Tutorial 3

Name: Sonam
Sec: B
Roll no: 2014892.

Q1 pseudo code for linear Search.

```
int linear (int * arr[], int n, int key)
{ {   for(int i=0; i<n; i++)
          if(arr[i]==key)
              return i;
    }
  return -1;
}
```

Q2. pseudocode of Insertion Sort

```
void insertion( int arr[], int n)
{
    for(int i=1; i<n; i++)
    { int key = arr[i];
      int j = i-1;
      while(j>=0 && arr[j] > key)
      {       arr[j+1] = arr[j];
              j--;
      }
    arr[j+1] = key;
    }
}
```

Insertion sort is also called online sorting as, as soon as an element comes in an array it is automatically inserted at its correct position.

Q3

Algorithm	Time Complexity
Bubble sort	$O(n^2)$
Insertion Sort	$O(n^2)$
Selection Sort	$O(n^2)$
Merge Sort	$O(n \log n)$
Quick Sort	$O(n \log n)$
Heap Algo.	$O(n \log n)$

Q4

	~~Stable~~ Stable	~~Unstable~~ Inplace
Bubble	✓	✓
Selection	×	✓
Insertion	✓	✓
Merge	✓	×
Quick	×	×
Heap	×	✓

Q5 pseudocode for Binary Search

```
int start=0
int end = size-1
while (start <= end)
{
    int mid = start + (end - start)/2;
    if (key = arr [mid])
        return mid;
    else if (key < arr(mid))
        end = mid-1;
    else
        start = mid+1;
}
return -1;
```

	Linear Search	Binary Search.
Time Complexity	O(n)	O(logn)
Space Complexity	O(1)	O(1).

Q6 Reaurrence relation of Binary Search

$= T(n) = T(n/2) + 1.$

Q8 Quick Sort is best sorting algo in practical uses as it follows the locality of reference and also best case time complexity is O(nlogn).

Q9 no. of inversions: for any array no. of inversion count indicates how far (or close) the array is from being sorted. If array is already sorted then the inversion count is 0. If an array is sorted in the reverse order then the no. of inversion count is the maximum.

if $a[i] > a[j]$ and $i < j$.

7 12 31 8 10 1 20 6 4 5

(7,1), (7,6), (7,4), (7,5),
(12,8), (12,10), (12,1), (12,6), (12,4), (12,5)
(31,8), (31,10), (31,1), (31,20), (31,6),
(31,4), (31,5),
(8,1), (8,6), (8,4), (8,5)
(10,1), (10,6), (10,4), (10,5)
(20,6), (20,4), (20,5)
(6,4), (6,5)
(4,.

no. of inversions = 30.

Q10 Quick Sort Best Case: array is totally sorted.
Worst Case: array is reverse sorted.

Q11 Recurrence Relation of

Merge Sort
$2T(n/2) + \Theta(n)$

Quick Sort
$T(n) = T(k) + T(n-k-1) + \Theta(n)$
$T(n) = T(n-1) + \Theta(n)$.

Similarity : Both are divide & Conquer Algorithm.

Difference : Worst case Complexity of Merge sort remains O(nlogn) while for Quick Sort it changes to $O(n^2)$.

Q13 Optimised Bubble Sort:-

```
for (int i=0; i<n; i++)
{
    swap = false;
    for( j=0; j<n-1-i; j++)
    {   if(arr[j] < arr[j+1])
        {
            swap(arr[j], arr[j+1]);
            swapped = true;
        }
    }
}
```

Q14 External Sort : data is divided into chunks and then sorted using Merge Sort.

Internal Sort : It is type of data sorting in which whole sorting takes place in main memory of computer.

If the physical memory of 2GB is used for an array of 4GB of sorting Merge sort would be efficient to use as it is a External Sorting Algorithm.