Theory of Computation: Programming Assignment 1 (DFA Simulation)

Arjun Chandrasekhar (borrowed from Dr. John Glick)

Due Sunday, 10/03/2021 at 11:59 pm (20 points)

For this assignment you will write a program that simulates the computation of a DFA on a series of input strings, and reports the results of those computations. Your program should behave exactly as described below.

The program should read input from a file that is specified as the command line argument to the program, and write output to standard output. Input consists of a description of the DFA to simulate, followed by a series of strings for which the DFA's computation should be simulated. The program should not prompt the user for interactive input.

The format of the DFA description is as follows:

1. An integer that is the number of states in the DFA. This appears by itself on the first line of input. (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 input. 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, $\text{qa} \in \{1, 2, \ldots, n\}$ and $\text{qb} \in \{1, 2, \ldots, n\}$. `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).

Following the DFA specification are the string inputs to be simulated on the DFA, one string per line. All characters on each line (except the terminating newline character) are part of the input string. The input strings should start on the line following the last line of the DFA specification. They end with the last line of the file.

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.

Your simulation program should output one line per input string, and the line should contain "Accept" if the DFA accepts the input string, and "Reject" if the DFA rejects the input string. (The quotes are not part of what you output.) No other characters should appear on a line of output, and there should be no other lines of output.

You may program in Java or Python3. I strongly suggest using Python if possible; this assignment will be much more convenient to code in Python, and it will be easier for me to help you.

You may work on this assignment alone or with a partner. On Canvas you should join a group (even if you are working alone) before you submit - here are instructions for joining a group. By joining a group, you only have to make one submission for each group. If you work with a partner, you should use the pair programming software development technique to ensure that both of you are contributing to and learning from the assignment.

Test inputs and correct outputs are in a compressed testcases folder on the assignment webpage. Your program should work correctly on all of these tests. Here is one of them (dfa1.txt):

```
6
01
1 '0' 4
1 '1' 2
2 '0' 4
```

```
2 '1' 3
3 '0' 3
3 '1' 3
4 '0' 4
4 '1' 5
5 '0' 4
5 '1' 6
6 '0' 4
6 '1' 6
1
3 6
010101010
111011110
01110011
11111
01010000
```

Correct output for the sample input (correct1.txt) is

```
Reject
Accept
Accept
Accept
Reject
```

Submit your assignment on Canvas. Your program should be contained in one source file (with the appropriate extension for the language you are using). **Do not submit more than one file, or an entire folder; all of your code should be in one source code file.** If you are coding in java, please do not include any "package ..." statements at the top of your file; my directory names are different from yours, and I will get an error when I try to compile your code (see section 4 of this page for a more detailed explanation).

Your work will be graded as follows:

- You will get 10 points for submitting a program that compiles and runs without errors, and passes at least one test.
- You will get 1 point for each test case that you pass
- If your program does not compile, does not run, does not follow the input/output directions (i.e. takes interactive input rather than a command line argument), or fails all test cases, you will receive 0 points.
- If your program follows the input/output directions correctly, and if your program passes at least one of the test cases, you will have one week to debug your program and re-submit for full points, starting from the time it is graded. If you fail to follow

directions, e.g. interactive input or multiple files, or if your program does not pass any test cases, your submission will be treated as late and you will only be eligible for half of the points.