Regd No.:- 19BQ1A05M3
Section :- CSE-D

D.S. ASSIGNMENT - 1

① Assume that there is a list [22, 22, 22, 22, 22, 22, 22]. What happens when selection sort is applied on the list ? Explain.

A) Selection Sort :- Selection sort is an algorithm that we select and search for the lowest element then the lower element is swapped with the current element.

Given Array,

22	22	22	22	22	22	22

min

Here no swap

22	22	22	22	22	22	22

min

Here no swap

22	22	22	22	22	22	22

min Here no swap

22	22	22	22	22	22	22

min
Here no swap

22	22	22	22	22	22	22

min

Here no swap

22	22	22	22	22	22	22

min

Here no Swap.

In the above list all the elements are same. So there will be no swappings at all.

Output :-

22	22	22	22	22	22	22

19BQ1A05M3

② Sort the following list using insertion sort:-

Varun Amar karthik Ramesh Bhuvan Dinesh Firoz Ganesh

A) Insertion Sort :- It is also a sorting algorithm. But it is more efficient because it replaces swappings with shiftings.

* Here every element is compared to it's previous elements. If we found any bigger element before the key then we shift their places.

Given Array,

Varun	Amar	karthik	Ramesh	Bhuvan	Dinesh	Firoz	Ganesh
0	Temp1	2	3	4	5	6	7

Varun > Amar

So shift Varun right and insert Amar at 0th position.

Amar Varun karthik Ramesh Bhuvan Dinesh Firoz Ganesh

temp

Varun > karthik (Insert karthik at 1st Position)

Amar karthik Varun Ramesh Bhuvan Dinesh Firoz Ganesh

temp

Varun > Ramesh

Shift Varun right and insert Ramesh at 2nd Position.

Amar karthik Ramesh Varun Bhvan Dinesh Firoz Ganesh

temp

Varun > Bhuvan, Ramesh > Bhuvan, karthik > Bhuvan

Shift karthik, Ramesh, Varun to right and insert Bhuvan at 1st Position

Amar Bhuvan karthik Ramesh Varun Dinesh Firoz Ganesh

temp

③ 15M3

Shift karthik, Ramesh, Varun to right and insert Dinesh at 2nd position

Amar Bhuvan Dinesh karthik Ramesh Varun Firoz Ganesh
temp

Shift karthik, Ramesh, Varun to right and insert Firot at the 3rd position

Amar Bhuvan Dinesh Firoz karthik Ramesh Varun Ganesh
temp

Shift karthik, Ramesh, Varun to right and insert Ganesh to the 4th position.

Output:-

Amar	Bhuvan	Dinesh	Firoz	Ganesh	karthik	Ramesh	Varun

This is the sorted list.

③ Sort the following numbers using Quick Sort.

67 54 9 21 12 65 56 43 34 79 70 45

A) Quick Sort:- It is based on Divide and Conqueor principle. Take first element of the list as pivot. Swappings are done until pivot element reaches it's correct position. Then again take two sub-lists and repeat the process until we get a sorted list.

Given Array:-

67	54	9	21	12	65	56	43	34	79	70	45
pivot											

Compare from right to left for smaller swap (67, 45).

5M3

45 54 9 21 12 65 56 43 34 79 70 67
pivot

Compare from left to right for largest element.

Swap (79, 67)

45 54 9 21 12 65 56 43 34 67 70 79
pivot

67 is at correct position.

Now divide right and left sublists.

L.S.L. :-

45 54 9 21 12 65 56 43 34
pivot

Compare from te Right to left

Swap (45, 34)

R.S.L. :-

70 79
pivot

Right to left no swap

70	79

34 54 9 21 12 65 56 43 45
pivot

Compare from left to right. Swap (54, 45)

34 45 9 21 12 65 56 43 54
pivot

Compare from right to left. Swap (45, 43)

34 43 9 21 12 65 56 45 54
pivot

Left to right. Swap (65, 45)

34 43 9 21 12 45 56 65 54
pivot

45 is at the correct position.

Now divide right and left sub lists.

L.S.L.

34 43 9 21 12
pivot

R→L Swap (34, 12)

12 43 9 21 34 pivot

L→R Swap (43, 34)

R.S.L.

56 65 54
pivot

R→L Swap (56, 54)

54 65 56 pivot

L→R Swap (65, 56)

19BQ1A05M3

5

54	56	65

12 34 9 21 43
pivot
R→L swap(34, 21)
12 21 9 34 43
pivot
34 is at the correct position. So divide it in to sub lists.

L.S.L.

12 21 9
pivot
R→L swap(12, 9)
9 21 12
pivot
L→R swap (21, 12)

9	12	21

R.S.L.

43

So the final Sorted list is:-

9	12	21	34	43	45	54	56	65	67	70	79

4 Implement linear and binary search using recurssion.

A.) Linear Search:-

```
public class Test{
        Static int arr[]={12, 34, 54, 12, 3};
  // Recurssive method to search X in arr[I,----,r]
  Static int recSearch(int arr[], int I, int r, int x)
  { if(r < I)
          return -1;
    if(arr[I]==x)
          return I;
if(arr[r] = = x)
        return r;
    return recSearch(arr, I+1, r-1, x);
}
// Driver method
public Static void main(String args[]) {
        int x=3;
    //Method call to find x.
        int index= recSearch(arr, 0, arr.length-1, x);
        if(index != -1)
             System.out.println("Element"+x+"is at index"+index);
        else
             System.out.println("The element is not found");
  }
}
```

Output:-

```
Element 3 is at index 4.
```

Binary Search:-

```
public class BinarySearch{
        int binarySearch(int arr[], int I, int r, int x)
        {   if(r>=I)
                int mid=I+(r-1)/2;
          if(arr[mid]==x)
                return mid;
          if(arr[mid]>x)
                return binarySearch(arr, I, mid-1, x);
        } return(binarySearch(arr, mid+1, r, x));
```

```
    return -1;
  }
  public static void main(String args[]){
        Binary search ab = new Binarysearch();
        int arr[] = {2, 3, 4, 10, 40};
        int n = arr.length();
        int x = 10;
        int result = ob.binary search(arr, 0, n-1, x);
        if(result = = -1)
            System.out.println("Element is not found");
        else
            System.out.println("Element found at index"+ result);
    }
  }
```

Output:-

```
Element found at index 3.
```

⑤ Explain in brief about the various factors that determine the selection of an algorithm to solve a computational problem.

A) ① Time Complexity:- Time Complexity of an algorithm quantifies the amount of time taken by the algorithm to solve a problem.

* It is based on processor, clock speed, OS etc. There are 3 types of time Complexities:-

i) Best Case (Omega Notation):-

The minimum no. of steps takes to solve a problem.

ii) Average Case (Theta Notation):-

The average no. of steps taken to solve a problem.

iii) Worst Case (Big O Notation):- 19BQ1A05M3

The maximum no. of steps taken to solve a problem.

* Usually Big O Notation is the most used one. Because algorith performance may vary different types of input data.

② Space Complexity:-

It represents the total amount of memory needed for an algorithm to solve a problem.

Space = fixed part + Variable part

It depends up on the process, hardware, OS etc.

Example:-

If we compare bubble Sort and Merge Sort. Bubble Sort requires less space compared to Merge Sort.

* So based on these two complexities we find a better algorithm to solve a computational problem.