IMPORTANT FIRST STEPS:

1. **Close your laptops and put them away.**
2. **Form a group of 2-3 students.**
3. **Clearly put your names and IDs on 1 copy of this worksheet.**
4. **Be sure to turn this exercise in at the end of class.**

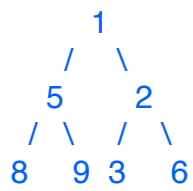
Heaps

In your own words, define a *nearly complete* binary tree:

Refer to lecture notes

Translate the following list of keys into a nearly complete tree:

1 5 2 8 9 3 6



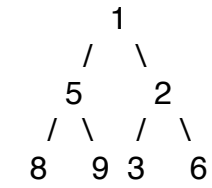
What happens if you try to translate the above list of keys into a tree that is not nearly complete?

The tree is ambiguous (there is no one correct tree)

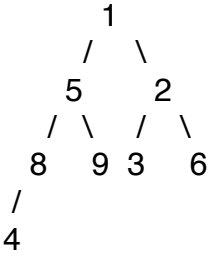
Is the above tree a heap?

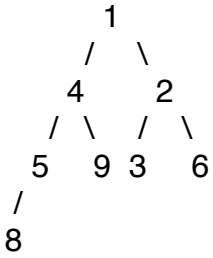
Yes (a minimum one)

Redraw the tree from above here:

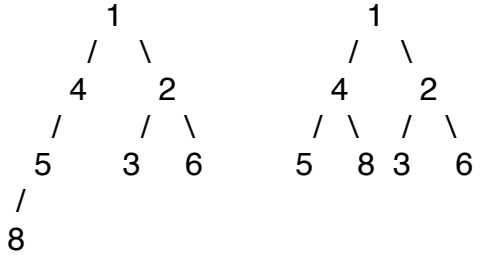


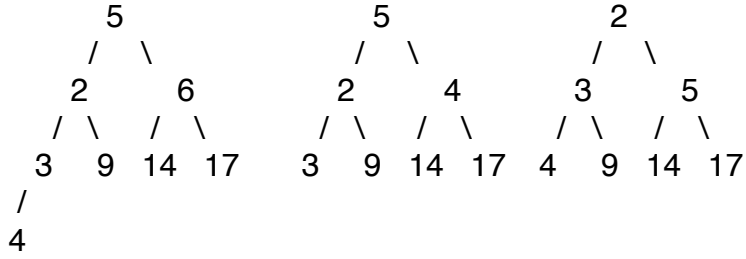
Insert key 4 (maintain the heap).





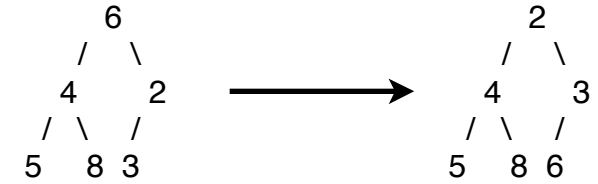
Delete key 9 (maintain the heap). **Note that here we replace 9 with 8, then we first reheap 8 up (it goes nowhere as it's not out of order) and then we reheap it down. You only need to reheapup first if you're not deleting the top most node (i.e root).**



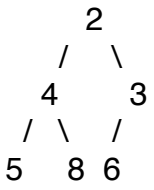


Different example, here we delete 6. We have to move 4 to the empty spot, reheap it up to the root, then apply reheap down

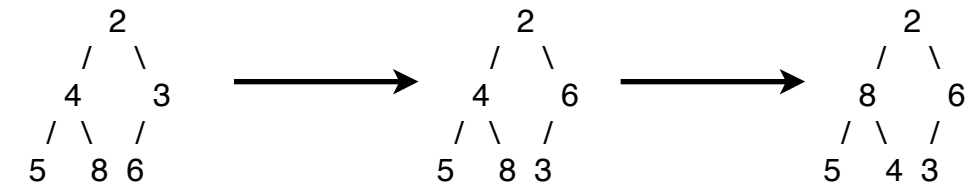
Delete key 1 (maintain the heap).

