Name: Tianlin Yang
ID: 40010303

I'm using stylus drawing to describe tree clearly

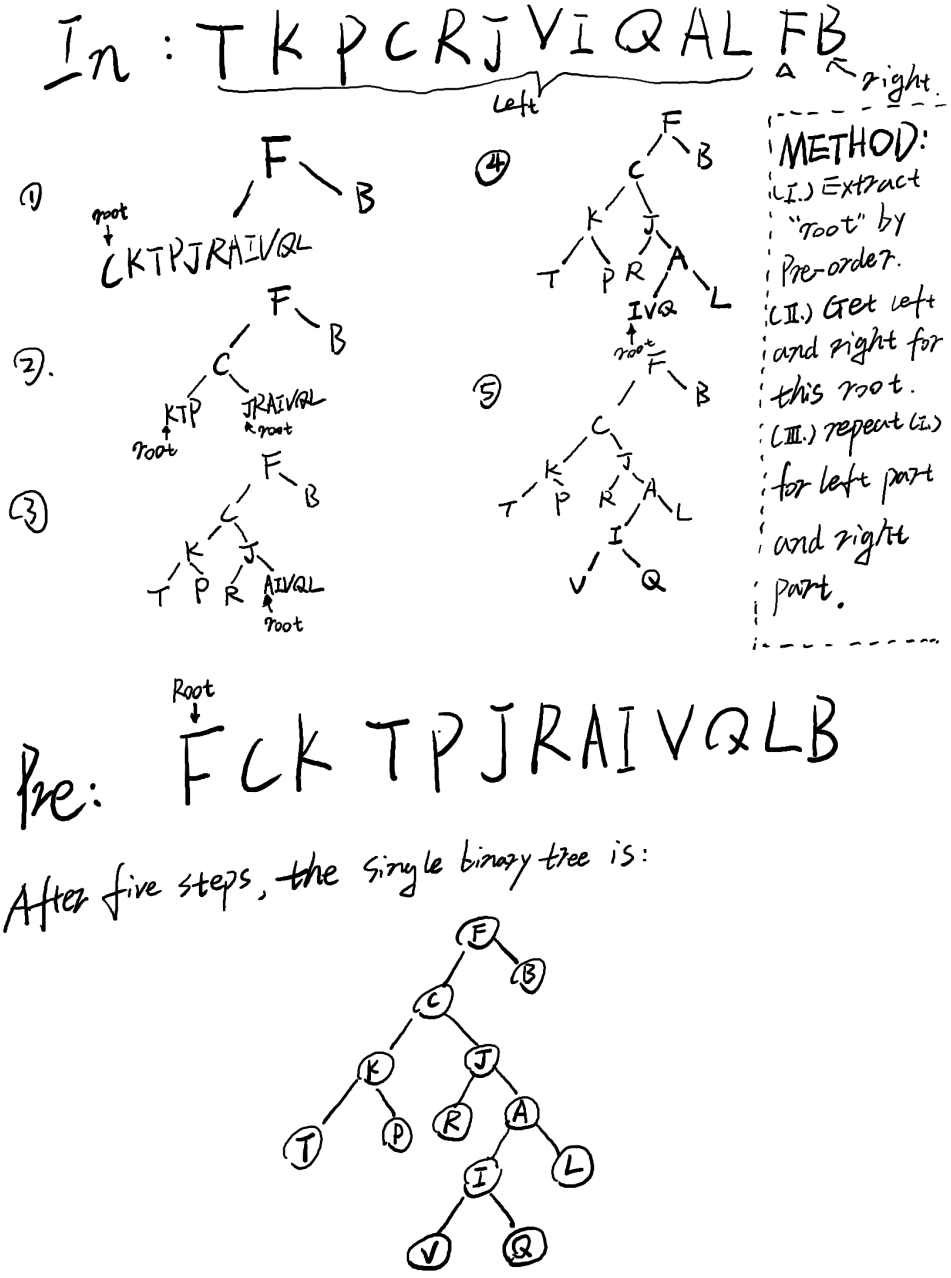
Comp 352
Data Structure and Algorithms
Assignment 3 – Fall 2018
Due date: Monday November 12th, 2018 by midnight

Written Questions (50 marks):

Question 1

Draw a single binary tree that gave the following traversals:

