Bryan Dong
24414164

- I spent quite some time on this project. I can't give you a specific amount of time because I didn't measure that at all. Probably the best thing I can say is more than 5 hours. I'm not sure if that's representative at all, especially because it wasn't all done right in one or even some long sessions.
- I went sequentially down the functionality that would be needed for a Turing Simulator. I didn't use many functions and no user defined classes because they didn't seem necessary for a simulation of a TM once per run. The first thing was to read the machine file and store all the information about the machine in some way to access it when simulating. Then you read the input file and save all the problems, which are each a line of input for each tape of the machine. So a 5 tape has problems of 5 lines and the file could have many problems. Then you loop through those problems. Make the lines into arrays to act as tapes, check for errors, and then initialize the starting state and head location. Then you can actually simulate the running of the TM on these starting tapes using a loop. Look for the correct rule that matches all the states and heads while doing error checking, and then do the rule that we got. That means go to that state, write on each tape and then move the head, as necessary. If a condition is met that halts, then we halt. The loop then goes to the next problem. I didn't use github because it's a program in 1 file, with only myself working on it. The only real benefit would be undoing stuff between sessions? But I doubted I would actually need to undo that much progress.
- I used python 3.13 with libraries sys and csv. I don't think any code I wrote should not work because of a different python version, but if it doesn't work on your end a different version may be the cause. Attached along with the simulator are the two Turing machines I created, along with some sample test cases. These are text files with the machine and input following the example format given by the project.
- I used so many small variables that are worth writing about here like the counters for for loops. At the start, when getting the TM information, I saved many things to variables that maybe get used once like the tmName being used solely to print the name of the TM out once. Other easy variables include things like tmMaxSteps, tmMaxTapeSize, tmAcceptState, etc which should be self-evident what they hold for later checking. The alphabets of the input and the tape are sets with the input being 1 set, and the tape alphabet being an array of sets TapesAlphas. The first interesting variable is the array of rules which I had to set up to go into transitionRules. Each rule is actually an array of size 3, with input, output, and rule number. [0] of a rule is the input and has current state and read head values. [1] of a rule has the next state, the write head values, and the direction for each head. Then [2] is just the rule number. inputLines is an array of strings that just holds all the input lines so i can easily store all the input file's lines for the later loop to access only the lines that matter to the current problem. That is why TapeStarts takes the number of tapes lines from that array. TapeStarts is initialized as the starting tapes of the problem, with each tape as an array of size(max tape length) where each index holds a char. This continues to act as the tape during simulation of the machine solving the problem. currHead is an array of int that holds the head position of each

head currently. It is initialized with all 0s. Some easy to understand variables used for checking the current progress are currState for seeing which state you're on, currSteps to make sure the simulation doesn't go too many steps. The output works fine meeting requirements, but csv output via console of > .csv is slightly weird in making each value a separate column. Read the top of my python file to find out how to make that show properly, if you care about csv. It works but it's weird that way to navigate. The trace files of executing my machines are in txt, so you can convert that to csv in that same way.

- I did not create any extra machines other than the 2 that were required. The readme of each individual machine will have further information about the purpose and functionality of each of those machines. These refer to TM-even-accept-test.txt and TM-computation-copy-test.txt.