# CSCI 36500 Spring 2025
# Turing Machine Simulator

Due on Wednesday, March 13 at 7:00pm EST

CSCI 72400 students can do this if they like, but it will not replace their paper nor count to their grade. CSCI 36500 students *must* do this.

Write a one tape, one track, two-way infinite tape, deterministic Turing Machine simulator.

Once running, your program should ask for a file name, which contains the "code" of the TM as input. It should then ask for an input word and output the sequence of IDs (one per line) that the TM goes through on that input. When it gets to a final state it should clearly indicate that the string was accepted. If it gets to any other state, symbol pair with no transition it should stop computing and indicate the string was not accepted. If looping, the TM should be stoppable by your choice of input. Be sure to indicate what that is. This should not crash the simulator, but should allow the user to input another string to the same TM. In any of these cases the user should then be prompted to run the TM on another input string.

The input file consists of multiple lines. Each line is either blank, a comment indicated by "//" or a 5-tuple consisting of
currentState currentTapeSymbol newState newTapeSymbol direction
followed by an optional comment, also indicated by "//". States are indicated by either an integer (with 0 representing the start state $q_0$) or the letter f for Final state. $\Sigma = \{0, 1\}$, B is "blank". You can construct the rest of $\Gamma$ and $Q$ from the input file (but you don't need to display them explicitly.)

The output should consist of the sequence of IDs the machine goes through. Indicate the states with []s, So, for example, the first ID on a computation on word 1010 would be [0]1010.

You can find two commented sample programs:
http://www.cs.hunter.cuny.edu/ eschweit/365stuff/zero-one-starTM.txt for $\{01^*\}$ and
http://www.cs.hunter.cuny.edu/ eschweit/365stuff/0n1n0n.txt for $\{0^n1^n0^n\}$.

You may use any programming language you like as long as I can interpret or compile it and run it. If it is reasonably recent C++ and is one file you can submit just that. If it is C++ and more than one file please tar the files and supply a make file. If you are writing in python, be sure to indicate which flavor of python as well as taring up multiple files. If you want to use another language please be sure I can run your code. If you program in MacOS, be sure to delete all the evil __MACOSX files and other hidden files and directories. They are annoying, so including them will cost a point. (If you use a Mac and don't know about them or the file structure used in MacOS,

you should.) In any case be *SURE* the name of the executable file is "run.me" and that it lives in the same directory as the (top level) source file. The arbiter of "compiles and runs" are the cslab machines. Be SURE to test your code there.

In addition to the program, you should write a one or two page paper that explains what your simulator does and how it does that. You should address this to an audience of non-CS people who have an idea of what a program can do. Grammar, spelling and overall use of English will count for $\frac{1}{2}$ the paper grade, a clear description of what and how is the other half. Writing in LaTeXis a good idea, and worth a point.

Bring a printed copy of your paper to class. Please do not include a cover page or folder. Have your name at the top of the first page and a single upper left corner staple. Email your source code as described above to eric.schweitzer@hunter.cuny.edu before the deadline. Be sure to have your name in the filenames. firstLastSomething.extension would be ideal (so for example I would send ericSchweitzerTM.cpp). Anything less than ideal will earn less than the maximum points.