K. J. Somaiya College of Engineering, Mumbai-77
(A constituent College of Somaiya Vidyavihar University)

Batch: B2	**Roll. No.:** 16010121110
Experiment:	
Grade: AA / AB / BB / BC / CC / CD /DD	

Title: Implementation of sorting algorithms

Objective: To understand various searching methods

Expected Outcome of Experiment:

CO	Outcome
CO3	Demonstrate sorting and searching methods.

Websites/books referred:

In text citations given.

Abstract: -

(Define sorting as a concept,)

Sorting refers to ordering data in an increasing or decreasing manner according to some linear relationship among the data items. ordering: arranging items in a sequence ordered by some criterion; categorizing: grouping items with similar properties.

ref-https://en.wikipedia.org/wiki/Sorting#:~:text=Sorting%20refers%20to%20ordering%20data,grouping%20items%20with%20similar%20properties.

Related Theory: *(Explain stable sort, in place sort, number of passes in a sorting algo)*

Stable means the order of input elements is unchanged except where change is required to satisfy the requirements. A stable sort applied to a sequence of equal elements will not change their order.

In-place means that the input and output occupy the same memory storage space. There is no copying of input to output, and the input ceases to exist unless you have made a backup copy. This is a property that often requires an imperative language to express, because pure functional languages do no have a notion of storage space or overwriting data.

ref- https://stackoverflow.com/questions/20375482/is-stable-and-in-place-the-same

Assigned Sorting Algorithm: *(Bubble/insertion/counting)*

1. *countingSort(array, n) // 'n' is the size of array*
2. *max = find maximum element in the given array*
3. *create count array with size maximum + 1*
4. *Initialize count array with all 0's*

```
5.  for i = 0 to n
6.  find the count of every unique element and
7.  store that count at ith position in the count array
8.  for j = 1 to max
9.  Now, find the cumulative sum and store it in count array
10. for i = n to 1
11. Restore the array elements
12. Decrease the count of every restored element by 1
13. end countingSort
```

ref - https://www.javatpoint.com/counting-sort

Code and output screenshots for assigned sorting algorithm:

```
#include <iostream>


using namespace std;


int main()

{   int len=5;

int range=5;

   int unsortArray[len+1];

   for(int i=0;i<=len;i++){

      cin>>unsortArray[i];

   }

    int countArray[range]={0};


    //update count array

    for(int i=0;i<=len;i++){

   countArray[unsortArray[i]]++;

   }


    //Cumulative addition

    for(int i=1;i<=len;i++){

   countArray[i]+=countArray[i-1];

   }

   //rightshift

    for(int i=len;i>0;i--){
```

```
   countArray[i]=countArray[i-1];

   }


   //sortArray

   int sortArray[len]={};

    for(int i=0;i<=len;i++){

        int temp = unsortArray[i];

   sortArray[countArray[temp]]=temp;

    countArray[temp]++;

   }


   cout<<"Hello World";

 for(int i=0;i<=len;i++){

   cout<<"\n"<<sortArray[i];

   }

   return 0;

}
```

```
3

2

1

3

3

2
```

Hello World

1

2

2

3

3

3

Bubble sort
https://github.com/Aatmaj-Zephyr/Solutions-to-first-year-practicals/tree/main/C/Practicals/Expt%205

Conclusion: -
Thus we have learned different types of sorting. We learned insertion sort, counting sort and bubble sort. We got idea of the implementation of the sorting techniques. Sorting techniques are very important and affect the efficiency of the entire system.

Post lab questions-

a. **Compare and contrast various sorting algorithms.**

Sorting Algorithm	Average Case	Best Case	Worst Case
Bubble Sort	O(n2)	O(n)	O(n2)
Insertion Sort	O(n2)	O(n)	O(n2)
Selection Sort	O(n2)	O(n2)	O(n2)
Quick Sort	O(n.log(n))	O(n.log(n))	O(n2)
Merge Sort	O(n.log(n))	O(n.log(n))	O(n.log(n))
Heap Sort	O(n.log(n))	O(n.log(n))	O(n.log(n))
Counting Sort	O(n+k)	O(n+k)	O(n+k)
Radix Sort	O(n*k)	O(n*k)	O(n*k)
Bucket Sort	O(n+k)	O(n+k)	O(n2)

ref - https://www.codeproject.com/Articles/5308420/Comparison-of-Sorting-Algorithms

b. **Comment on the input (sorted in ascending order/descending order/random) and time required for execution of sorting algorithm.**

Bubble sort is faster for random inputs while for duplicated inputs counting sort is bit more efficient.