University of Waterloo
CS240 Fall 2020
Programming Question 2

Due Date: Wednesday, Oct 28 at 5:00pm

The integrity of the grade you receive in this course is very important to you and the University of Waterloo. As part of every asessment in this course you must read and sign an Academic Integrity Declaration before you start working on the assessment and submit it **before the deadline of October 28th** along with your program; i.e. **read, sign and submit PQ02-AID.txt now or as soon as possible**. The agreement will indicate what you must do to ensure the integrity of your grade. If you are having difficulties with the assignment, course staff are there to help (provided it isn't last minute).

The Academic Integrity Declaration must be signed and submitted on time or the assessment will not be marked.

Please read `http://www.student.cs.uwaterloo.ca/~cs240/f20/guidelines.pdf` for guidelines on submission. Submit the file `avlvsskip.cpp` to MarkUs.

Problem 1 [10 marks]

In this programming question you are asked to implement both an AVL tree and a skiplist to compare the number of key comparisons made as items are inserted, searched for and deleted. Your implmentations must follow the algorithms given in class so the number of key comparisons will match. Your runtimes should also match those discussed in class. For simplicity, the key and value of an item will be int values. Also, you do not need to consider duplicate key values within the data structures - we will not attempt to insert a key that already exists in the data structure.

You may not use any pre-existing code that would trivialize the implementation (e.g. built in data structures from STL, smart pointers, etc). You may, however, use the C++ vector data structure. If in doubt, make a private Piazza post and ask.

Implement your program in C++ and provide a main function that accepts the following commands from stdin (you may assume that all inputs are valid):

- `i key value coin` - inserts an item $(key, value)$ into both your AVL tree and skiplist and prints the number of key comparisons required by AVL.insert and skipList.insert, respectively, followed by a newline. Use the value *coin* as the number of heads flipped before a tail is reached when inserting into the skiplist.

- `d key` - deletes the item with the given key from both the AVL tree and skiplist and prints the number of key comparisons required by AVL.delete and skipList.delete, respectively, followed by a newline. For a BST, delete should use the inorder successor (if needed).
- `savl key` - searches the AVL tree for the item with the given key and prints the corresponding value or string `ERROR` if not found, the number of key comparisons used in the search, followed by a newline.
- `ssl key` - searches the skiplist for the item with the given key and prints the corresponding value or string `ERROR` if not found, the number of key comparisons used in the search, followed by a newline.
- `stats` - prints 3 int values followed by a newline corresponding to: number of items in the AVL tree/skiplist, height of the AVL tree, and total number of nodes in the skiplist (do not count $+\infty$ or $-\infty$).
  The runtime for this operation is $\Theta(1)$ time.
- `r` - initializes an empty AVL tree and skiplist. If a nonempty AVL tree or skiplist already exists, they are destroyed and new empty data structures are created.

Place your entire program in the file `avlvsskip.cpp`