CS340 – Advanced Data Structures and Algorithm Design – Fall 2020
Assignment 3 – September 25, 2020

Dr. Sandra Zilles, Department of Computer Science, University of Regina

answer key

Problem 1 (3+3+1 marks). see hand-written sheets

Problem 2 (2+3 marks).

(a) Proof by contradiction: Assume the max. item k is not in a leaf, but in an inner node n. Then n has a child whose key is smaller than k. This violates the heap-order property. [partial marks if some reasoning makes sense.]

(b) Proof by induction on N.

induction base: $N = 1$. The heap consists only of one node, which is a leaf. Since $\lceil \frac{N}{2} \rceil = 1$, there are exactly $\lceil \frac{N}{2} \rceil$ many leaves. [1 mark]

inductive hyposthesis: Suppose any binary heap with $N \geq 1$ nodes has exactly $\lceil \frac{N}{2} \rceil$ leaves. [0.5 marks]

inductive step: $N \rightsquigarrow N+1$. Consider a binary heap B with $N+1$ nodes.

If $N+1$ is even then removing a node does not change the number of leaves. By inductive hypothesis, the heap B has $\lceil \frac{N}{2} \rceil$ leaves. Since N is odd, this number equals $\lceil \frac{N+1}{2} \rceil$.

If $N+1$ is odd then removing a node decreases the number of leaves by one. By inductive hypothesis, the heap B has $\lceil \frac{N}{2} \rceil + 1$ leaves. Since N is even, this number equals $\lceil \frac{N+1}{2} \rceil$.

[1.5 marks; give partial marks if students forget to do the case distinction]

Problem 3 (4+2+2 marks).

(a) [give partial marks for partially correct code: 1 for buildHeap, 1 for percolateDown, 2 for rest]

(b) It sorts the given array in decreasing order. [1 mark]

To see this, note that after the heap is built, the smallest entry sits in the root (array position 0). This smallest element is then swapped to the end of the array and never touched again. After the swap, the heap order property is re-established by percolating down. Then again the smallest of the remaining entries is in position 0 of the array and the process is repeated. [1 mark]

(c) The worst-case running time is $\Theta(N \log(N))$. [1 mark; give this mark also if the students just write $O(N \log(N))$.]

Building the heap has a worst case running time of $\Theta(N)$. This is the worst case cost of step 1.

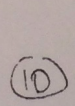
Step 2 executes a loop N times: for each value of j starting at $j = N-1$ and decrementing down to $j = 1$, the inner part of the loop percolates down in an array of j elements, at a cost of $\Theta(\log(j))$. This results in a cost of

$$\sum_{j=1}^{N-1} \log(j) = \log((N-1)!) = O(N \log(N)) \,.$$

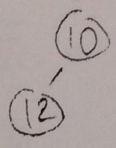
[1 mark for explanation of $\Theta(N)$ or of $O(N)$.]

a) 10, 12, 1, 14, 6, 8, 8, 15, 7, 9, 7, 4, 1, 13, 2

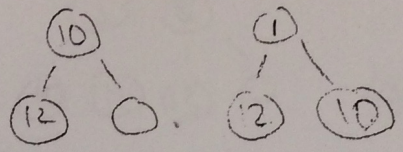
10:



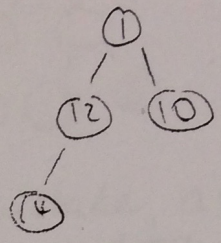
12:



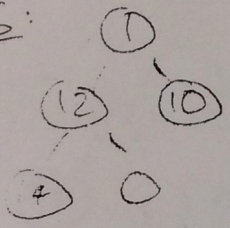
1:

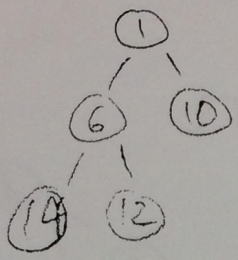


14

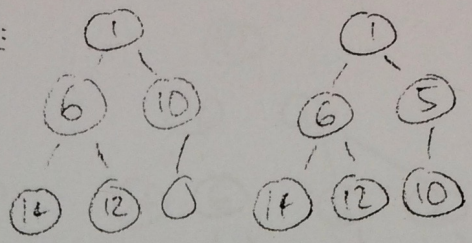


6:

