COMP 370, Fall, 2020
Program #1, DFA Simulation
Date Assigned: Wednesday, August 26
Date Due: Thursday, September 10, 10PM
Possible Points: 100

For this assignment you will write a Python 3 program (in a file named `pa1.py`) that simulates the computation of a DFA M on an input string s, and reports if s is accepted by M. (That is, if s is in the language of M, or $s \in L(M)$.) Your program should behave exactly as described further below.

1 Initial Setup

Both you and your partner will need to clone the starter code for your group using git. The following assumes that you are using VSCode as your IDE. If you are using some other IDE, follow the instructions for that IDE to clone a repository, or just use command line git ("git clone url").

1. You will need your group number in what follows. Get this from the PSA Group Numbers file under the Programming Assignments tab on Blackboard. This file also has your assigned partner.

2. In VS Code, open the command palette and select the Git Clone option.

3. When prompted for the repository URL, enter the following, with X replaced by your group number (e.g. 7 or 12):
   ssh://git@code.sandiego.edu/comp370-fa20-pa1-groupX

4. Choose the Open Repository option in the window that pops up in the lower-right corner of the screen. The repository should contain the following files:

   - `pa1.py`: You will write your code in this file. It includes the header for the `DFA` class that you will write, and the headers for the constructor and `simulate` methods that you must add.
   - `test_pa1.py`: This program will test your `pa1.py` program. See the discussion below on this.
   - `dfai.txt` for i = 1, ..., 10: Test dfa files.
   - `stri.txt` for i = 1, ... 10: Test input strings for the corresponding dfa's.
   - `correcti.txt` for i = 1, ... 10: Correct answer for the test input strings for the corresponding dfa's

5. Each programming team will have it own repository that has been initialized with the same starter code, so when you sync your code, you are sharing with your partners, but with no one else in the class. (I can see your repository also.)

6. Remember that if you close VSCode, then when you reopen it, you should see your repository. But if you dont see it, then just select `File->Open`, and then select the directory containing your repository.

7. I also recommend that you stage changes, commit those changes, and sync the changes periodically as you are working on the program, and certainly when you are done with a session with your programming partner. This ensures that you wont lose any of your work in case your computer gets lost or a file gets accidentally deleted.

2 Program Specifications

As said above, your program should be contained in a file named `pa1.py`. In this file you should define a class called `DFA` (all capital letters). `DFA` should have a constructor that takes as its parameter the name of a file, and initializes the `DFA` from the contents of the file. (Specifications for the file format are further down.) `DFA` should also have a method that takes as its parameter a string, and returns `True` if the string is in the language of the DFA, and `False` if not. The project repository contains a `pa1.py` file that has the headers for these.

The format of the DFA description file is as follows:

1. An integer that is the number of states in the DFA. This appears by itself on the first line of the file. (In the remainder of the description, each state must be referred to by an integer in the range $[1, 2, ..., n]$, where n is the number of states.)

2. The alphabet of the DFA. This appears by itself on the second line of the file. Every character in the line (not including the terminating newline character) is a symbol of the alphabet.

3. The transition function of the DFA. There will be one line of input per entry in the transition function table, starting with the third line of input. The format of an entry is

   ```
   qa 'c' qb
   ```

   This entry indicates that if the DFA is in state `qa` and the next symbol scanned is a `c`, then the DFA transitions to `qb`.

   `qa` and `qb` must be valid states; that is, qa $\in \{1, 2, \ldots, n\}$ and qb $\in \{1, 2, \ldots, n\}$. `c` must be in the alphabet, and the single quotes must be present.

   The entries of the transition function can appear in any order.

4. An integer that is the start state of the DFA. This appears by itself on the first line after the transition function lines.

5. The set of accept states of the DFA. These appear together on the line following the line containing the start state.

Multiple entries on a line are separated by 1 or more whitespace characters. The first entry on a line is preceded by 0 or more whitespace characters, and the last entry on a line is followed by 0 or more whitespace characters (not counting the newline character).

Your program can assume that the input file will be correctly formatted, with no inconsistent data. (For example, if there are only 6 states in the DFA, there will be no transition function entries that specify a state of greater than 6 or less than 1.)

You should work with a programming partner, using the pair programming software development technique.

The project repository contains test data for you. For each test DFA there is the DFA specification file (e.g. `dfa1.txt`), a file containing test strings (e.g. `str1.txt`), one string per line, and a file containing the correct answers for each test string (e.g., `correct1.txt`), one answer (Accept or Reject) per line. (The lines of `correct1.txt` correspond to the lines of `str1.txt`.) There are 10 test DFAs, in the files `dfa1.txt`, `dfa2.txt`, ..., `dfa10.txt`, and each has its own set up test inputs and outputs, in an `str` file and a `correct` file.

The project repository contains a Python program called `test_pa1.py` that tests your DFA class on all of the test DFAs in the files described above, and it reports the results. Your program must pass all tests before you submit your program for grading. If it fails any test, I'll return the program to you for further work. There is not partial credit concerning correctness of your program.

Your program your follow the programming style guidelines contained in the programmingStyleGuidelines.txt document on blackboard-¿Programming Assignments. Failure to follow these guidelines will result in points off.

To submit your program for grading, be sure that you've pushed it the repository, and post a note on campuswire, category `PA1_submit`, giving just your group number. (Questions about pa1 should go to category PA1.)