1. List the members of your group below:

2. A *subsequence* of sequence S is any sequence that can be obtained from S by deleting zero or more of its elements. For example, $(1, 4, 9, 2)$ is a subsequence of $S_1 = (3, 1, 4, 5, 9, 2)$, but $(1, 9, 4)$ is not. A subsequence S' of S is called a *k-subsequence* if each pair of adjacent elements in S' has $k-1$ intermediate elements in S. For example, $(1, 5, 2)$ is a 2-subsequence of S_1, and $(3, 5)$ is a 3-subsequence of S_1, but $(1, 5, 9)$ is not a k-subsequence of S_1 for any value of k (although it is a subsequence of S_1). A k-subsequence with n elements is called *maximal* if there is no k-subsequence with $n + 1$ elements. List all *maximal 5-subsequences* and *maximal 7-subsequences* of the following sequence:

```
50  40  60  70  65  75  62  63  41  42  51  52  53  54
```

3. We say a sequence is *k-sorted* if all of its k-subsequences are sorted. For each of the following, provide an example of a sequence with the indicated properties, or explain why no such sequence exists.

   (a) 7-sorted but not 5-sorted.

   (b) 5-sorted but not 7-sorted.

   (c) 6-sorted but not 3-sorted.

   (d) 3-sorted but not 6-sorted.

4. Sort the following array in ascending order using *shellsort with increment sequence* $(1, 5, 7)$.[1] Depict the state of the array after each k-sort, for $k = 1, 5, 7$ and highlight the moved elements at each stage.

[1] Mark Allen Weiss, *Data Structures and Problem Solving Using Java*, 3rd edition (Addison-Wesley, 2006), §8.4.

5. Consider the process of sorting the array of Question 4 in ascending order using *merge-sort*.[2] Depict the recursive invocations of the `mergeSort` method using a tree in which nodes represent `mergeSort` invocations and are labeled with the indices of the sub-arrays sorted by them. Further, the parent of a node n is the node p corresponding to the `mergeSort` invocation (if any) from which n's invocation is called.

[2] *Idem*, §8.5.

6. Augment, or redraw, the tree of Question 5 by adding to each node's label the state of the sub-array corresponding to that node's invocation (1) immediately before the invocation and (2) immediately after the invocation.