Week 5 Section

This week's section handout has practice with trees and heaps. Follow along at Code StepByStep: https://codestepbystep.com/problemset/view/26

1. Tracing Trees

Draw what the resulting binary search tree would be if the elements were added in the following order.

a. 1, 2, 3, 4, 5, 6, 7, 8
b. 8, 6, 4, 2, 1, 3, 5, 7
c. 5, 7, 3, 8, 1, 2, 4, 6

2. Tracing Heaps

Draw what the resulting min heaps would be if the elements were added in the following order. Bonus: what happens after the first dequeue of the smallest element?

a. 1, 2, 3, 4, 5, 6, 7, 8
b. 8, 6, 4, 2, 1, 3, 5, 7
c. 5, 7, 3, 8, 1, 2, 4, 6

Thanks to Marty Stepp and other CS106B and X instructors and TAs for contributing problems on this handout.

3. Height (on CodeStepByStep)

Write a member function named **`height`** that could be added to the `BinaryTree` class. Your function should return the height of a tree. The height is defined to be the number of levels (i.e., the number of nodes along the longest path from the root to a leaf). For example, an empty tree has height 0. A tree of one node has height 1. A tree whose root has one or two leaves as children has height 2, and so on.

Assume that you are adding this method to the `BinaryTree` class as defined below:

```
class BinaryTree {
private:
   BinaryTreeNode* root;   // nullptr for an empty tree
   ...

public:
   // your code goes here
};

struct BinaryTreeNode {
   int data;
   BinaryTreeNode* left;
   BinaryTreeNode* right;
   ...
}
```