D·S ASSIGNMENT-1

19BQ1A05D2
Lakshmi Niharika·M
CSE-IIC

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 & search for the lowest element. Then the lowest element is swapped with 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 are no swappings at all.

Output:

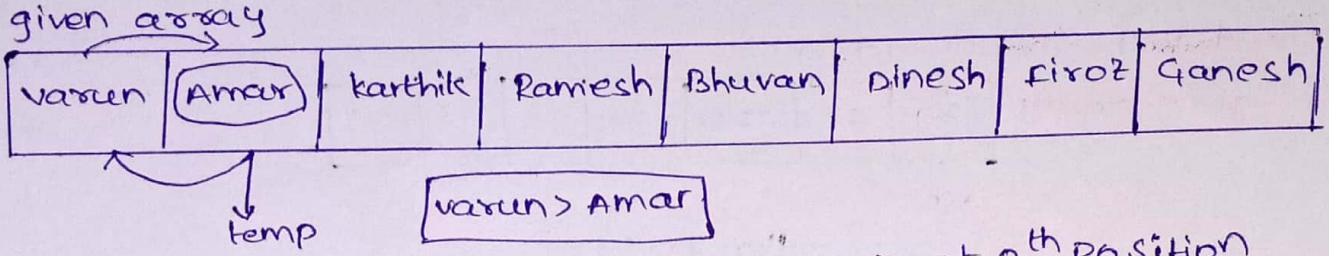
22	22	22	22	22	22	22

2) Sort the following list using insertion sort: varun Amar karthik Ramesh Bhuvan Dinesh Firoz Ganesh.

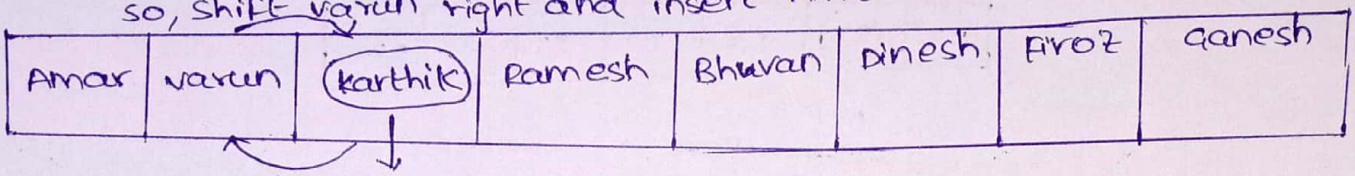
Ans: **Insertion Sort:** It is also a sorting algorithm. But it is more efficient because it Replaces sorting scoapping with shifting with shifting.

Here every element is compared to its previous element. If we found any bigger element before the key then we shift their places.

