Homework 5

Out: 10.22.20
Due: 11.2.20

1. [Binary Search Trees, 6 Points]
   Draw binary search trees with heights of 3, 4, 5, and 6, containing the following nodes: {5, 7, 9, 10, 12 15 17}

2. [Binary Search Trees, 24 Points]
   For each of the following claims, explain why the claim is correct, or give a counter example:
   a. If the search for key x in a binary search tree reaches a leaf node, then all keys on the left of the search path <= all keys on the search path <= all keys on the right of the search path. For example, if we search for key 3 in the following tree, then the keys on the search path are {4, 2, 3}, keys left of the search path are {1}, and keys right of the search path are {5}.
```
            4
          /   \
        2       5
      /   \
    1       3
```
   b. If a binary search tree has two children, then its successor has no left child, and its predecessor has no right child.
   c. If we delete node x and then node y from a binary search tree, we will end up with the same resulting tree that we will get if we first delete node y and then node x. Assume that we utilize the Delete method seen in class.

3. [Hashing, 20 points]
   Insert the following keys into an initially empty hash table of size 11:
   89, 19, 50, 59, 70, 26.
   Show the resulting hash table, using the division method to generate hash values with hash function h(x) = x mod 11.
   a. Using open hashing (i.e. chaining).
   b. Using closed hashing (open addressing) with linear probing.
   c. Using closed hashing (open addressing) with quadratic probing, $C_0=0$, $C_1=C_2=1$.
   d. Using double hashing with the secondary hash function $h_2(x) = 5 – (x \bmod 5)$.

4. [Cuckoo Hashing, 50 points]
   Implement a Cuckoo hash table in which two separate tables are maintained. Each table should have a fixed size of 13. Use the hash function *$h_1(x) = x \bmod 13$* for the first table, and *$h_2(x) = 11 – (x \bmod 11)$* for the second table.

   Your program should read an input file, *input.txt*, consisting of one input per line, insert the input to the table in order, and print out the final content of the hash table in the provided format. Your implementation should utilize the provided

CuckooHashTable.h file as is, without any edits. Note that the hash tables are implemented as a 2x13 matrix (vector of vectors), and that a print method declaration is provided, for you to implement.

Inserting an element that already exists in the table should have no effect. Your program should be able to detect if an insert results in an infinite loop in the hash, and exit with the following error message: "Error: Insert causes infinite loop".

Sample *input.txt* file content:

```
    1
    4
    6
    27
    123
    14
    17
    195
```

Sample output for the above input:

```
Table 1:
    195
    14
    -
    -
    17
    -
    123
    -
    -
    -
    -
    -
    -

Table 2:
    -
    -
    -
    -
    -
    6
    27
    4
    -
    -
    1
```

-
-

Submit your solution, which should be contained in a single *CuckooHashTable.cpp* file, and should be compiled with the provided *CuckooHashTable.h* and *main.cpp* files.

Make sure to include an explanation of your approach in a comment at the top of the program.