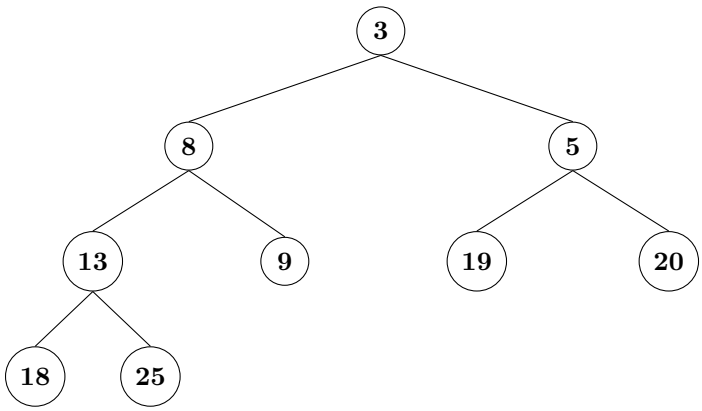
A **priority queue** is a data structure that has the following functions available to it:

- `insert(e)` : inserts the given element into the priority queue.
- `min()` : returns a reference to a smallest element in the priority queue.
- `extractMin()` : remove the element that `min()` would return.

We will be discussing implementing the **priority queue interface** with a *binary heap*, which is a *complete binary tree* with the *min-heap property*.

In Lecture, I suggest you focus on understanding how this works conceptually.

For reinforcement after lecture, try to write pseudo-code for these functions based on your understanding. Chapter 8 of textbook of Goodrich and Tamassia has code, as does your Zybook. You will be writing a priority queue as part of project 5.



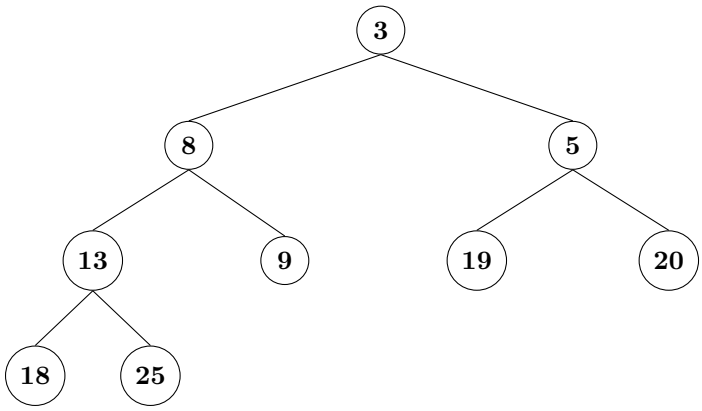
Question 1. How do we insert a key into a heap? What is the above heap if we insert '7'?
Remember: after inserting, we must still have a heap

Question 2. From the result of that, insert the keys 4, 2, 6, in that order. Show the resulting heap and vector after each.

Question 3. How do we implement finding min() in a heap?
Hint: This is the easy part.

Question 4. How do we extract-min from a heap? What do we have after performing an `extract-min` from the following heap?
Remember: after extract-min, we must still have a heap

Perform extract-min several times, showing the heap after each.



Question 5. Suppose you have two min-heaps, A and B, with a total of n elements between them. You want to discover if A and B have a key in common. Give a solution to this problem that takes time $\mathcal{O}(n \log n)$. For this problem, do not use the fact that heaps are arrays. Rather, use the API of a heap (e.g., calls to `A.min()` or `B.removeMin()`). Give a brief explanation for why your code has the required running time.