Inorder: T K P C R J V I Q A L F B
Preorder: F C K T P J R A I V Q L B



Question 2

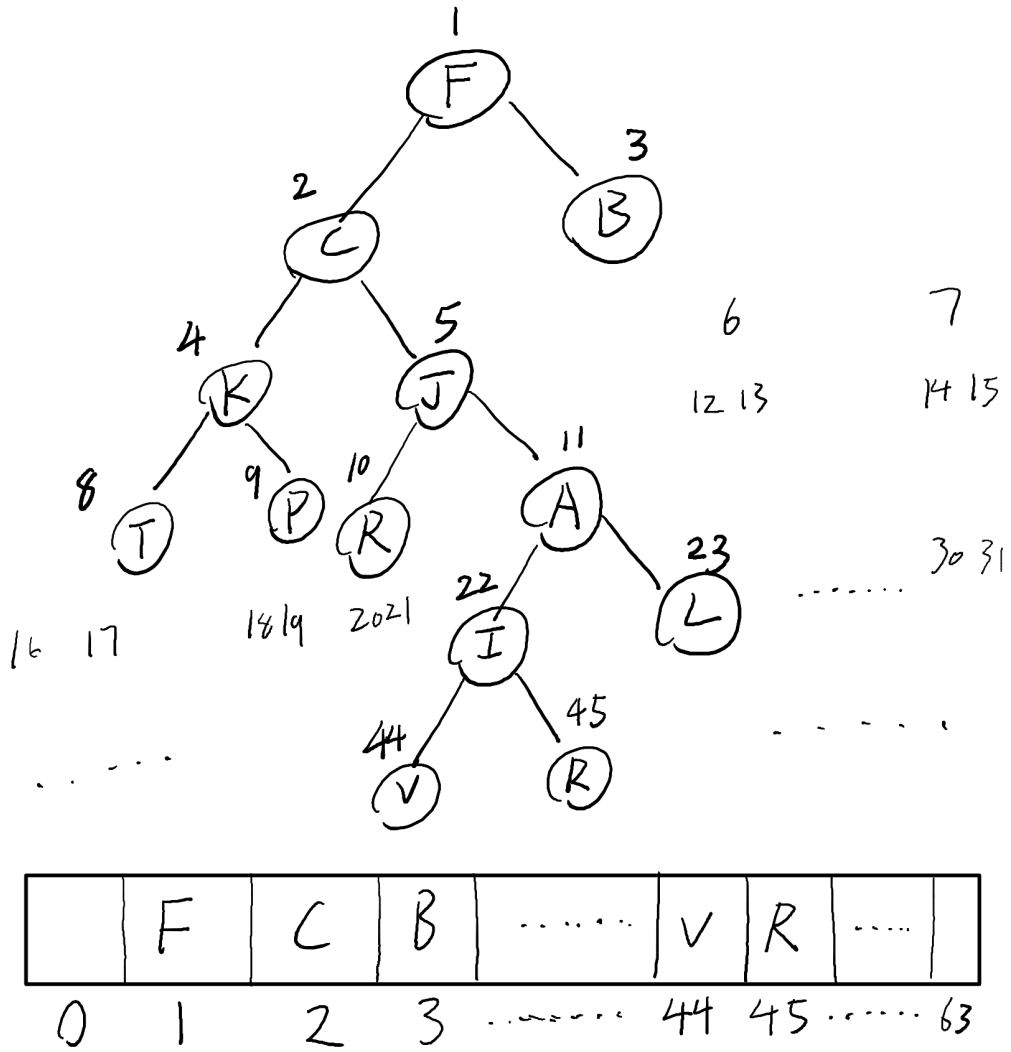
Assume that the binary tree from the above Question_1 is stored in an array-list as a complete binary tree as discussed in class. Specify the contents of such an array-list for this tree.

According to the PDF note from course COMP352, the representation equations are:

Index (root) = 1

Index (left child) = Index (parent (node))*2

Index (right child) = Index (parent (node))*2+1



Question 3

A sequence is a list that supports the index-based operations of a List and the position-based operations of a Positional List (aka NodeList). The following questions concern sequences.
It is possible to implement the Sequence ADT using either a doubly linked lists, or an array of positions, as the underlying data structure. For each of the following cases, indicated which design is preferable and explain your answer:

i. A massive number of insertion operations are needed some of which are based on indices and some are based on insertion of positions at the start of the Sequence (addFirst).
ii. A massive number of addition before and after a position as well as adding a position at the end of the Sequence.
iii. A massive number of removal operations at positions and a massive number of setting the values at indices.

