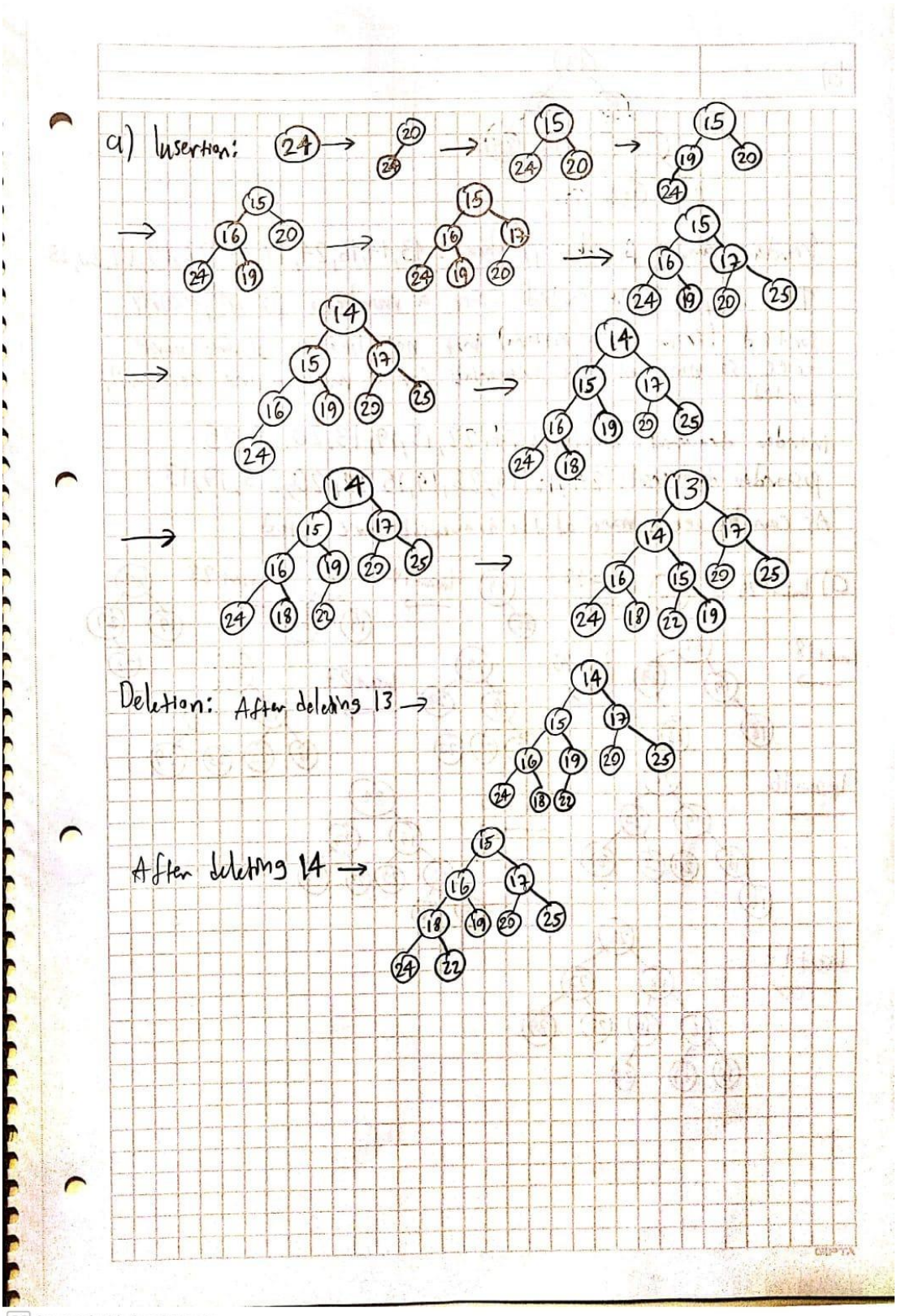
Efe Beydoğan

21901548

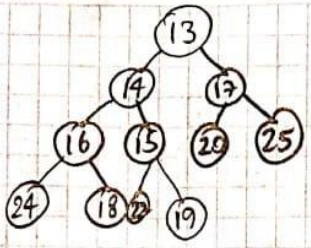
CS202 Section: 1

HW3 Report

Question 1:



b)