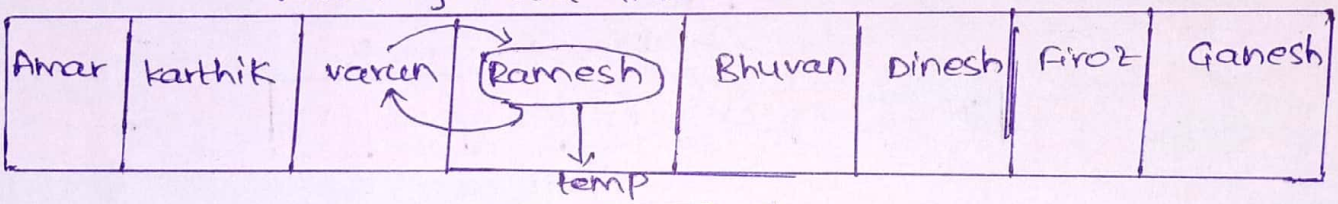
given array



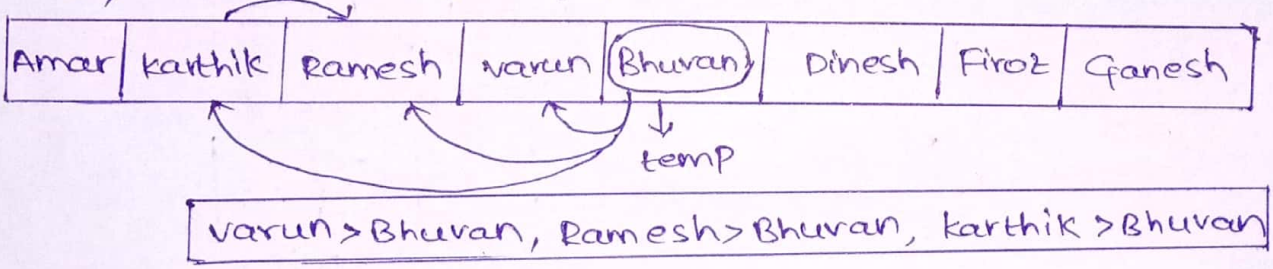
so, shift varun right and insert Amar at 0th position



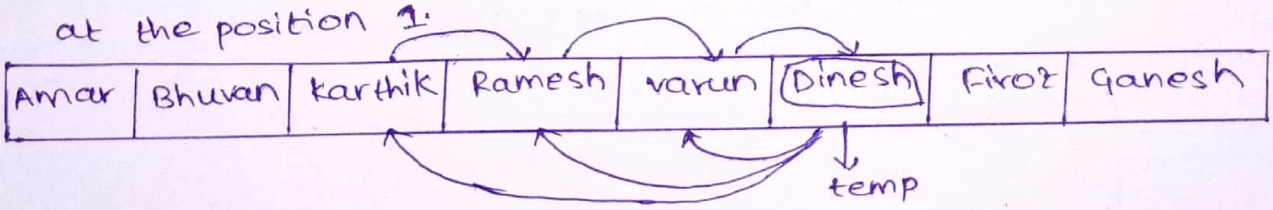
So, shift varun right and insert karthik at 1st position



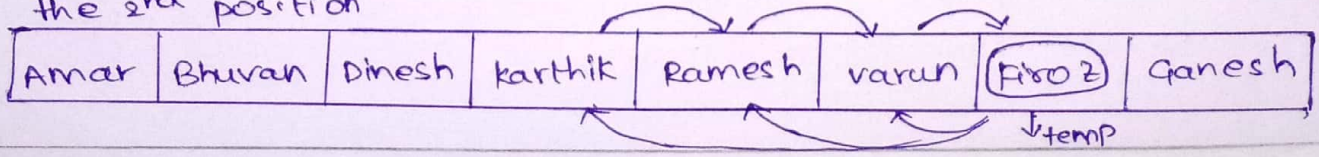
So, shift varun Right & insert Ramesh at 2nd position.



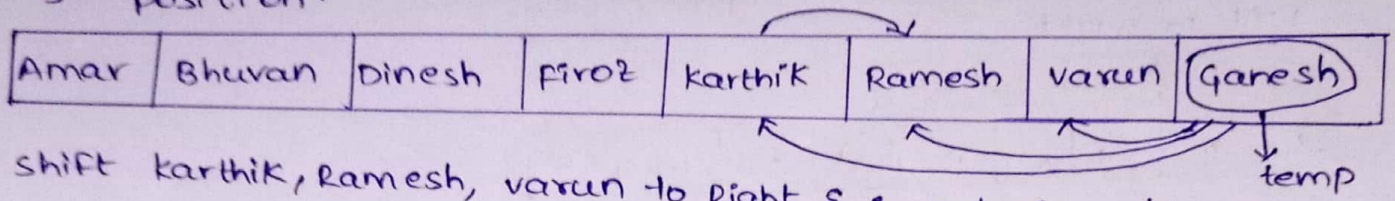
So shift karthik, ramesh, varun to right & insert Bhuvan at the position 1.



shift karthik, Ramesh, varun, to right and insert Dinesh at the 3rd position



shift karthik, Ramesh, varun to right and insert Firoz at 3rd position.



shift karthik, Ramesh, varun to Right & insert Ganesh at 4th pos.

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?

ANS: Quick sort: It is based on divide and conquer principle. Take first element of the list as pivot swappings are done untill pivot element reaches its correct position. Then again take the 2 sublists and repeat the process untill 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)

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

→ pivot

compare from left to Right for Bigger element. (swap (79, 67)

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

pivot.

