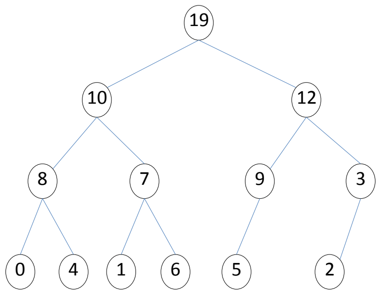
Module IN2002—Data Structures and Algorithms
Answers to Exercise Sheet 3

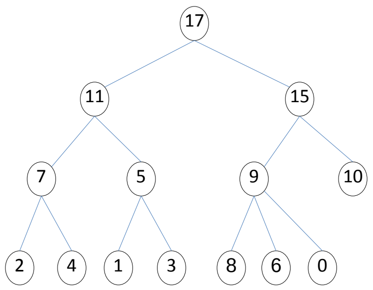
1. Only one of the following trees is a heap. Indicate which one and why the others are not.

a)



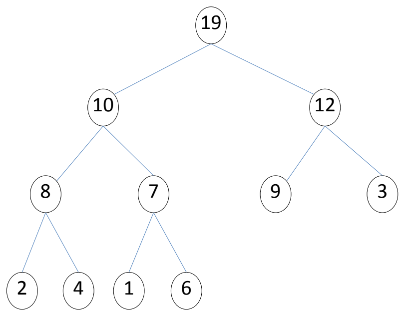
- This is not a heap because the leaves are not as far left as possible: (2) is a child of (3) even though (9) has room for one more child.

b)



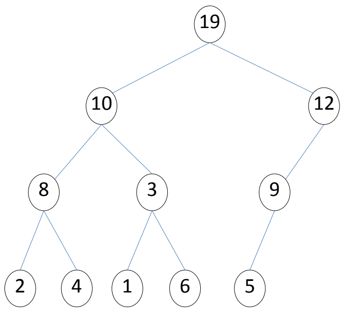
- This is not a heap because it is not a binary tree: (9) has more than two children.

c)



- This is a heap because it is a perfectly balanced binary tree with all its leaves as far left as possible, and with no child larger than its parent.

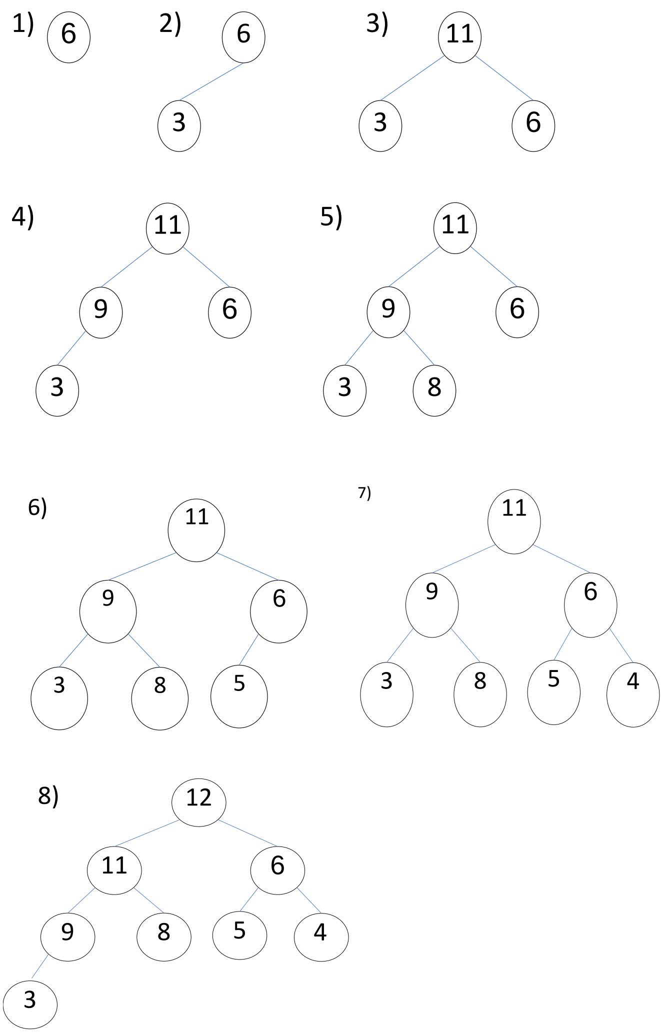
d)



- This is not a heap because (6) is larger than its parent (3) and (12) has only one child (making the tree unbalanced).

2. Show the heaps that are generated as you add the following sequence of keys to an empty heap (one at a time): 6, 3, 11, 9, 8, 5, 4, 12.

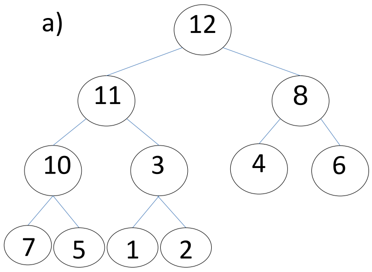
- This is what the heaps look like after adding each element.