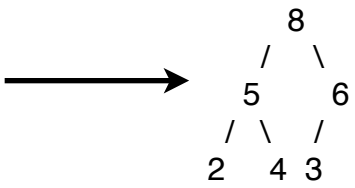


Redraw your tree from the preceding page here:

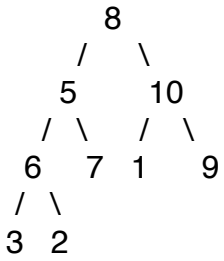


Apply the *heapify* algorithm to generate a **maximum** heap (simply switch the conditions to match the definition of a maximum heap). Show each step-- if there is no change, write “no change”:

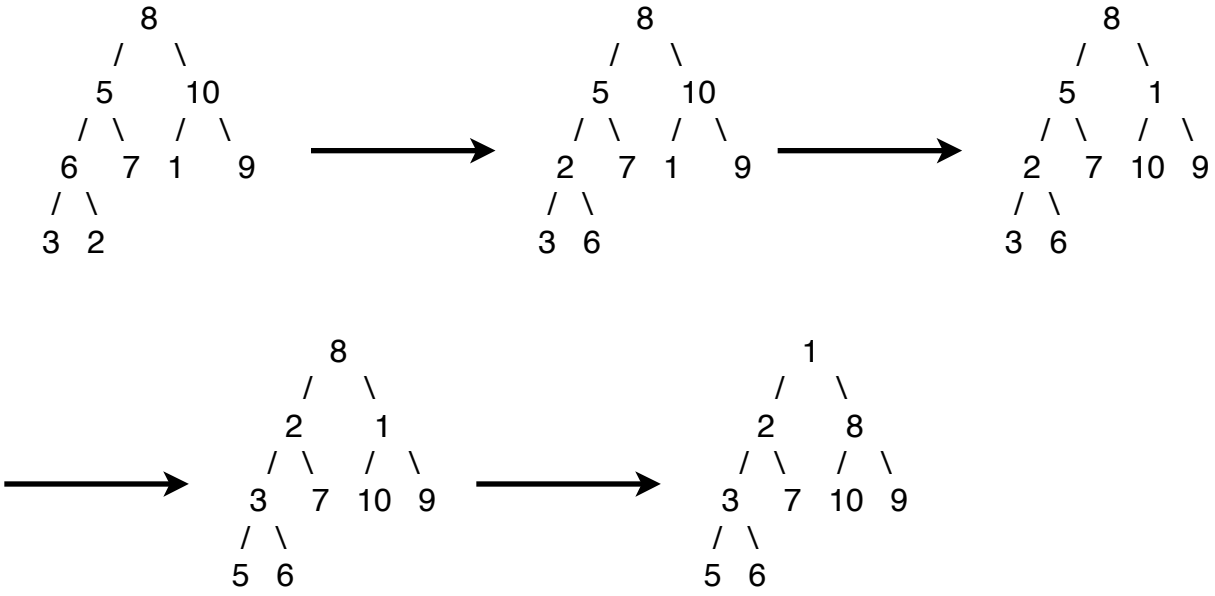




Draw a nearly complete tree from the following keys: 8 5 10 6 7 1 9 3 2

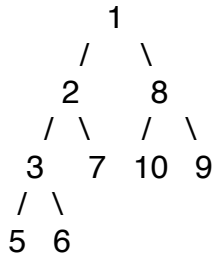


Using this tree, apply the *heapify* algorithm to generate a **minimum** heap (this is part one of *heapsort*). Show the tree after each step:

