CSCI 2100: Project

Prepared by Yufei Tao

This coding project should be attempted by every student individually. The objective is to implement the AVL-tree, and more specifically: its insertion and predecessor-query algorithms. Each submission will be graded based on correctness and efficiency. The rest of the document explains the details.

How Your Submission Will Be Tested. You will be given a sequence of operations, each of which is one of the following types:

- insertion: which adds a new integer v to the dataset S (which is empty at the beginning);
- predecessor query: which finds the predecessor of an integer q in the current S.

The sequence will be given in a file named "ops.txt", where the i-th ($i = 1, 2, ...$) line describes the i-th operation as follows:

- If the operation is inserting v, the line is "`ins` v" (namely, starting with the letters `ins`, followed by a space, and then by the value v).
- If the operation is a query with value q, the line is "`qry` q".

For example, if the sequence has 5 operations — insert 5, 10, query with 3, insert 7, and finally, query with 9 — then the file looks like:

```
ins 5
ins 10
qry 3
ins 7
qry 9
```

Your program should build an AVL-tree from scratch, and use the tree to answer queries. Specifically, (i) every time you see an "`ins` v"' operation, you should insert v into the tree, and (ii) every time you see a "`qry` q", you should find the predecessor of q, and write the answer to a file named "**output.txt**" (if the predecessor does not exist, write "no"). For example, after processing the aforementioned 5 operations, your file should have two lines:

```
no
7
```

The sequence used to test your program will contain 1,000,000 insertions and 1,000,000 queries, which may be interleaved in an arbitrary manner. The first half of the sequence will be made available for download on the course homepage, whereas the other half will be hidden from you.

Programming Language. C++ (including variants like C, C#, ...), Java, or Python. Your implementation must be from *scratch*, namely, you can use only functions from a standard library:

- C++: Standard MSDN library (msdn.microsoft.com/en-us/library/cscc687y.aspx) or GNU library (www.gnu.org/software/libc).
- Java: Standard edition 6 (docs.oracle.com/javase/6/docs/api).
- Python: The Python standard library (docs.python.org/3/library).

Grading Scheme. Your score will be calculated as follows:

- 0 marks, if your source code cannot be compiled.
- 0 marks, if you did not implement *by yourself* the insertion and predecessor-search algorithms of the AVL-tree.
- We will first obtain the number m of queries that your program answers correctly.
- (C++/Java) If your program finishes within 10 seconds—this time covers the cost of *everything*, including file reading and writing—then you final score will be $m/(10000)$. If your program finishes within 20 seconds, your final score will be $m/(20000)$. If your program takes more than 20 seconds, 0 marks.
- (Python) Replace the numbers 10 and 20 in the above bullet with 100 and 200, respectively.

Submitting Your Source Code. The deadline of submission is 11:59pm, **Dec 16, 2018**. Late submissions will receive 0 marks. The detailed arrangements will be announced later.

Zero Tolerance for Cheating. You are required to do the implementation on your own. This means, in particular, that you must be able to indicate the code for the insertion and predecessor-search algorithms which **you wrote by yourself**. You may be required to attend an interview to explain details about the code. All submissions must pass through Veriguide, and may be subject to additional plagiarism checks. Any confirmed case will receive a 0 mark outright, and be reported to the university for disciplinary actions.