Answer:

i. Using doubly linked lists is preferable, because you don't know how many items will be in the list. With arrays, you may need to re-declare and copy memory if the array grows too big, Arrays have O(1) random access, but are really expensive to add stuff onto or remove stuff from. Linked lists are really cheap to add or remove items anywhere and to iterate, even though random access is O(n), but based on insertion of positions at the start of the Sequence, no need for random access.

ii. Using doubly linked lists is preferable. First, because addition before and after a position in an array, although it does not take extra time to locate, it takes O(n) for insertion and deletion. On the contrary, doubly linked lists takes time to address, but it is easy for insertion and deletion, also we don't need to consider capacity. Second, insert at the end generally O(1) worst case O(n) in array list, but insert in end position, the time complexity is O(1).

iii. Using an array of positions lists is preferable. Linked lists are really cheap to remove items anywhere and to iterate, but random access is O(n). Arrays have O(1) random access, but are really expensive to add stuff onto or remove stuff from. Also, when we consider the setting the values at indices, Linked lists need traveling through all the list to find this location, however, arrays does not need to, It costs O(1) to find this location and change its value.

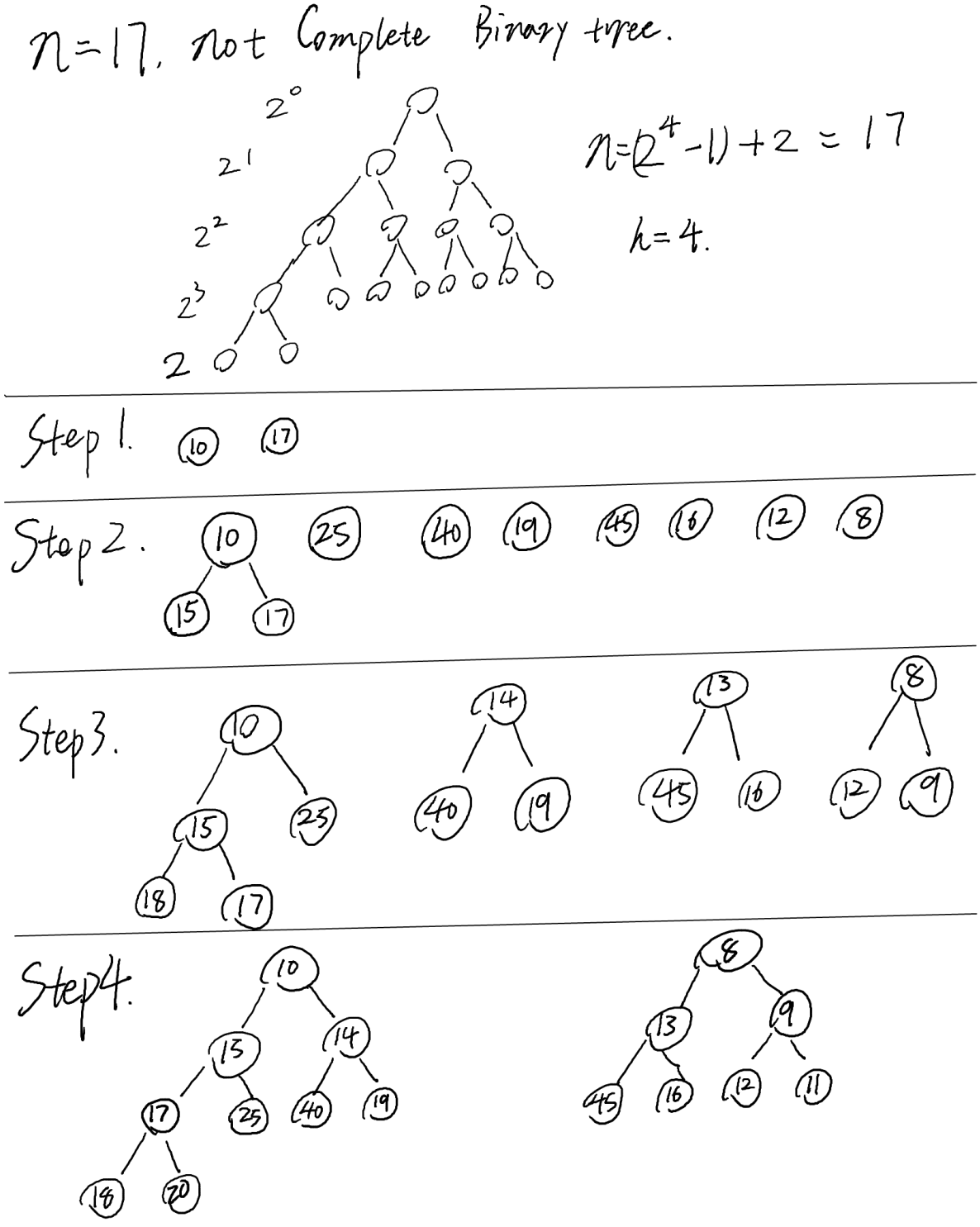
Question 4

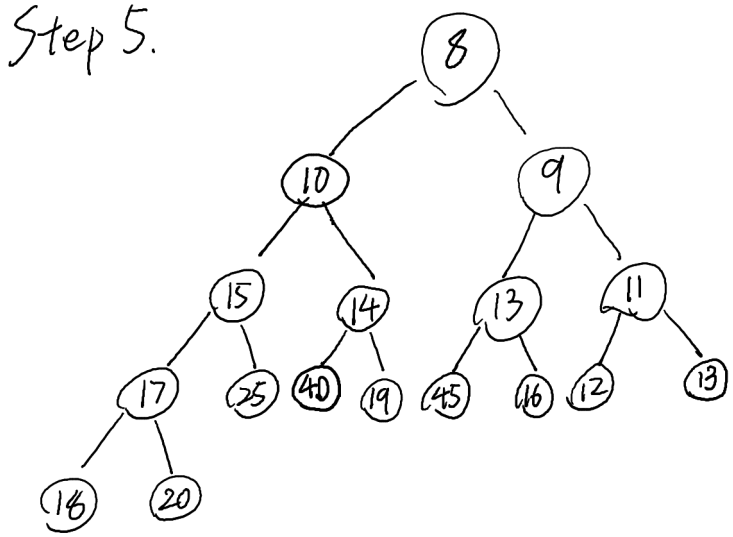
Draw the min-heap that results from the **bottom-up heap construction algorithm** on the following list of values:

10, 17, 15, 25, 40, 19, 45, 16, 12, 8, 18, 14, 13, 9, 20, 11, 13

Starting from the bottom layer, use the values **from left to right** as specified above. Show immediate steps and the final tree representing the min-heap. Afterwards perform the operation **removeMin 6 times** and show the resulting min-heap after each step.

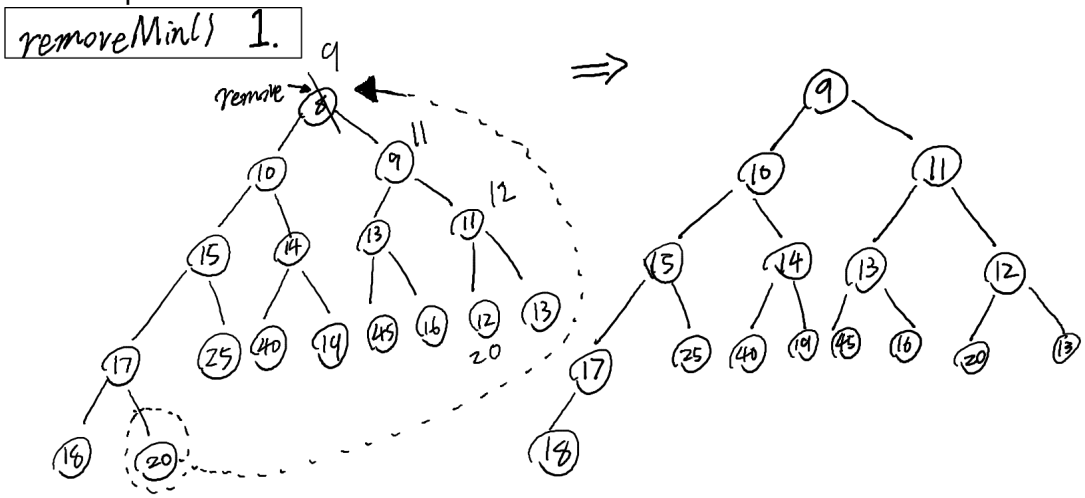
A)