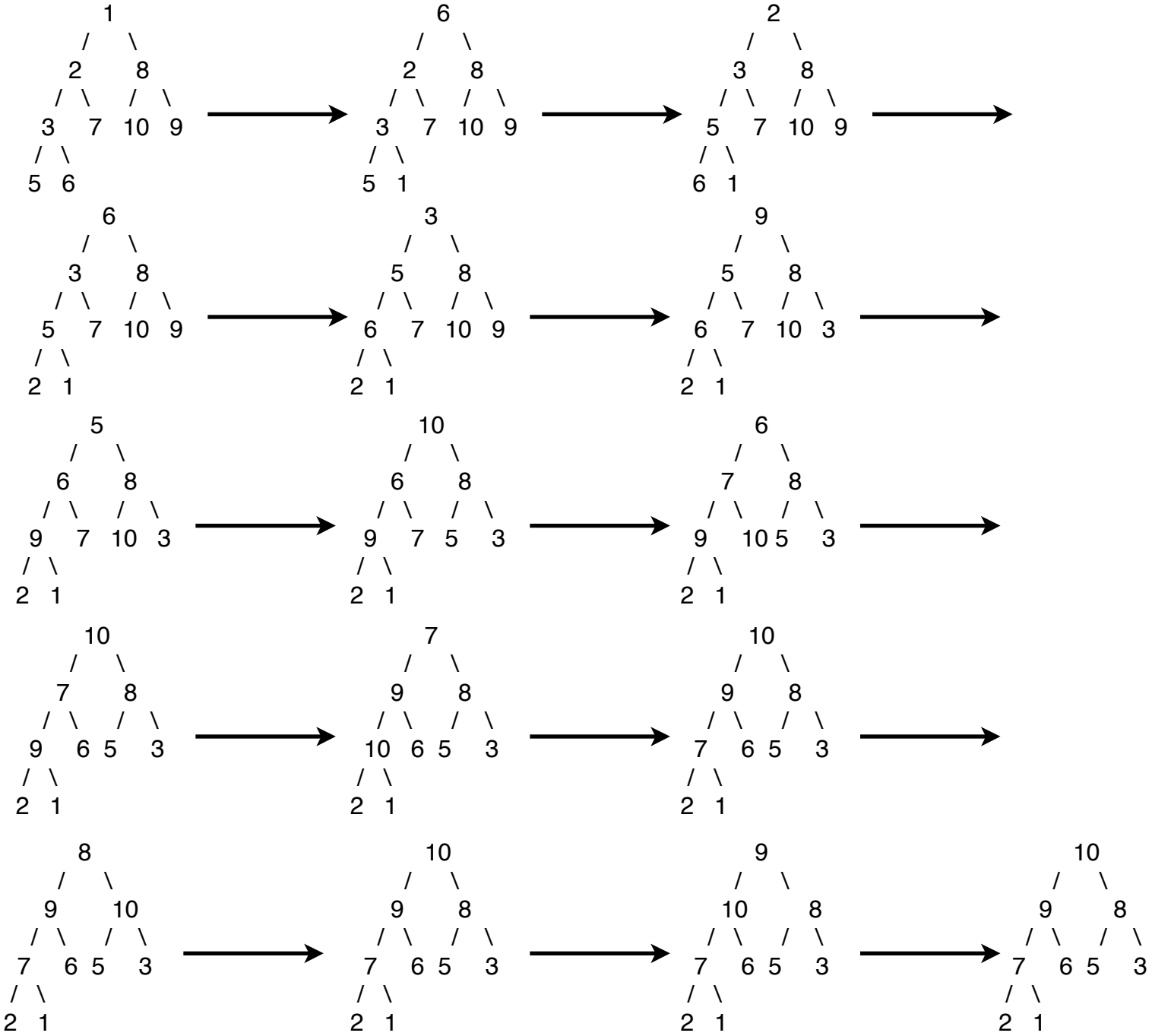


SAMPLE SOLUTIONS

Redraw your tree from the preceding page here:



Using this tree, apply the *sort* algorithm (this is part two of applying *heapsort*). Show the tree after each step:



Rewrite both the ReheapUp and ReheapDown algorithms (from your lecture notes) for **maximum** heaps:

Algorithm:

```
ReheapUp( root, bottom )  // subscripts are passed
   if ( bottom < root )
        set parent to subscript of parent of bottom element
        if ( data[parent] < data[bottom] )
             swap( data[parent], data[bottom] )
             ReheapUp( root, parent )
```

Algorithm:

```
ReheapDown( root, bottom )            // subscripts are passed
    if ( root node is not a leaf )
       set maxChild to subscript of child's smallest data value
       if ( data[root] < data[maxChild] )
          swap( data[root], data[maxChild] )
          ReheapDown( maxChild, bottom )
```