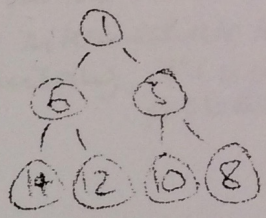




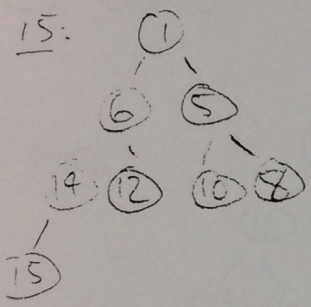
5:



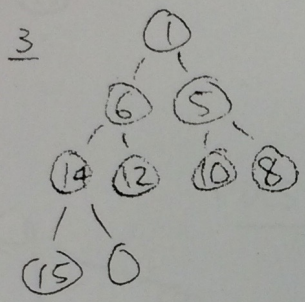
8:

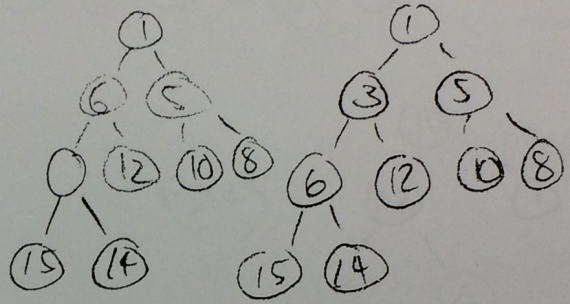


15:

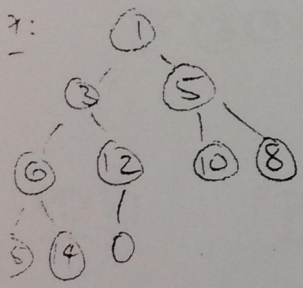


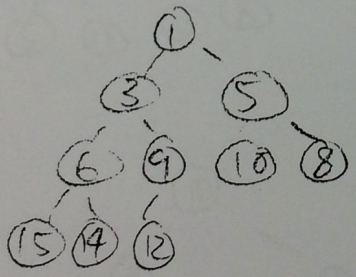
3



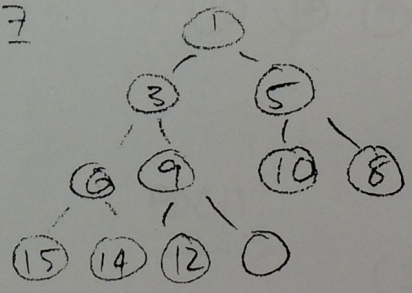


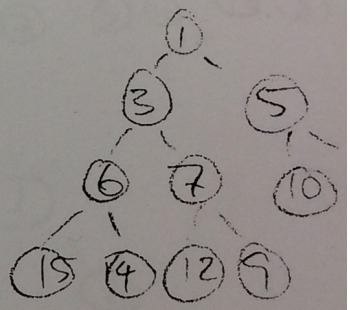
7:

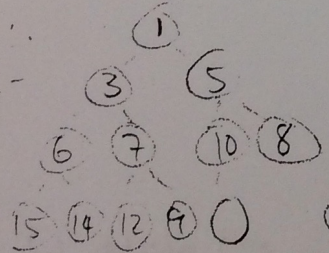


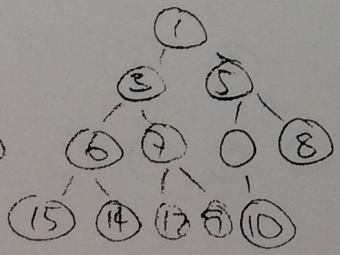


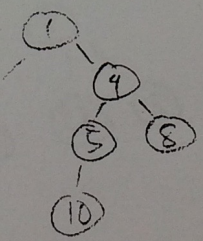
7



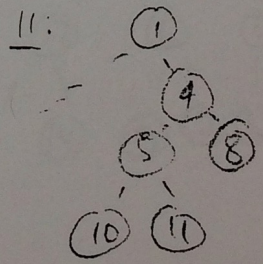




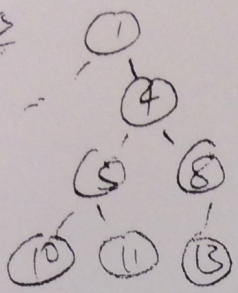




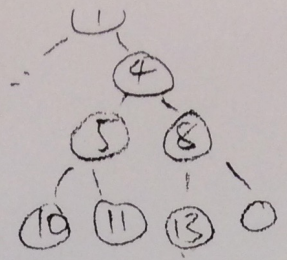
11:

