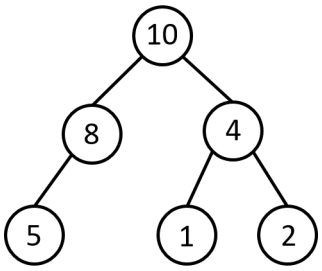
Data Structures
Spring 2019
Midterm 1 Sample Questions

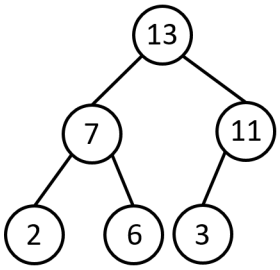
1. For each of the following trees:

   - Is it a heap? Circle **yes** or **no**.
   - If it's not a heap, why not? That is, state the structural and/or organizational properties that are violated (you may list more than one violated property).



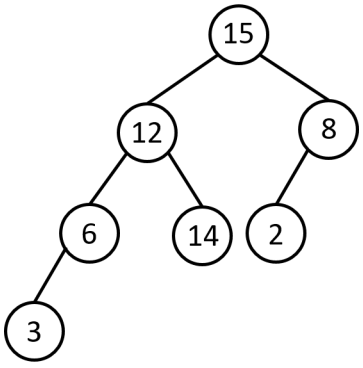
Is it a heap? **YES** **NO**

If not, what property or properties does it violate?



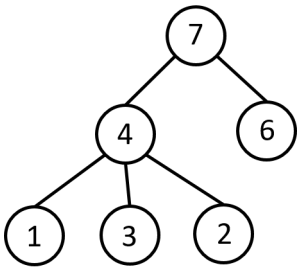
Is it a heap? **YES** **NO**

If not, what property or properties does it violate?



Is it a heap? **YES** **NO**

If not, what property or properties does it violate?



Is it a heap? **YES** **NO**

If not, what property or properties does it violate?

2. Suppose we have available for our use a `PriorityQueue` class that stores a priority queue of `Task` objects, and contains the following methods:

- `int size()`: returns the number of elements currently stored in the priority queue
- `boolean isEmpty()`: returns `true` if the priority queue currently stores 0 elements, and `false` otherwise
- `void add(Task toAdd, int prio)`: adds the `Task toAdd` to the priority queue with a priority level given by the `int prio` (for example, calling `add(myTask, 4)` adds a `Task` called `myTask` to the priority queue and assigns it priority level 4)
- `Task remove()`: removes and returns the highest-priority `Task` from the priority queue

1. Suppose you wanted to use this `PriorityQueue` class to simulate a Stack. What quantity would you use as the priority level when you call the `add()` method to insert a new `Task` into the priority queue? (Don't actually write code to do this, just tell me what the priority level would be. You can add fields to the `PriorityQueue` class if you are so inclined.)
2. Suppose you wanted to use this `PriorityQueue` class to simulate a Queue. What quantity would you use as the priority level when you call the `add()` method to insert a new `Task` into the priority queue? (Don't actually write code to do this, just tell me what the priority level would be. You can add fields to the `PriorityQueue` class if you are so inclined.)

3. Imagine you live in a world where the only data structure in existence is the Stack. You can declare Stack objects, but you can't declare any arrays, queues, priority queues, etc.—nothing but Stacks. Now suppose you want to create a Queue class that will provide the functionality of a queue, but with an underlying implementation that only uses Stacks.

Specifically, consider the scaffolding that I've provided for a `Queue` class below. I have declared two Stacks as fields. Your job is to implement the `enqueue()` and `dequeue()` methods for the `Queue` class *without declaring any additional data structures* (arrays, priority queues, additional stacks, etc.). (This can be done without declaring any additional *variables* at all, but I'll let you declare some `int`s if you'd like.)

```
public class Queue<E> {

    private Stack<E> s1 = new Stack<E>();
    private Stack<E> s2 = new Stack<E>();

    public int size() {
        return s1.size() + s2.size();
    }

    public boolean isEmpty(){
     return s1.isEmpty() && s2.isEmpty();
    }

    public void enqueue(E x) {
        // Fill this in

    }

    public E dequeue() {
        // Fill this in

    }


}
```