CSCI 431 Project 4

Create software that takes in three command line parameters, as follows:

The first parameter: This is the name of an NFSA definition file

the second parameter: This is the name of a file holding strings, one string per line.

The third parameter: This is the output file.

1 NFSA definition file

The file is formatted as follows:

Line 1: This holds a string. The string represents the alphabet, so every character in the string (including a blank) is part of the alphabet except for the newline character at the end. A newline character is never part of the string.

Only printable characters with ASCII codes in the range 32-126 can be in the alphabet. In the rest of this description we use α as the the number of characters in the alphabet.

However, when building a transition table, the first column will represent λ-transitions (transitions that don't consume input).

Line 2: This line holds a single number. The number is the number of states in the machine. In the rest of this description, η is used to represent the number of states.

State 0 is always taken to be the starting state.

Line 3: This line holds one or more numbers in the range $0 - (\eta - 1)$. These are the numbers of the accepting states. Adjacent numbers are separated by a single space.

The next η lines: Each of these lines represents a row in the table for the transition function (δ). The first line represents the row for state 0, the next line the row for state 1, etc.

Each line has $\alpha + 1$ sets of states. (Recall that the first column is reserved for λ-transitions so the number of columns is the number of characters in the alphabet *plus* the λ-transitions column)

There should be no leading or ending space in the line. There should be no space between sets.

The format of a set is as follows:

1. The set delimited at the beginning with: `{` and at the end with `}`.
2. If the set is empty, then it will just have `"{}"` as the string.
3. If the set is not empty, then it will contain state numbers, separated by commas.
   For instance, `{4,12}` would be the set containing state 4 and state 12 and `{5}` would be the set containing only state 5.

2 The input file

This file has a string on each line. Empty lines are allowed. An empty line represents the empty string.

The program should take each of these strings, run it through the FSA, and indicate if the string is accepted or rejected. These results are written to the output file.

3 The output file

There should be a line in the output file for each line in the input file. The output file should first have the string being checked (from the input file) followed by a vertical bar (|), followed by + if the string is accepted and − if the string is rejected.

If a string has illegal or unknown characters, then instead of a + or − there should be a short message indicating that the string had an unknown character.

4 How it works

The program should:

1. Check for the proper number of parameters - quit w/ an error message if wrong.
2. Read in the NFSA definition and configure itself internally based on the definition. If the file is erroneous, then output an error message and quit.
3. Process lines from the input file, outputting corresponding lines to the output file as you go.

You should have a makefile for your program.

5 Teams

You may work in pairs or individually. You may pick your own partner if you work with someone else.