Makeup Assignment:

Problem 1 (30 points) stable merge sort

Consider the following unsorted list of integers containing duplicates where the relative position of each duplicated integer in the unsorted list is noted as a subscript. For example, 1_1 is at a smaller index than 1_2. The subscript is ignored when comparing two values since it would not actually be present in the underlying Python representation of this list.

8_1	4_1	1_1	4_2	8_2	1_2

A sorting algorithm is **stable** if the relative position of duplicate items is preserved. Stable sorting can be a useful property in many instances. The list below represents a stable sort of he list above.

1_1	1_2	4_1	4_2	8_1	8_2

Another example would be sorting a list by first name and then by last using a stable sorting algorithm. This would order everybody with the same last name by their first name.

You will modify the merge sort procedure from lecture 20 to demonstrate that merge sort can be stable. Hint: consider how ties should be broken during the merge phase.

To demonstrate that the modified merge sort procedure modify it so it sorts a list of tuples by their second position. The tuples contain the first and last names of some random individuals.

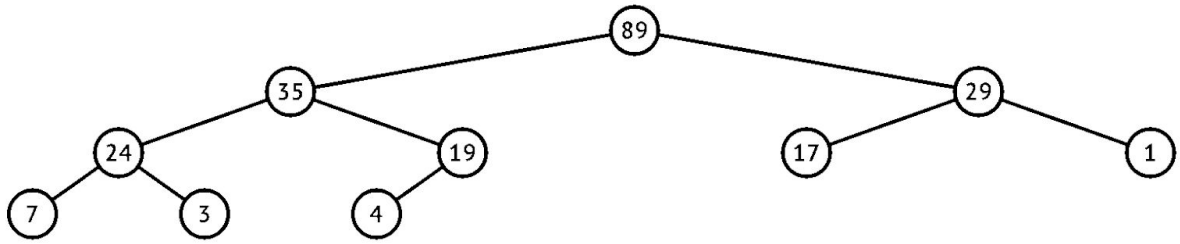
Problem 2 (15 points) unstable selection sort

You will show that selection sort is unstable by sorting the same list of tuples in part 1 by their second position. Note that the first names of people with the same last name are no longer in sorted order.

Problem 3: (30 points) Heap Representations

Part 1:

Convert the following maxheap from a tree representation to a Python list representation.



Part 2:

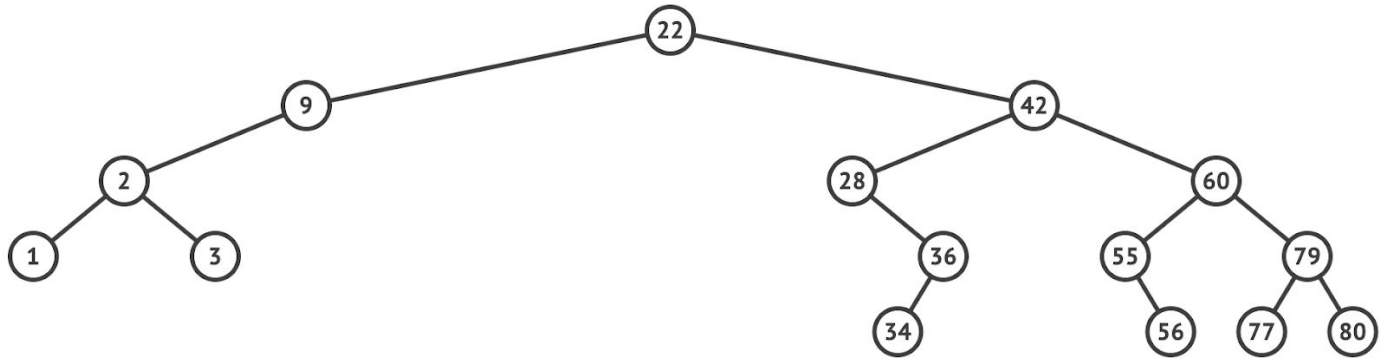
Show the new state of the maxheap in a Python list representation after inserting the value 63 into the maxheap and restoring the heap order property.

Part 3:

Show the state of the maxheap in a Python list representation after removing the value 89 from the *original* maxheap and restoring the heap order property.

Problem 4 (25 points) Tree traversal

Consider the following tree containing integer values at each node. Traverse the tree in the order specified. When visiting each node, the operation you will perform is to append the integer to the end of a Python list. In your homework file specify the list of integers that results from each traversal approach.



Part a. inorder
Part b. preorder
Part c. postorder
Part d. levelorder (see lecture 23)