# DFA Simulator

To run the simulator, the file dfa.py (one of the files you copied above) needs to be in the current directory:

python3 dfa.py *<DFA file> <strings file>*where <*DFA fil*e> is a file that describes the DFA (and the file you submit for each problem) and

<*strings file*> is a set of test strings. The format for these files is described on the next page.

The *DFA file* described the DFA using the following format:

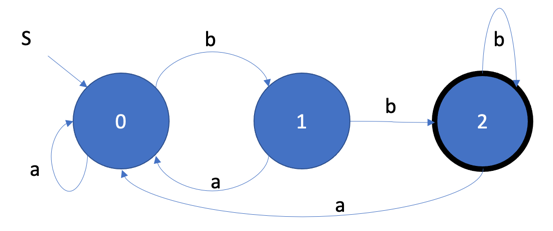
* The first line will contain one or more characters that define the input alphabet.
  + Each character in the input alphabet is separated by a single space.
  + Each character in the input alphabet is a unique *single* letter or digit.
* All subsequent lines will each refer to a different state in the DFA. The second line in the file (the first line after the input alphabet definition line) will be state 0. The third line in the file will be state 1, the fourth line in the file will be state 2, and so forth.
  + The first character in the line will either be a '+' (accepting state) or '–' (rejecting state).
  + Then there will be *n* nonnegative integers where *n* is the number of characters in the input alphabet. Each integer refers to the destination state for a transition for the corresponding input character.
  + The corresponding input character is based on the order of the states and the order of the input characters on the first line of the file. If the input alphabet line is "D O G". The first integer the destination state for input D, the second is for input O, the third is for input G.
  + The integers are separated by a single space. There is also a single space between the first character (+ or –) and the first integer.
* State 0 is the starting state.
* The simulator does not have any explicit error checking for incorrectly formatted files. Many errors (but not necessarily all of them) will cause an exception to be thrown, aborting the program.

The *strings file* contains one or more strings that will be executed in the DFA.

* Each line will be interpreted as a test string.
* Each string will be processed by the DFA and the simulator will print out the final state along with whether the string is accepted or rejected.
* The lines in the string file must only contain characters from the DFA alphabet. If a letter outside the alphabet is found, a keyError exception will abort the script.
* You will need to create your own test scripts but there is no requirement to hand them in.

Files dfa1.txt, dfa2.txt, and dfa3.txt are provided. They contain a single line with the input alphabet for that problem.

Here is a sample DFA that corresponds to the following DFA (note that state 0 must always be the start state):



The corresponding file, say dfa\_example.txt. First line is the alphabet, next line is state 0, next line is state 1, and the last line is state 2.

a b

- 0 1

- 0 2

+ 0 2

Then with the following strings\_example.txt

abb

abababaabb

aab

the output of the program would be:

'abb' ends in state 2: accepted

'abababaabb' ends in state 2: accepted

'' ends in state 0: rejected

'aab' ends in state 1: rejected