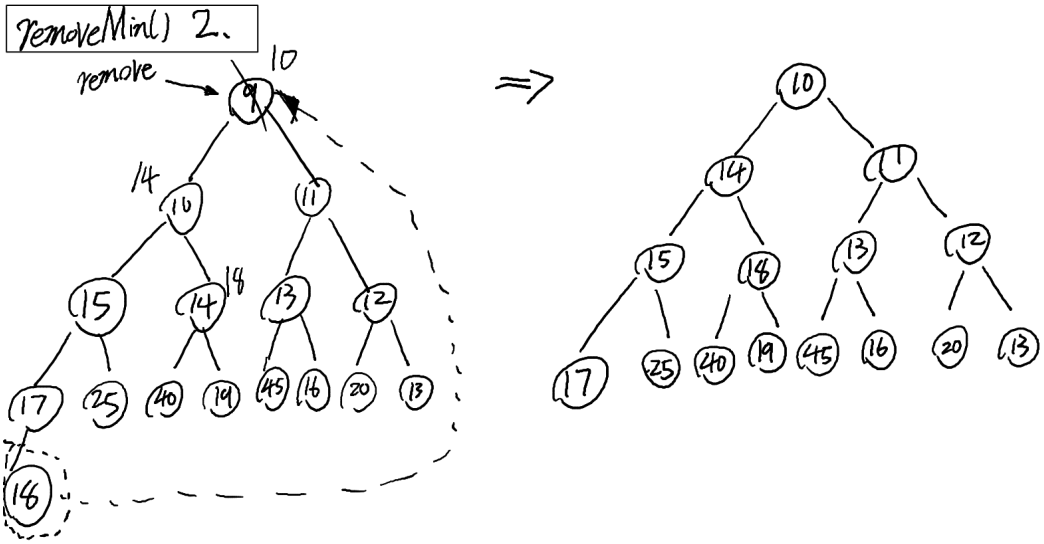




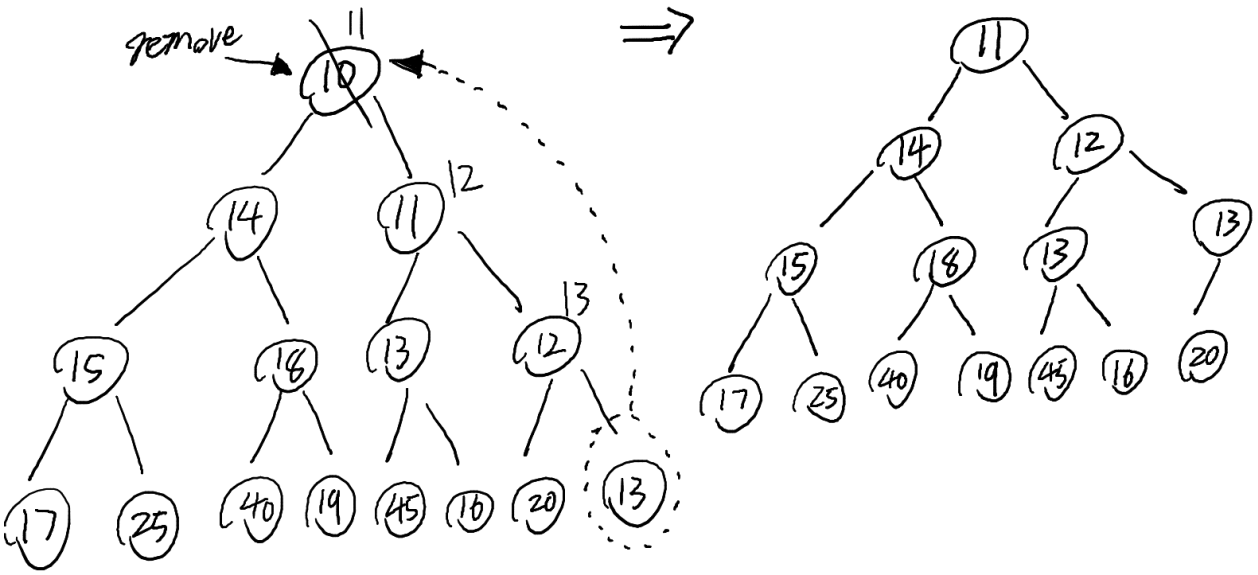
Therefore, the final tree show above needs five steps to complete from bottom to top layer.

