# **Tutorial 10 (Data Structure)**

**Name**: Krishna Pandey

**Admn No**: U20CS110

**Roll No**: B110

### Q1

1. **Bubble sort**
   1. Given array arr of length n
   2. For i = 0 to n-1
      1. Swapped = False
      2. For j = 0 to n-i-1
         1. If arr[i] > arr[i+1]
            1. Swap(arr[i], arr[i+1])
            2. Swapped = True
      3. If Swapped = False, then break
2. **Selection sort**
   1. For i = 0 to n-1
      1. Min\_idx = i
      2. For j = i+1 to n-1
         1. If arr[j] < arr[Min\_idx], then Min\_idx = j
      3. Swap(arr[i], arr[Min\_idx])
3. **Insertion sort**
   1. For i = 0 to n-1
      1. Key = arr[i]
      2. j = i - 1
      3. While j >= 0 and arr[j] > key:
         1. Arr[j+1] = arr[j]
         2. J = j - 1
      4. Arr[j+1] = key
4. **Quick sort**
   1. quickSort(arr[], low, high)
      1. if low < high
         1. pivot\_index = partition(arr, low, high);
         2. quickSort(arr, low, pivot\_index - 1); // Before pivot\_index
         3. quickSort(arr, pivot\_index + 1, high); // After pivot\_index
   2. Partition(arr[], low, high)
      1. Pivot = arr[high]
      2. i = low - 1
      3. For j = low to high-1
      4. If arr[j] < pivot:
         1. i = i + 1
         2. Swap arr[i] and arr[j]
      5. Swap arr[i+1] and arr[high]
      6. Return i+1
5. **Merge sort**
   1. Mergesort(arr[], l, r)
      1. If r > l
         1. middle m = l+ (r-l)/2
         2. Call mergeSort(arr, l, m)
         3. Call mergeSort(arr, m+1, r)
         4. Call merge(arr, l, m, r)
   2. Merge function merges sorted arrays into a new, combined sorted array.

### Q2

List1 = [40, 60, 1, 200, 9, 83, 17]

List2 = [11, 9, 7, 5, 3, 2]

1. Bubble sort
   1. List 1

[40, 60, 1, 9, 83, 17, 200]

[40, 1, 9, 83, 17, 60, 200]

[40, 1, 9, 17, 60, 83, 200]

[1, 9, 17, 40, 60, 83, 200]

* 1. List 2

[9, 7, 5, 3, 2, 11]

[7, 5, 3, 2, 9, 11]

[5, 3, 2, 7, 9, 11]

[3, 2, 5, 7, 9, 11]

[2, 3, 5, 7, 9, 11]

1. Selection sort
   1. List 1

[1, 40, 60, 200, 9, 83, 17]

[1, 9, 40, 60, 200, 83, 17]

[1, 9, 17, 40, 60, 200, 83]

[1, 9, 17, 40, 60, 200, 83]

[1, 9, 17, 40, 60, 83, 200]

b. List 2

[2, 11, 9, 7, 5, 3]

[2, 3, 11, 9, 7, 5]

[2, 3, 5, 11, 9, 7]

[2, 3, 5, 7, 11, 9]

[2, 3, 5, 7, 9, 11]

1. Insertion sort
   1. List 1

[40, 60, 1, 200, 9, 83, 17]

[40, 60, 1, 200, 9, 83, 17]

[1, 40, 60, 200, 9, 83, 17]

[1, 40, 60, 200, 9, 83, 17]

[1, 9, 40, 60, 200, 83, 17]

[1, 9, 40, 60, 84, 200, 17]

[1, 9, 17, 40, 60, 84, 200]

* 1. List 2

[11, 9, 7, 5, 3, 2]

[9, 11, 7, 5, 3, 2]

[9, 7, 11, 5, 3, 2]

[9, 5, 7, 11, 3, 2]

[3, 9, 5, 7, 11, 2]

[2, 3, 9, 5, 7, 11]

1. Quick sort
   1. List 1

[1, 9, 17, 40, 60, 200, 83]

[1, 9, 17, 40, 60, 200, 83]

[1, 9, 17, 40, 60, 83, 200]

* 1. List 2

[2, 11, 9, 7, 5, 3]

[2, 3, 11, 9, 7, 5]

[2, 3, 5, 11, 9, 7]

[2, 3, 5, 7, 11, 9]

[2, 3, 5, 7, 9, 11]

1. Merge sort
   1. List 1

[40, 60, 1, 200] [9, 83, 17]

[40, 60] [1, 200] [9] [83, 17]

[40] [60] [1] [200] [9] [83] [17]

[40, 60] [1, 200] [9, 83] [17]

[1, 40, 60, 200] [9, 17, 83]

[1, 9, 17, 40, 60, 83, 200]

* 1. List 2

[11, 9, 7] [5, 3, 2]

[11, 9] [7] [5, 3] [2]

[11] [9] [7] [5] [3] [2]

[9, 11] [5, 7] [2, 3]

[5, 7, 9, 11] [2, 3]

[2, 3, 5, 7, 9, 11]