Problem 0. Document how much time you spend on each of the following problems and cite any resources you received help from.

Problem 1. Previously we encoded the problem of adding two numbers as a language. Here we will encode the problem in a simplified way, and we begin by defining the alphabet for the language. Define the digit symbols $\Sigma_d = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$, and let $\Sigma = \Sigma_d \cup \{\#\}$ be the alphabet for this problem. A string $w \in \Sigma_d^*$ is said to be a *well-formed number* if it consists of a single 0 symbol or a non-zero digit followed by zero or more digits—in other words, if $w \in \Sigma_d^*$ is recognized by the regular expression $0 \cup (\Sigma_d \setminus \{0\}) \circ \Sigma_d^*$.

Now we can define the language

$$\begin{aligned}\mathrm{ADD}_2^{\#} = \{w_1\#w_2\#w_3 \mid\ & w_1, w_2, w_3 \in \Sigma_d^* \text{ are well-formed numbers} \\ & \text{and } w_1 + w_2 = w_3\}.\end{aligned}$$

For example, the string $23\#17\#40 \in \mathrm{ADD}_2^{\#}$ because $23 + 17 = 40$. However, $1\#2\#4$ is not because the third number is not the sum of the previous two numbers.

Prove that $\mathrm{ADD}_2^{\#}$ is not regular using the pumping lemma.

The rest of this assignment is all about designing and working with Turing machines. Many of the problems require you *construct* a Turing machine to decide or recognize a language. The format required for these constructions is the simulator used in class:

`http://morphett.info/turing/turing.html`

You may use the standard "two-way infinite tape" model or the Sipser "semi-infinite tape" model.

The code for these Turing machines must be emailed to

`titus.klinge@drake.edu`

by 11:00am of the deadline. Please use the subject `[CS 139] Assignment 5` and attach each solution separately with a sensible name such as `hw5-p2.txt` for your Problem 2 solution. Note that your solution to Problem 1 should still be uploaded to Gradescope as usual.

Please use descriptive state names and comment your code so that it is easier to follow.

Problem 2. Construct a Turing machine that decides the language

$$A = \{\texttt{010}\},$$

over the alphabet $\Sigma = \{0, 1\}$. (Yes this language contains only one string.)

Problem 3. Construct a Turing machine that decides the language

$$B = \{\texttt{a}^n\texttt{b}^n\texttt{c}^n \mid n \geq 0\}.$$

over the alphabet $\Sigma = \{\texttt{a}, \texttt{b}, \texttt{c}\}$.

Problem 4. Recall that the *self-delimited pair* of two strings $x, y \in \{0, 1\}^*$ is the string

$$p(x, y) = 0^{|x|}1xy.$$

Construct a Turing machine that interprets its input as a self-delimited pair of strings $p(x, y)$ and then outputs the string x. By "output," we mean that when the Turing machine accepts, the only non-blank symbols left on the tape forms the string x. (Note that blank spaces to the left of x are permitted.) If the input is not a properly formatted self-delimited pair, your machine should reject.

Bonus Problem (Extra Credit)**.** Construct a Turing machine that interprets its input as a self-delimited pair of strings $p(x, y)$ and then outputs the maximum of x and y. (Note the values being compared are the unsigned integer values of the bits of x and y.)