Homework 1: Basic Data Structures and Time Complexity

1. [2 + 3 = 5 points] What does the following algorithm do? Analyze its worst-case running time, and express it using "Big-Oh" notation.

```
Algorithm Bar (a, n)
Input: two integers, a and n
Output:?
k ← n
b ← 1
c ← a
while k > 0 do
    if k mod 2 = 0 then
        k ← k/2
        c ← c * c
    else
        k ← k − 1
        b ← b * c
return b
```

2. [3 points] Algorithm A and B spend exactly $T_A(n) = 0.1n^2 \log_{10} n$ and $T_B(n) = 2.5n^2$ microseconds respectively, for a problem of size n. Choose the algorithm, which is better in the Big-O sense, and find out a problem size n_0 such that for any larger size $n > n_0$ the chosen algorithm outperforms the other. If your problems are of the size $n \leq 10^9$, which algorithm will you recommend to use?

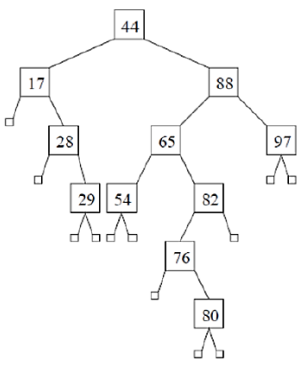
3. [3 points] Describe in pseudo-code a linear-time algorithm for reversing a queue Q. To access the queue, you are only allowed to use the methods of queue ADT. Hint: Consider using an auxiliary data structure.

4. [3 points] Draw a single binary tree T such that
   - each internal node of T stores a single character
   - a preorder traversal of T yields ABDGHEICFJ; and
   - a inorder traversal of T yields GDHBEIAFJC

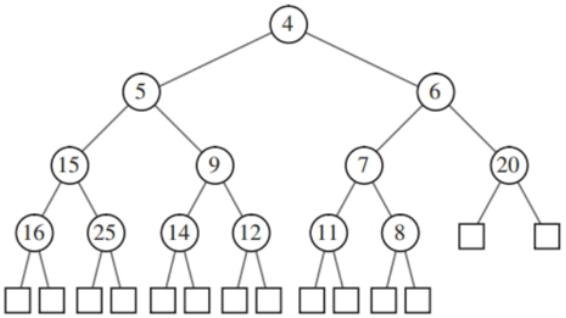
5. [2 points] You are asked to implement a stack. You can use either a singly linked list or doubly linked list. Which one will you choose and why?

6. [3 + 1 = 4 points] Give pseudo-code of the algorithm least-common-ancestor that takes as input two nodes *v* and *w* in a tree T, and that gives as output the least common ancestor of *v* and *w* in T. What is the time complexity of your algorithm in terms of O?

7. [4 points] Remove from the given binary search tree the following keys (in this order): 65, 76, 88, 97. Draw the tree after each removal.



8. [3 points] Let T be a heap storing n keys. Give an efficient algorithm for reporting all the keys in T that are smaller than or equal to a given query key x (which is not necessarily in T). For example, given the heap of below figure and query key x = 7, the algorithm should report 4, 5, 6, 7. Note that the keys do not need to be reported in sorted order. Your algorithm should run in $O(k)$ time, where k is the number of keys reported.



9. [3 points] Given two binary trees, write a function to check if they are the same or not. Two binary trees are considered the same if they are structurally identical and the nodes have the same value.

```
Input:     1         1      Input:     1         1      Input:     1         1
          / \       / \               /           \              / \       / \
         2   3     2   3             2             2            2   1     1   2
        [1,2,3], [1,2,3]           [1,2], [1, null,2]         [1,2,1], [1,1,2]
Output: true                Output: false               Output: false
```