# CS 4110
# Turing Machine

Write and test a commandline program that implements a Turing Machine as specified in a ASCII file, and then report if that machine will accept or reject a given WORD. The file and word are presented to your program on the command line in this manner:

```
<yourprogram> <TMAsciiFile> "WORD"
```

Where `<yourprogram>` may be named as you see fit (except you Java devs, see below), the `<TMAsciiFile>` is a file of Turing machine transitions (see below) and `"WORD"` will initialize your Turing Machine input tape. Your output from should include one of the following:

```
Accepted: WORD
```

OR

```
Rejected: WORD
```

where WORD was the word presented to your program. The Turing Machine program will be presented to your program in an ASCII text file of the following format:

```
# this is a comment
FROMSTATE, LETTER, NEW_LETTER, DIRECTION, TOSTATE
```

Items are separated by a comma and a space (", ") and can contain imbedded spaces. No double-quotes are used to surround fields. A BLANK is always represented in the program file as an underscore ('_').

Here is a sample program file (note that not all lines have 5 columns).

```
# double a, from text, p. 446
1 Start, _, _, R, 1 Start
1 Start, b, b, R, 1 Start
1 Start, a, a, R, 2
2, a, a, R, Halt 3
2, b, b, R, 1 Start
Halt 3
```

# CS 4110
# Turing Machine

Any FROMSTATE that contains the string "halt" is a Halt State. The FROMSTATE containing the string "start" is the…? The order of the lines in the program file should not be not important and sample Turing Machine files will be provided for testing.

Languages supported: Java, Swift, C# or Script. Naming of the source files is up to the student, unless it is written in Java. Script programs are ASCII files with a correct "shebang" on line 1, such as "#!/usr/bin/python3"

Java developers: the .java file that contains Main() shall have class TuringMachine and of course, be named TuringMachine.java.

Getting started:

Your program must reside in a Github repo that is created when you click on this link: https://classroom.github.com/a/fACH0tfm

A Jenkinsfile and two sample TuringMachine files are provided in a public repo you can clone: https://github.com/tcowan/cs4110tmtemplate.git

Place this Jenkinsfile in the root of your repo and customize as provided in the Jenkinsfile comments. Do not rename the Jenkinsfile. Do. Not.

I reserve the right to use my own special Turing Machine files and words on your program, so manually test your programs completely. I promise that the Turing Machine files will be syntactically correct. Cucumber will run on the Jenkins server to exercise your code and award points. I also reserve the right to inspect your code for proper implementation and execution of the Turing Machine.

Read the Grading Rubric below for more details on the requirements.

There are a total of **200** points possible for this assignment. Did I mention that you must not rename the Jenkinsfile?

# CS 4110
# Turing Machine

## Grading Rubric

Here is how you earn points for this assignment:

| **FEATURE** | **POINTS** |
|---|---|
| Program successfully compiles on the Jenkins server using the Jenkinsfile in the student's repo. The student is permitted to modify the Jenkinsfile Build step only to name the source files to produce an executable for the following program types: script, C#, Swift and Java. | 10 |
| The build step successfully creates an executable runnable by Cucumber on the Jenkins server. | 10 |
| Program implements every valid Turing Machine of the form documented in Chapter 19 of the text (excluding INSERT and DELETE), and implements an updatable input tape, state transitions and tape head movement, without crashing or looping. | 20 |
| Program rejects the appropriate words without crashing or looping and produces a proper Rejected: WORD message when the Turing Machine finds no transition for the input character currently under the tape head. | 40 |
| Program rejects the appropriate words without crashing or looping and produces a proper Rejected: WORD message when the Turing Machine is requested to make an invalid tape head movement. | 40 |
| Program rejects the appropriate words without crashing or looping and produces a proper Rejected: WORD message if the Program exceeds 1,000 transitions. | 40 |
| Program accepts the appropriate words without crashing or looping and produces the Accepted: WORD message | 40 |
| **Grand Total** | **200** |