Problem 1

> Give an algorithm to find all nodes less than some value, *X*, in a binary heap. Your
> algorithm should run in O(
> *K*), where *K* is the number of nodes output. You need to
> give the main ideas of your algorithm and analyze the running time.

Thoughts:

Look at the root, and determine if lt X. If so recurse through its children

if the root is ≥ X then we can safely prune that branch

```
def'n findLTx(heap, x, index)
    if not heap:  # if the heap is empty, no way to check
        return []
    value ← heap.getValue(index)
    if value < x # recurse through children
        return [value] + findLTx(heap, 2*index + 1) + findLTx(heap, 2*index+2)
    else # prune
        return []
```

Runtime Analyse:

In the worst case this algorithm will run in O(n) time if every node in the heap is less than X.

On the average case it runs in O(k) because we are only going to consider going down other branches if the root node is less than X. If the node is greater than or equal to X we can prune the entire branch, shrinking the search space.