Tutorial - 3

Ans-1 Pseudocode for linear Seach is

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

Ans-2 Pseducode 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;
    }
}
```

Ans-3 Average case complexity of Sorting algos

- 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 Sort → $O(n \log n)$

Ans-4

Stable → (appears in same order)

Bubble, insertion, merge

Inplace → (using constant space)

Bubble, Selection, insertion, heap

Ans-5 Pseducode for Binary Search

```
int BS( int arr[ ], int n, int Key) {
    int low = 0, high = n-1;
    While ( low <= high) {
        int mid = [low + (high - low)/2]
        if ( Key == arr[mid] ) {
            Return mid;
        }
        else if ( Key > arr[mid] ) {
            Start = mid + 1;
        }
        else {
            end = mid - 1;
        }
    }
    Return -1;
}
```

TC → O(logn)

SC → (O)(1)

Ans-6 Recurrence relation of Binary Search is

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

Ans-7 Assuming given array is sorted then we can find sum in linear complexity.

```
int Sum (int arr[], int target) {
    int i = 0; j = n-1;
    while (i < j) {
        if (arr[i] + arr[j] == target)
            return 1;
        else if (arr[i] + arr[j] > target)
            j--;
        else
            i++;
    }
    return -1;
}
```

Ans-8 Quick sort is the best sorting algorithm in Practical use as it follows the locality of reference and also its best case time complexity is $O(n \log n)$.

Ans-9 No of inversion tells us how far the array is from being sorted.

```
if (arr[i] > arr[j] && i < j)
```

→ 4, 7, 21, 31, 8, 10, 1, 20, 6, 45

inversion = 4 + 7 + 7 + 4 + 4 + 3 + 2

= 31

Ans-10 Quick Sort

Best Case → When array is totally unsorted.

Worst Case → When array is sorted or reverse sorted.

Ans - 11

	Merge Sort	Quick Sort
Best	$2T(n/2) + \Theta(n)$	$T(n) = T(k) + T(n-k-1) + \Theta(n)$
Worst	$2T(n/2) + \Theta(n)$	$T(n) = T(n-1) + \Theta(n)$

Similarity → Both are based on divide and conquer technique.

Differences → Worst case TC of merge sort is $O(n \log n)$ while of Quick sort is $O(n^2)$

Ans - 12

```
Void StableSelectionSort(int arr[], int n) {
    for(int i=0; i<n-1; i++) {
        int min = i;
        for(int j=i+1; j<n; j++) {
            if(arr[min] > arr[j]) {
                min = j;
            }
        }
        int key = arr[min];
        while(min > i) {
            arr[min] = arr[min-1];
            min--;
        }
        arr[i] = key;
    }
}
```

Ans-13 Optimised Bubble Sort

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

Ans-14 In such case, merge sort would be efficient as it is an External sorting algorithim. ie data is divided into chunks and then sorted using merge sort.

→ Sorted data is dumped into files.

- Internal Sorting → It is a type of sorting technique in which whole sorting takes place in main memory of computer.