Homework 7

Question 1

minimum: 2^h
maximum: $2^{h+1} - 1$

Question 2

parent node is always $\geq$ number of children
for any subtree the max element must be root of that subtree

Question 3

minimum value of max heap must be in leaf nodes. For heap of size n, leaf nodes are located from index $\left\lfloor \frac{n}{2} \right\rfloor$ to $n - 1$

Question 4

- **Purpose:** maintain max heap property by ensuring that a node is $\geq$ its children
- **Function:** compares node with children and swaps with largest child if necessary, then recursively ensures max heap property for affected subtree
- **Correctness:** only swapping root with largest child ensures largest element is at root of subtree and leaves do not need heapifying

Question 5>

works by calling max_heapify on each non-leaf node starting at bottom of tree; ensures all subtrees are max heaps

Question 6

1. builds max heap from unsorted array
2. repeatedly swaps root of heap with last element and reduces heap size by one
3. calls max_heapify on root to restore max heap property
4. continues until heap size = 1

Question 7

- if binary seach tree node has two children, successor is leftmost node in right subtree, which has no left child
- predecessor is rightmost node on left subtree, which has no right child

Question 8

in-order walk visists each node once, and each call to TreeSuccessor takes constant time, therefore time complexity is $\Theta(n)$

Question 9

- **Binary Search Tree Property:** for any node, all keys in left subtree are smaller, and all keys in right subtree are larger
- **Max Heap Property:** for any node, node's key is $\geq$ key of its children
    - max heap property cannot be used to print keys in sorted order in linear time because extracting max element and maintaining heap property takes logarithmic time for each element; sorting time complexity $= O(n \log n)$