (input[j] > b)

                {
         comparisions++;

                        j--;

                        }
                else

                {

                        int temp = input[i]; input[i]

                        = input[j]; input[j] = temp;

                        i++;

                        j--;

                }

                comparisions++;

                comparisions++;

        }

        return c;

}

else

{

        int b = input[start]; int c = 0;

        comparisions++;

        for (int i = start + 1; i <= end; i++)

        {

                comparisions++;   if

                (input[i] >= b)

                {

                        comparisions++; c++;

                }

                comparisions++;

        }
```

```
c = start + c;

int     temp     =     input[start];

input[start] = input[c]; input[c]

= temp;

int i = start; int j =

end;

comparisions++;

while (i < c && j > c)

{

        comparisions++;   if

        (input[i] >= b)

        {

                comparisions++; i++;

        }

        else if (input[j] > b)

        {

                comparisions++; j-

                -;

        }

        else

        {

                int temp = input[i]; input[i]

                = input[j]; input[j] = temp;

                i++;

                j--;

        }
```

```
                        comparisions++;

                        comparisions++;

                }

                return c;

        }

}

void quickSort(int input[], int start, int end, int flag)

{


        comparisions++; if

        (start >= end)

        {

                return;

        }

        int a = partition(input, start, end, flag); quickSort(input, start, a - 1,

        flag); quickSort(input, a + 1, end, flag);

}

void quickSort1(int input[], int size, int flag)

{

        quickSort(input, 0, size - 1, flag);

        cout << "Total number of comparisions are:" << comparisions << endl;

}
```

Best case : 37

Avg case : 34

Worst case: 43

k) ***Write a function to sort all even-placed numbers in increasing and odd-place numbers in decreasing***

order. The modified array should contain all sorted even-placed numbers followed by reverse sorted

odd-placed numbers. Analyse the complexity of your implemented approach Note that the first element

is considered as even because of its index 0. Ans: Input: arr[]

= {0, 1, 2, 3, 4, 5, 6, 7}

Output: arr[] = {0, 2, 4, 6, 7, 5, 3, 1}

Input: arr[] = {3, 1, 2, 4, 5, 9, 13, 14, 12}

Output: 1, 2, 4, 6, 7, 5, 3, 1

Even-place elements : 3, 2, 5, 13, 12

Odd-place elements : 1, 4, 9, 14

Even-place elements in increasing order : 2, Odd- *3, 5, 12, 13*

Place elements in decreasing order : 14, Code: *9, 4, 1*

#include &lt;vector&gt; #include

&lt;algorithm&gt; using

namespace std;

void helper(int *arr,int n){ vector&lt;int&gt;v1;

    vector&lt;int&gt;v2;

    //int *arr1 = new int[size/2];

    //int *arr2 = new int[size/2];

```
        for(int i = 0;i&lt;n;i++){ if(i%2 == 0
                || i == 0){
                        v1.push_back(arr[i]);
                }
                else{
                        v2.push_back(arr[i]);
                }
        }
        sort(v1.begin(),v1.end());
        sort(v2.begin(),v2.end());
        reverse(v2.begin(),v2.end()); int k = 0;
        while(k &lt; v1.size()){
                arr[k] = v1[k]; k++;
        }
        int j = 0; while(j&lt;v2.size()){
                arr[k] = v2[j];


                k++;j++;
        }
}
int main(){
        int arr[] = {2,3,8,-1,7,10};
        int size = sizeof(arr)/sizeof(arr[0]); helper(arr,size);
        for(int i = 0;i&lt;size;i++){ cout&lt;&lt;arr[i]&lt;&lt;&quot;&quot;;
        }
}
```

Time complexity: O(nlogn) Space complexity: O(n)

1) ***Given an integer array of which both first half and second half are sorted. Task is to merge two***

sorted halves of array into single sorted array. Analyse the complexity of your implemented approach

Example:

Input : A[] = { 2 ,3 , 8 ,-1 ,7 ,10 }

Output : -1 , 2 , 3 , 7 , 8 , 10

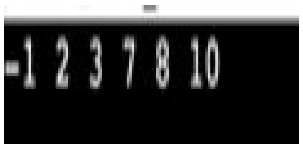
Input : A[] = {-4 , 6, 9 , -1 , 3 }

Output : -4 , -1 , 3 , 6 , 9

Ans:

```
#include <iostream>


using namespace std;


void merge(int *arr1, int size1, int *arr2, int size2)

{

        int [size1+size2];

   int i=0,j=0,k=0;

        int l=size1+size2; if(size1!=0 &&

        size2!=0) while(k<l)

        {

            if(arr1[i]<=arr2[j] && i<size1)

        {

                [k]=arr1[i]; k++;

                i++;
                if(i==size1)
                {
```

```
                              for(int p=j;p<size2;p++)

                              {

                                    [k]=arr2[p]; k++;

                              }
                    }
          }
          else if(arr1[i]>arr2[j] && j<size2)

          {
                                                  [k]=arr2[j];

                    k++;

                    j++;

if(j==size2)
{
for(int p=i;p<size1;p++)

                              {

                                    [k]=arr1[p]; k++;

                                    }}

                    }

          }
          else if(size1==0){ for(int

          j=0;j<size2;j++){


                    [j]=arr2[j];

                    }
          }

          else if(size2==0){ for(int

          j=0;j<size1;j++){

                    [j]=arr1[j];

}
          }

else{ [0]='\
          0';

          }

          for(intj=0;j<(size1+size2);j++){ if(j<size1)

          arr1[j]=[j]; else

          arr2[j-size1]=[j];

          }

}

int main()

{

      int input[]={2,3,8,-1,7,10};

      int arr[6]; merge(input,3,input+3,3);

      for(int j=0;j<6;j++){

      cout<<input[j]<<" ";


      }
          return 0;
          }
```



```
-1 2 3 7 8 10
```