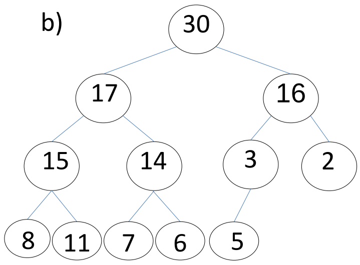


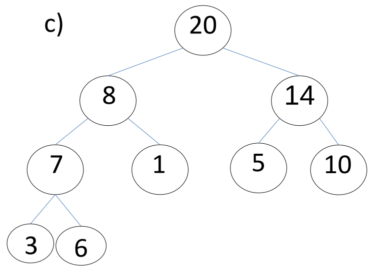
3. Only one of the following arrays is not a heap. Indicate which one.

a) 12 11 8 10 3 4 6 7 5 1 2
b) 30 17 16 15 14 3 2 8 11 7 6 5
c) 20 8 14 7 1 5 10 3 6

- For this, we build the heaps in tree format and check whether the children have lower values than their parents:



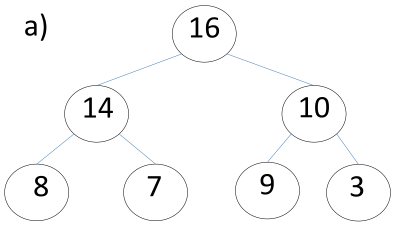




- So "a" and "c" are heaps; and "b" is not ("b" has node "5" as a child of "3", which is smaller.

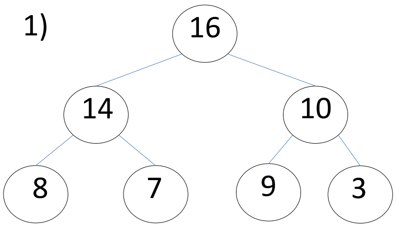
4. Consider heap 16 14 10 8 7 9 3

a. Show it in tree format.

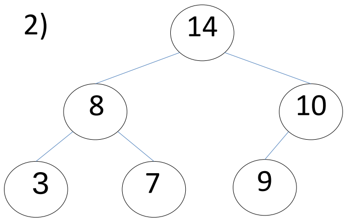


b. Show the heaps that result if extractMax is applied repeatedly until the heap is empty.

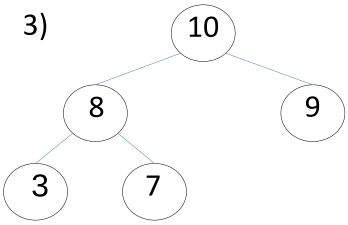
1)



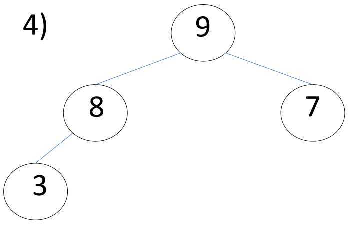
2)



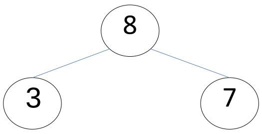
3)



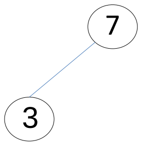
4)



5)



6)



7)

5. Provide pseudocode for a queue implemented using an array. This implies the functions *isEmpty*, *enqueue*, and *dequeue*.

```
public class ArrayQueue implements Queue {
        private int[] a;
        private int count = 0;
        public ArrayQueue(int size) {a = new int[size];}

        public boolean isEmpty() {     return count == 0; }
        public void enqueue(int elt) { a[count++] = elt; }
        public int dequeue() {
                int value = a[0];
                int i = 1;
                while (i<count) {
                        a[i-1] = a[i];
                        i++;
                }
                count--;
                return value;
        }
}
```

6. Write functions `void add(int elt)` and `int extractMax()` in Java, implementing the pseudocode in the lecture nodes.

```
/**
* Add an element to a heap.
* @param elt The element to add.
*/
public void add(int elt) {
	int pos = count; // add after last element
	while (pos > 0 && data[parent(pos)] < elt) { // while
		// heap condition not satisfied
		data[pos] = data[parent(pos)]; // move parent down
		pos = parent(pos); // move to parent position
	}
	data[pos] = elt; // insert elt
	count++; // increment after adding
}

/**
* Get the greatest element, remove it, and reorganise the
* heap.
* @return The greatest element of the heap.
*/
public int extractMax() {
	int max = data[0]; // max is a 0, make a copy
	count--; // decrement count to use it as index
	data[0] = data[count]; // move last elt to 0
	moveDown(data, 0, count - 1); // reorganise
	return max; // return the old max
}

/**
* Reorganizes a heap by moving the elt 'first' down until
* heap condition is satisfied.
* @param data The data array.
* @param first Index of element to move down.
* @param last Index of last valid position in heap.
*/
void moveDown(int data[], int first, int last) {
	while (left(first) <= last) {// determine
		int largest = left(first); // the greater child
		if (right(first) <= last // if greater
			&& data[largest] < data[right(first)])
			largest = right(first); // use right
			// test heap condition
		if (data[first] >= data[largest]) // if heap
			break; // cond. satisfied, we can finish
		swap(data, first, largest); // swap w. greater child
		first = largest; // start again from greater child
	}
}
```

And a bit of programming (note that answers to this will be released much later than for the other questions, giving you time to experiment with it)