B）The operation removeMin 6 times:

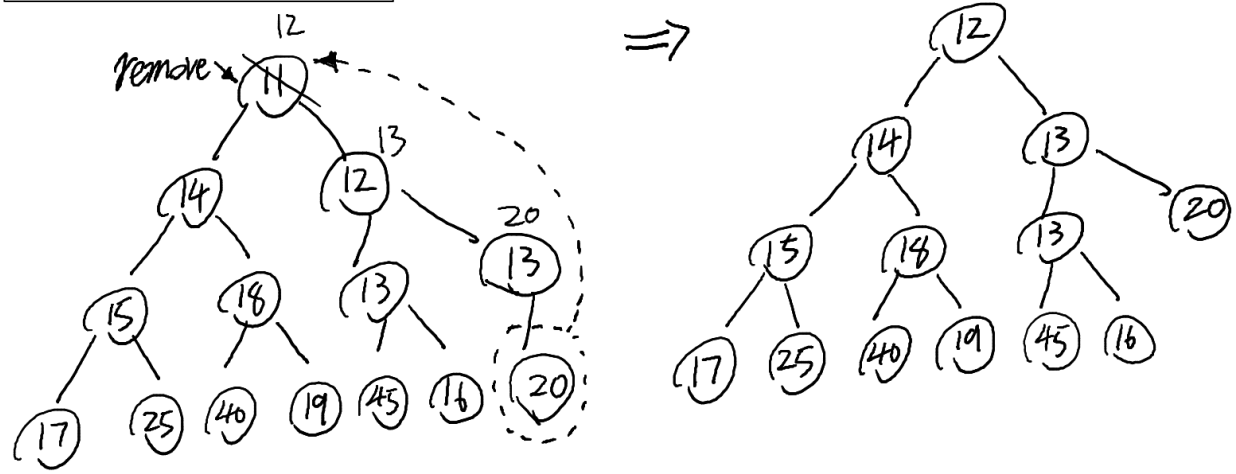




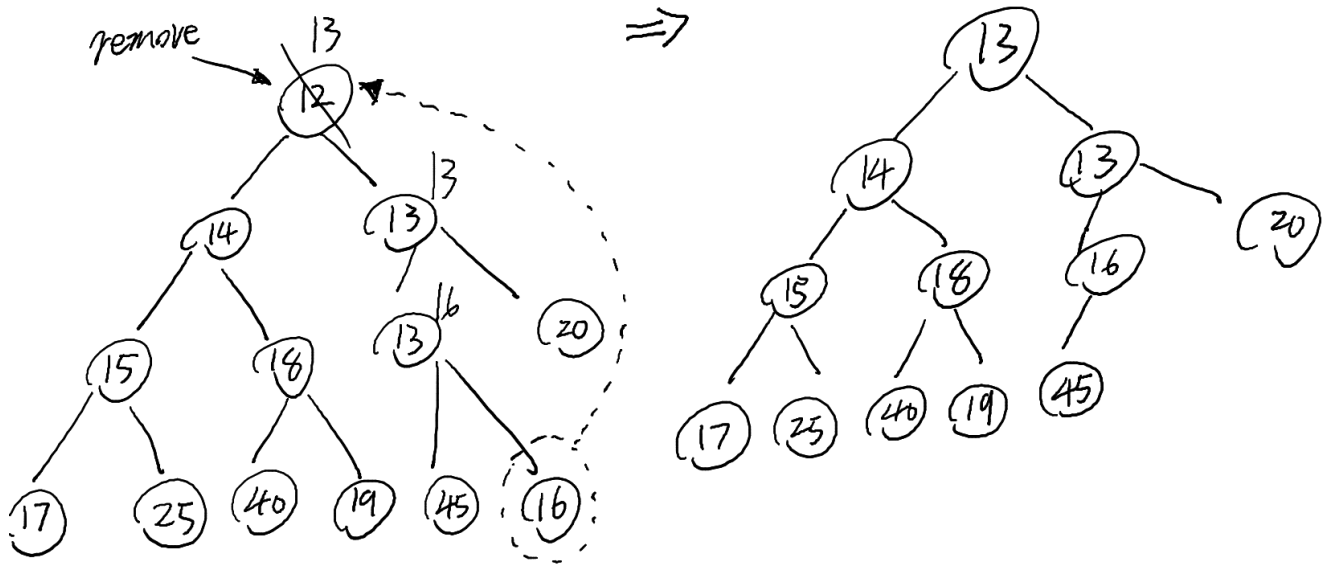
removeMin 3.



