1 Heaps of Fun

Assume that we have a binary min-heap (smallest value on top) data structure called Heap that stores integers, and has properly implemented insert and removeMin methods. Draw the heap and its corresponding array representation after each of the operations below:

1. Heap h = new Heap();
2. h.insert(5);
3. h.insert(7);
4. h.insert(3);
5. h.insert(1);
6. h.insert(2);
7. h.removeMin();
8. h.removeMin();
Your friend Sahil Finn-Garng challenges you to quickly implement an integer max-heap data structure. “Hah! I’ll just use my min-heap implementation as a template to write MaxHeap.java,” you think to yourself. Unfortunately, two Destroyer Penguins manage to delete your MinHeap.java file. You notice that you still have MinHeap.class. Can you still complete the challenge before time runs out?

*Hint:* You can still use methods from MinHeap.

Yes. For every insert operation, negate the number and add it to the min-heap. For a `removeMax` operation call `removeMin` on the min-heap and negate the number returned. Any number negated twice is itself (with one exception in Java, $2^{-31}$), and since we store the negation of numbers, the order is now reversed (what used to be the max is now the min).

### 2 Tree Traversals

Write the pre-order, in-order, post-order, and level-order traversals of the above binary search tree.

- **Pre-order:** 10 3 1 7 12 11 14 13 15
- **In-order:** 1 3 7 10 11 12 13 14 15
- **Post-order:** 1 7 3 11 13 15 14 12 10
- **Level-order (BFS):** 10 3 12 1 7 11 14 13 15
3 Quadtrees

3.1 Draw the quadtree built by inserting the following nodes with the given coordinates.

insert A (2, 3);
insert B (-1, 1);
insert C (3, 2);
insert D (0, 0);
insert E (4, 4);
insert F (-3, 2);