Preorder traversal for this minheap: 13,14,16,24,18,15,22,19,17,20,25

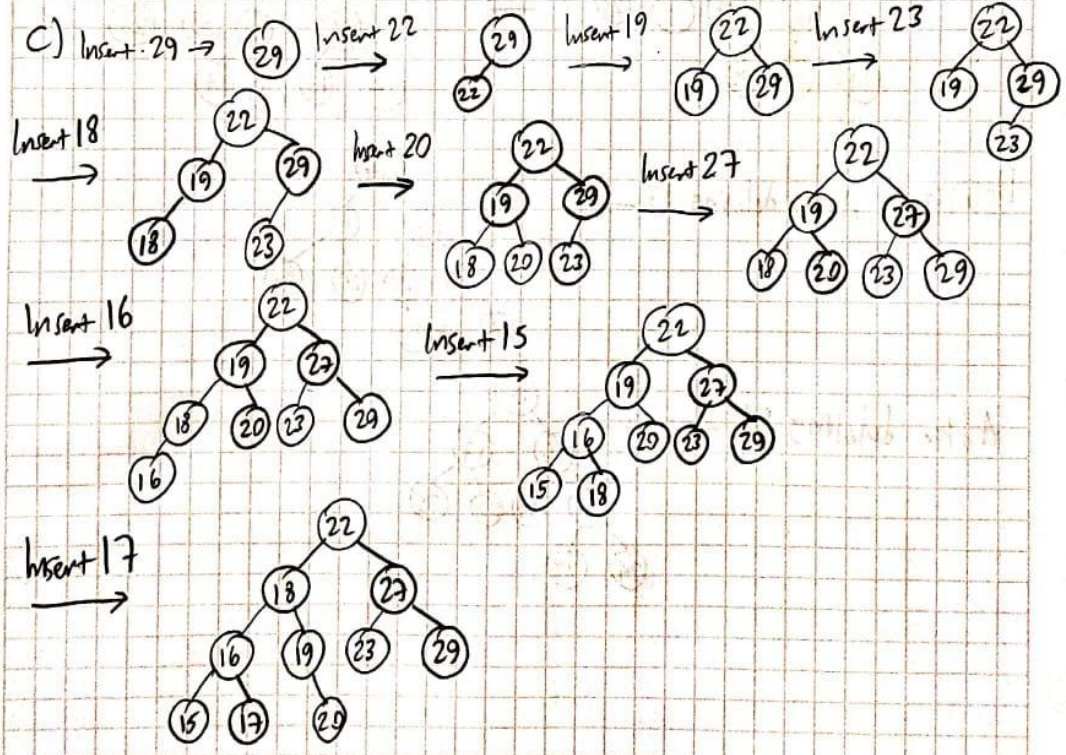
The output isn't sorted since a minheap is only loosely sorted (from top to bottom) but not like a binary search tree, so none of the traversals for a minheap are necessarily sorted.

inorder traversal: 24,16,18,14,22,15,19,13,20,17,25

postorder traversal: 24,18,16,22,19,15,14,20,25,17,13

As can be seen, none of the traversals are sorted

c)



Question 3:

If a very large library with many potential printers and many more print requests was being considered, then starting with 1 printer and increasing the number of printers until the constraint was satisfied wouldn't be a very efficient way to solve this problem. Instead, in order to increase efficiency, we could first set the number of printers to half the requests. After calculating the average waiting time for this situation, depending on the outcome, we could again try the middle point. For example, if for this situation the average time turns out to be more than the desired one, we could try again by setting the number of printers to the half of (total requests + half of total requests). Also, when in one of these situations the desired time constraint is satisfied, we could then set the number of printers to the middle of the lower half, meaning if we first tried the number (total requests / 2) and this satisfied the constraint, then we could try (0 + total requests / 2) / 2, to see if a smaller number of printers could also satisfy the constraint, and thus we can find the optimum number of printers this way and without having to try every number from 1 to K. This could tremendously increase the efficiency and the time complexity would be O(logn) rather than O(N) where N is the total number of printers.