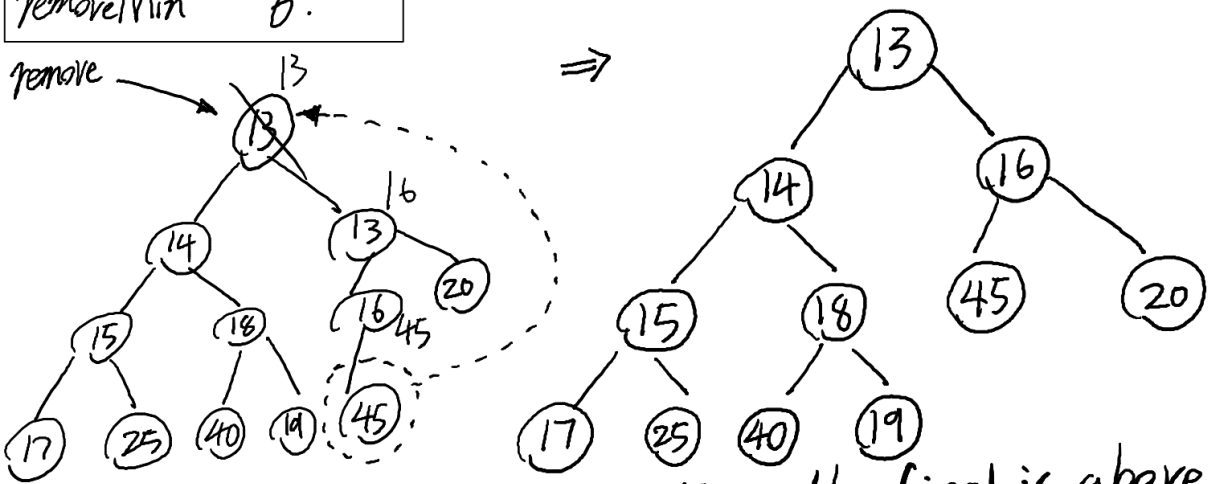
removeMin 4.



removeMin 5.



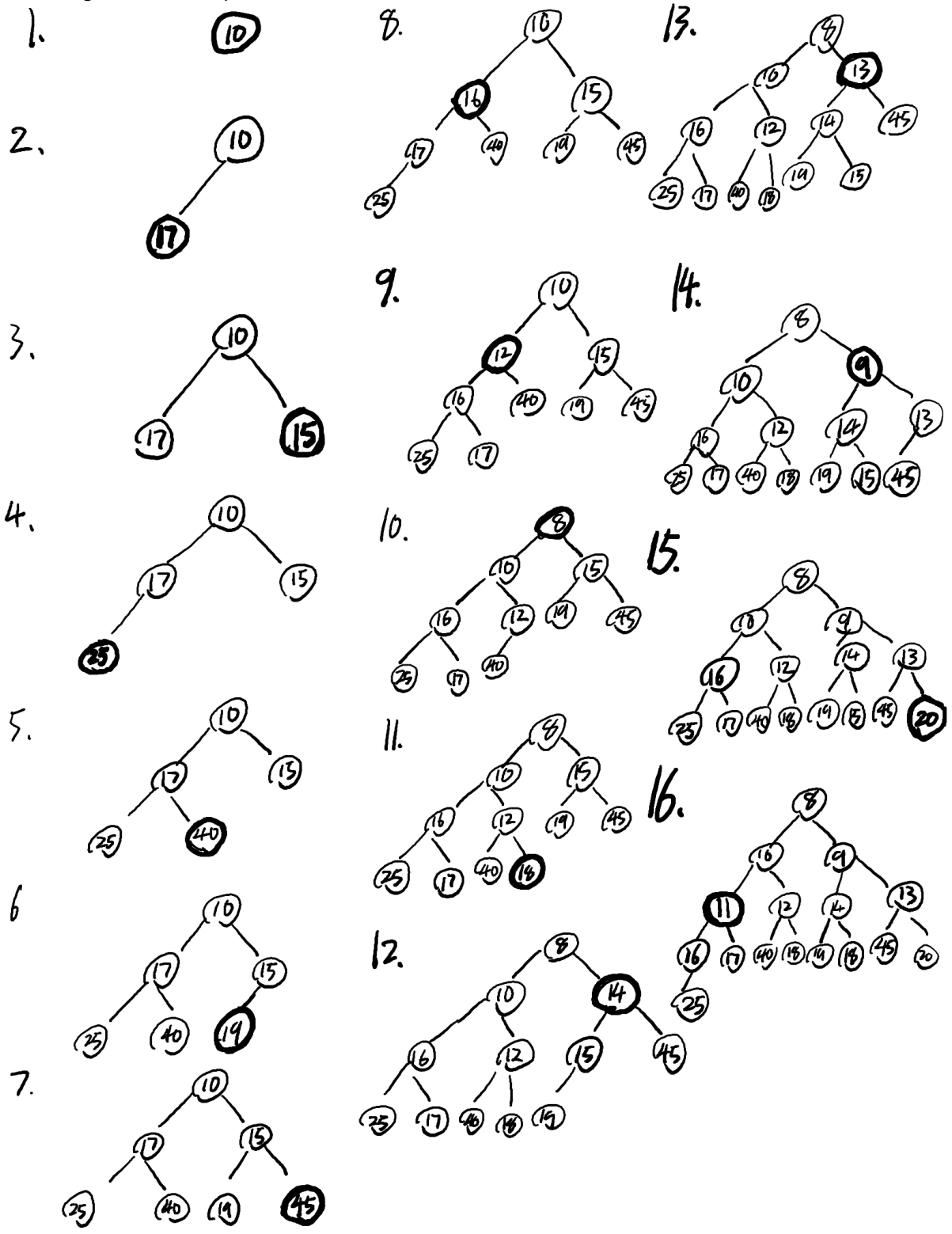
removeMin 6.