67 is in the correct position.

Now divide left sublist & Right sublist

L.S.L

(45)	54	9	21	12	65	56	43	(34)

pivot

compare left from Right swap (45, 34)

34	(54)	9	21	12	65	56	43	(45)

→ pivot

left to right swap (54, 45)

34	(45)	9	21	12	65	56	(43)	54

pivot

Right to left swap (45, 43)

R.S.L

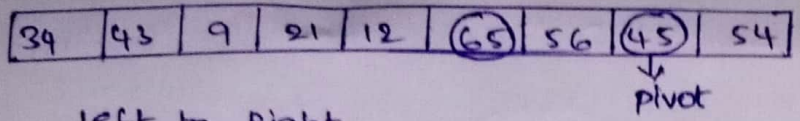
(70) 79

pivot

Right to left

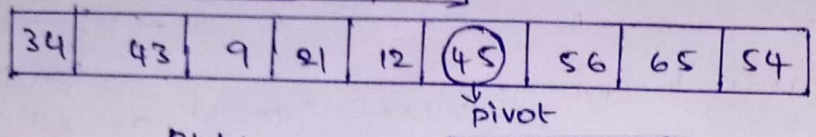
No swap

70 | 79



left to Right

Swap (65, 45)

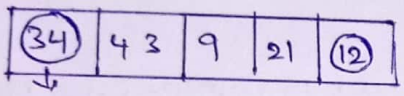


Right to left (No swap)

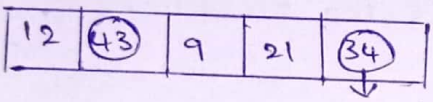
So 45 is in correct position

Now divide left sub list & Right sub list

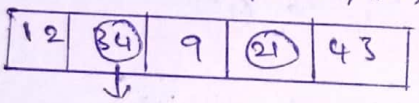
L.S.L



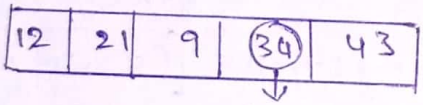
R → L swap (34, 12)



L → R (swap (34, 43))

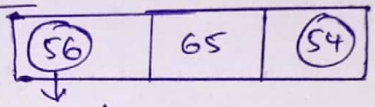


R → L (swap (34, 21))



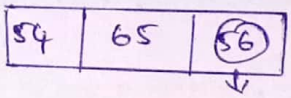
34 is in correct position

R.S.L



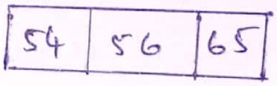
Right to left

Swap (56, 54)

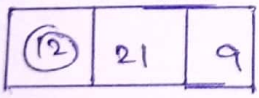


left to Right

Swap (65, 56)

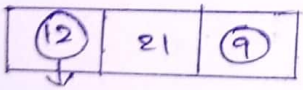


Left sub list

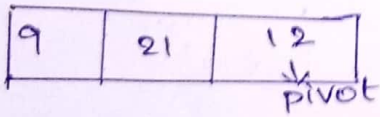


Right sub list

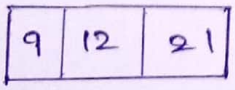
Left sub list



Right to left (swap 12 & 9)



left to Right (swap 21, 12)



The final sorted list is

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

4) Implement linear search & Binary using Recursion?

Ans: Linear search: It is used to find position of an element in the given list. It is also called as sequential search.

Algorithm: 1) Take a list of elements

2) compare the key with all the elements in the list sequentially.

Program:

```
import java.util.Scanner;
public class linear search
{
  public static void main(string args[])
    int a[], n, i, key, pos;
    scanner sc= new scanner (system.in);
    System.out.println ("Enter size of array");
    n=sc.nextInt(); a=new Int[n];
    System.out.println ("Enter elements of array");
    for (i=0; i<n; i++)
       a[i] = sc.nextInt();
    system.out.println("the array is: ");
    for(i=0; i<n; i++)
       System.out.print(a[i]+" ");
    System.out.println();
    //pos = linear search (a, 0, n-1, key)
    System.out.println ("Enter search key");
    key = sc.nextInt();
    pos = linear search (a, 0, n-1, key);
    if(pos==-1)
       System.out.println ("key not found !");
    else
       system.out.println("key found at: "+(pos+1));
}
public static int linear search (int a[], int lb, int ub, int key)
   {
      if (lb>ub)
        return -1;
      else if (a[lb]==key)
        return lb;
      else
        return linear search (a, lb+1, ub, key);
} }
```

Output: Enter size of array
5
Enter elements of array
5
9
12
14
15
The array is:
5 9 12 14 15
Enter search : 14
key found at 4.

