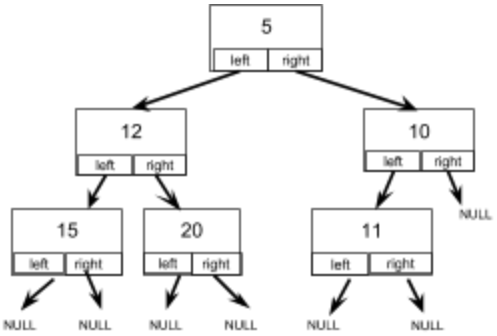
Lab 4 PRE-LAB
COMP15, Spring 2018
Week of 5 March, 2018

About Pre Labs
This pre-lab is designed to ensure that you come into each week's lab prepared and ready to solve the problem in front of you. Read up on abstract classes and templates, and then answer the questions below. Bring the completed paper to your lab section.

About Lab 4
Lab 4 uses Heaps, a specialized type of binary tree. There are two subtypes of heap: min-heaps and max-heaps. There is no such thing as a "plain" heap.

In a MinHeap (example below), every node is smaller than or equal to its children. Other than that, no ordering or values are taken into consideration. In a MaxHeap, every node would be larger than or equal to its children.

A heap is also a complete binary tree: like our original Binary Tree data structure, the tree is filled up from top to bottom and left to right.



Min Heap example. Every node is smaller than or equal to its children and larger than or equal to its parent.

Heaps have two purposes in life: to implement priority queues, and to sort data. We'll do both in Lab 4.

Question 1
This lab will simulate prioritizing patients as they come into an emergency room. The priority of a patient is a number from 1-5. We don't want to make this data type an integer, though, because we want to prevent someone from setting a patient's priority to an unacceptable integer like 100. Instead, we'll use C++ enumerated types to do this.

Declare an enum that we could use for 5 priority levels.

Question 2

The patient class has four attributes that can be used in calculating the patient's priority in the ER:

- chest_pain (bool)
- head_wound (bool)
- temp (double)
- pulse (unsigned) // unsigned int because a pulse would never be negative

How would you use these 4 data points to determine a patient's priority from 1-5? For this lab, you can calculate the priority of a patient any way you like (and later for this week's homework we'll have a more specific way to approach it). Write the code you would put in calculate_priority member function of the Patient class, assuming it also has an attribute called `priority` whose value needs to be assigned in this function.

```
void Patient::calculate_priority(bool chest_pain, bool head_wound,
                                 double temp, unsigned pulse)
{









}
```

Question 3 -- Insert

Once we've got patients, we'll insert them all into a MinHeap. Here's how insert works:

- Put the new item in the next "open" slot, just like we do with Binary Trees.
- Now we need to "heapify up": find the correct place for the new item, where it is larger than or equal to its parent and smaller than or equal to its children. We proceed:
    - Compare the new item to its current parent. If it's smaller than its parent, swap the two.
    - If we made a swap, compare the new item to its current parent (it has a "new" current parent after the swap).
    - Repeat until the new item is in the correct position.

Draw the MinHeap that would result after inserting patients in the following order:

- Hawkeye, Priority 1
- McDreamy, Priority 5
- Elliot, Priority 3
- J.D., Priority 1
- Marcus Welby, Priority 1
- Doctor Who, Priority 4

Question 4 -- Extract

The function a MinHeap specializes in is returning the item with the smallest value. Here's how that operation works:

- Return the item in the heap's root, which is always the smallest.
- Take the "last" item (bottom level, right-most node) and put it in the root position. Call this the *replace item.* We want to "heapify down": go down the heap until the replace item is smaller than or equal to its children and larger than or equal to its parent. We proceed:
    - Compare the replace item to its left and right children. If it's larger than either/both of its children, replace it with whichever child is smaller.
    - If we made a swap, compare the replace item with its new left and right children
    - Repeat this until the replace item is in the correct position

Suppose we remove the minimum item 3 times from the MinHeap you drew in Question 3. Draw the resulting MinHeap after all 3 removals.