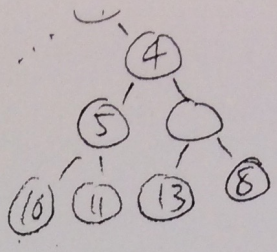


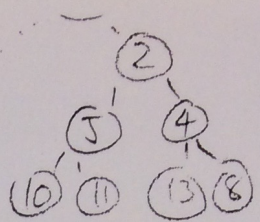
13



2.

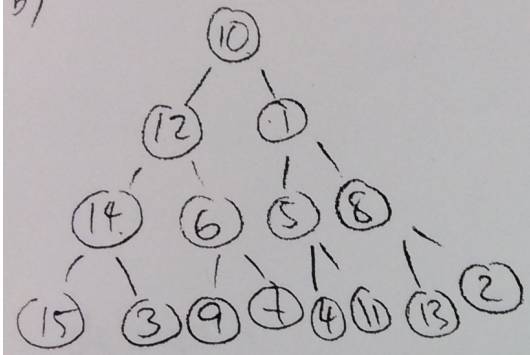




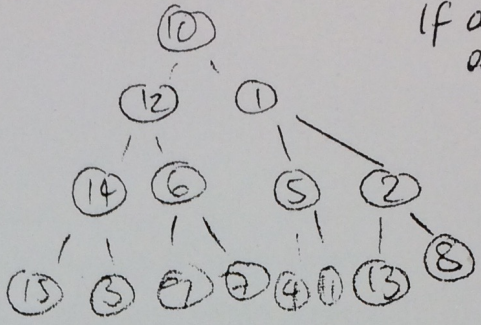


subtract 1 mark for first mistake, 0.5 marks for every further mistake.
If a few intermediate trees are missing, it's no problem

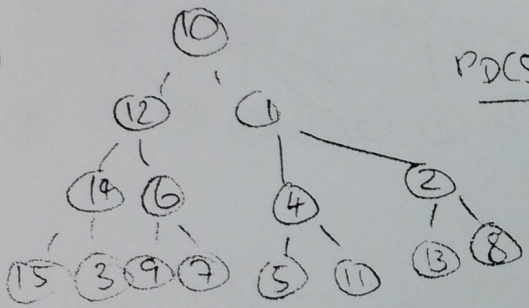
b)



