COMP 3240
Programming Assignment 2
Automating Exhaustive Counterproofs

Daniel Tauritz, Ph.D.

February 15, 2023 – Rev1

Synopsis

The goal of this assignment is for you to apply your understanding of predicate logic to automate an exhaustive counterproof approach for the magic numbers hypothesis. These are individual assignments and plagiarism will not be tolerated. You must write your code from scratch in either Python 3 or Java.

Problem statement

A positive integer is "magic" if, and only if, it can be reduced to 1 by repeatedly dividing it by 2 if it's even or multiplying it by 3 and then adding 1 if it's odd. So, for example, 3 is magic because 3 reduces first to 10 ($3 \cdot 3 + 1$), then to 5 ($10/2$), then to 16 ($5 \cdot 3 + 1$), then to 8 ($16/2$), then to 4 ($8/2$), then to 2 ($4/2$), and finally to 1 ($2/2$). The magic numbers hypothesis states that all positive integers are magic, or, formally: $\forall x \in \mathbb{Z}, MAGIC(x)$ where $MAGIC(x)$ is the predicate "x is magic". Write a Python 3 or Java program to automate the search for a counter example to the magic numbers hypothesis for user definable ranges of positive integers.

To code this, you should modify the given Python or Java file. Your function should accept two integers that will act as the lower and upper bounds respectively for the range of numbers that you should search. Upon completion, your function should return an integer as follows:

- If you find a counter example i in the specified range that is *not* a magic number, return $-1 \cdot i$ (i.e., the negative of the non-magic number).
- Otherwise return the number of steps (where $x/2$ or $3x + 1$ each count as one step) it took to reduce the number to 1.

For example:

- the number 1 would generate the reduction sequence: 1, which is zero steps.
- the number 2 would generate the reduction sequence: $2 \rightarrow 1$, which is one step.
- the number 3 would generate the reduction sequence: $3 \rightarrow 10 \rightarrow 5 \rightarrow 16 \rightarrow 8 \rightarrow 4 \rightarrow 2 \rightarrow 1$, which is seven steps.

Resubmissions, penalties, documents, and bonuses

If you submit before the deadline, then you may resubmit up to a reasonable number of times till the deadline but not thereafter, your last on time submission will be graded. If you do not submit before the deadline, then your first late submission will be graded.

The penalty for late submission is a 5% deduction for the first 24 hour period and a 10% deduction for every additional 24 hour period. So 1 hour late and 23 hours late both result in a 5% deduction. 25 hours late results in a 15% deduction, etc. Not following submission guidelines can be penalized for up to 5%, which may be in addition to regular deduction due to not following the assignment guidelines.

Some assignments may offer bonus points for extra work, but note that the max grade for the average of all assignments is capped at 100%.

Submission Details

Please ensure that you have not changed the headers of any of the methods or classes or else the autograder will not run properly. If you are using the Java version of this code, note that **you should *NOT* declare any packages** like you did in Programming Assignment 1. Also ensure that your class and its methods are all marked public, and make sure that the methods (not the class) are marked static.

Deliverables & Due Date

The deliverables of this assignment are:

1. your source code with your name and the string “COMP 3240 Programming Assignment 2” at the top of the file

Please name your file `Assignment2_Java.java` or `Assignment2_Python.py` and make sure to rename the class to `Assignment2_Java` or `Assignment2_Python`. Submit your file directly to Gradescope–no need to compress or zip. The due date for this assignment is 10:00 PM on February 24, 2023.

Grading

The maximum number of regular points you can get is 50. The point distribution is as follows:

Algorithmic	80
Good programming practices including code reliability and commenting	20