* Binary search: It is also used to find position of an Element in the given list. It is based on divide & conquer principle.

It reduces no. of comparisions when compared to linear search

Algorithm:
1) Take array of elements.
2) Find mid position
3) If key is found at mid, return mid.
4) If key is greater than mid element and repeat the procedure (steps 1 to 3)
5) If key is less than mid element then take left sublist & repeat the procedure (step 1 to 3)

Program:

```
import java.util.Scanner;
public class binary search
{
  public static void main(string args[])
  {
   int a[], key, pos, n, i;
   Scanner sc = new scanner(system.in);
   System.out.println("Enter site of array");
   n = sc.nextInt();
   a = new int[n];
   System.out.println("Enter elements of array");
   for(i=0; i<n; i++)
      a[i] = sc.nextInt();
   System.out.println("The array is: ");
   for(i=0; i<n; i++)
      system.out.print(a[i]+" ");
   System.out.print();
```

```
        Sort (a,n);
        System.out.println("The sorted array is ");
        for (i=0; i<n; i++)
          System.out.print(a[i]+" ");
        System.out.println();
        System.out.println("Enter Search key");
        key = sc.nextInt();
        pos = binarysearch(a, 0, n-1, key);
        if (pos=-1)
          system.out.println("No key found");
        else
          system.out.println("key found at" + (pos+1));
    }
public static int binarysearch (int a[], int lb, int ub, int key)
  {
    int mid = 0;
    if (ub >= lb)
    {
      mid = lb + (ub - lb)/2;
      if(a[mid] == key)
        return mid;
    }
    if (a[mid] > key)
      return binarysearch(a, lb, mid-1, key);
    else
      return binarysearch(0, mid+1, ub, key);
  }
public static void (int a[], int b)
  {
    int i, j
    for (i=0; i<n-1; i++)
      for (j=0; j<n-i-1; j++)
        if (a[j] > a[j+1])
          {
            int temp = a[j];
            a[j] = a[j+1];
          }  a[j+1] = temp;
  }
}
```

Output:

```
Enter size of array
5
Enter elements of array  5 4 3 2 1
The array is
5 4 3 2 1
The sorted array is
1 2 3 4 5
Enter search key: 3
key found at 3
```

5) Explain breif, the various factors that determine the selection of any algorithm to solve a computational problem?

Ans: Analysis of algorithms is the determination of the amount of time & space resources required to solve a computational problem (or) any other. usually the efficiency (or) running time of an algorithm is stated as a function relating the Ip length to the number of steps, known as time complexity, or volume of memory, known as space complexity. By considering algorithm for a special problem. we can begin to develop pattern recognition. so that similar types of problems can be solved with the help of this algorithm.

Efficiency of Algorithm is measured in two different stages.

1) space complexity: It represents the total amount of memory needed for an algorithm to solve a problem.

Space = fixed post + variable part.

It depends on processor, hardware, OS, etc..

Ex: If we compare bubble sort & merge sort Bubble sort requires less space compared to merge sort.

2) Time complexity: The amount of time required to for an algorithm to solve a problem. It is mainly based on processor. clock, speed, OS, etc. There are 3 types of time complexities.

1) Best case (omega notation): The minimum number of steps taken to solve a problem.

2) Average case (Theta notation): The Average number of steps taken to solve a problem.

3) worst case (Big O notation): - The maximum number of steps taken to solve a problem.

Usually Big-O Notation is the most used one.

Because, algorithm perform may vary with different types of Ip data.

So Based on the two complexities are we find a better algorithm to solve a computational problem.