PD(7) →

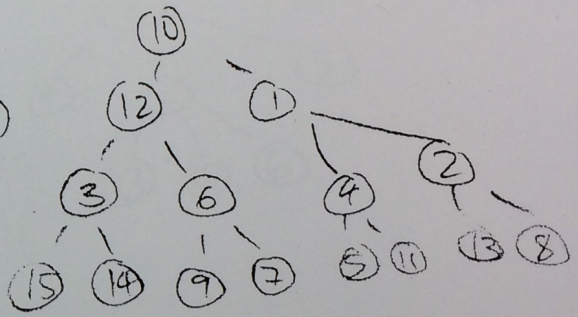


PD(6) →



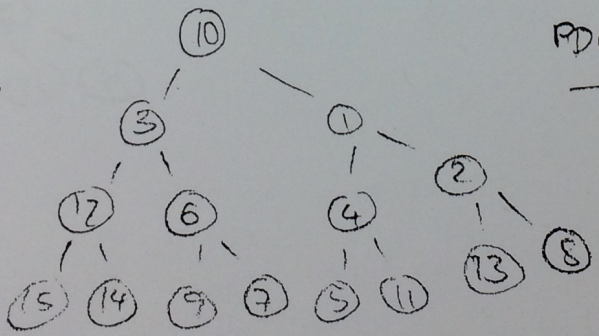
PD(5) →

PD(4) →

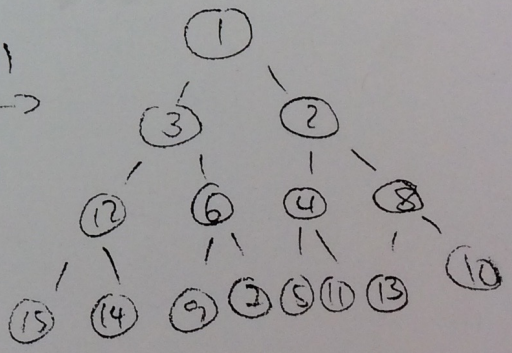


PD(3) →

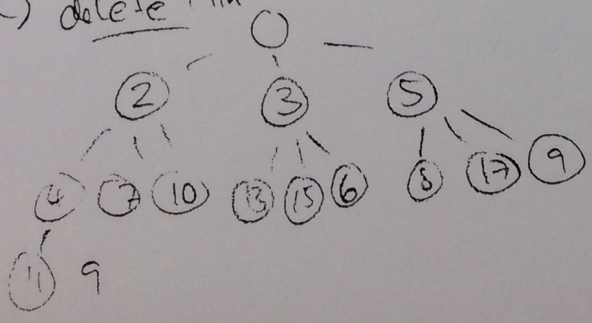
PD(2) →

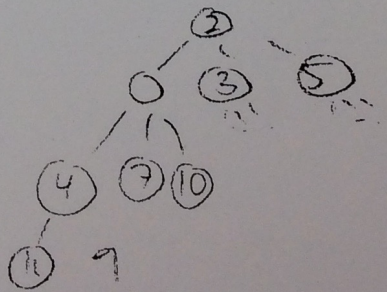


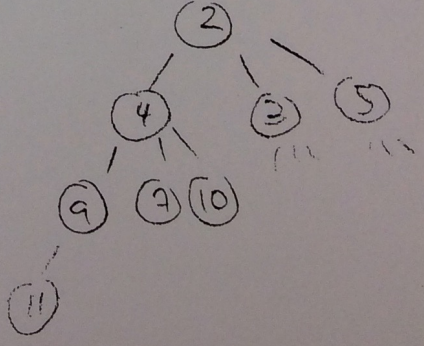
PD(1) →



c) delete Min

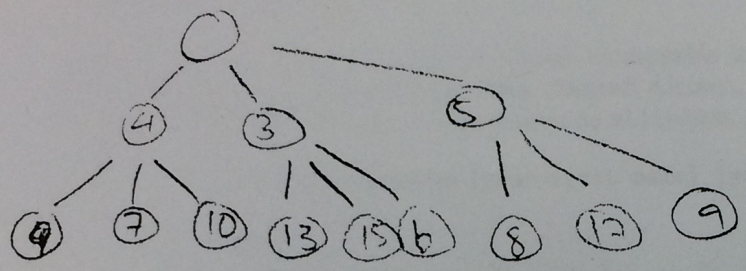


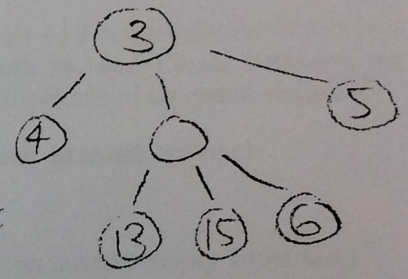


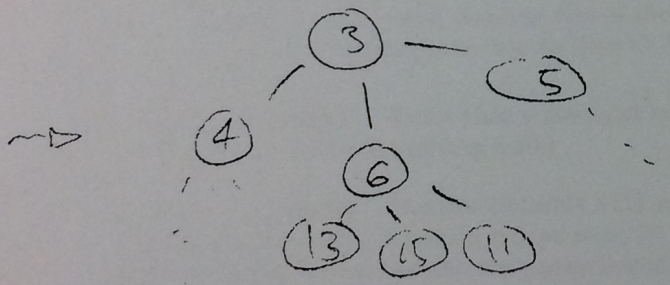


0.5 marks

deleteMin







0.5 marks