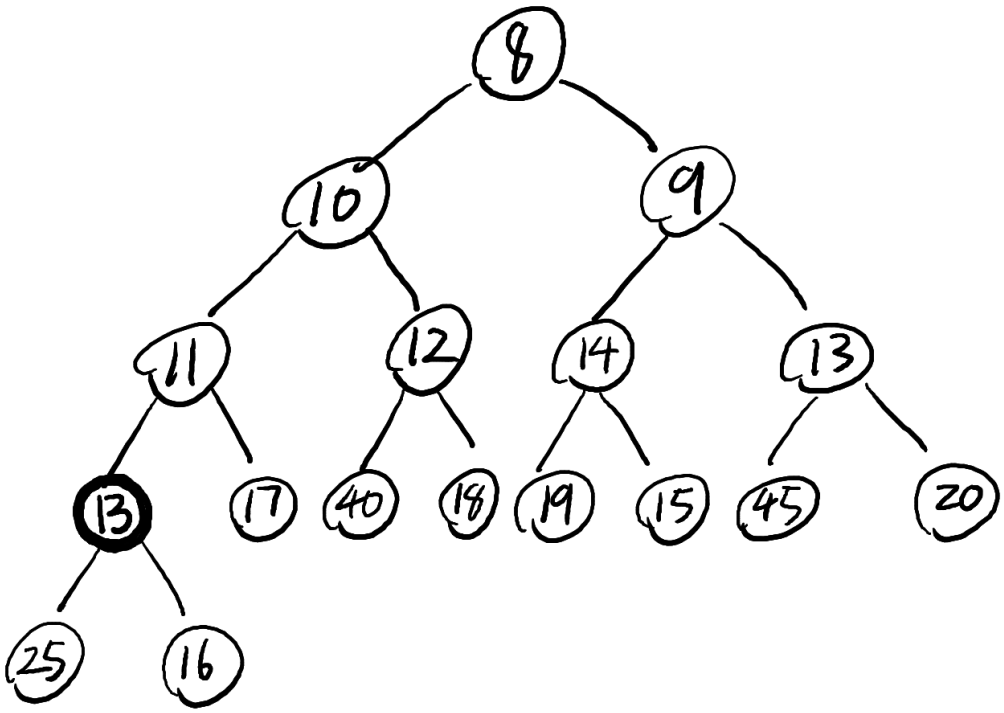
After 6 times removeMin, the final is above.

Question 5

Create again a min-heap using the list of values from the above Question_4 but this time you have to insert these values step by step using the order from left to right (i.e. insert 10, then insert 17, then 15, etc.) as shown in the above question. Show the tree after each step and the final tree representing the min-heap.



17.



Note: You must submit the answers to all the questions above. However, only one or more questions, possibly chosen at random, will be corrected and will be evaluated to the full 50 marks