Homework Assignment #2
Due: September 22, 2016 at 7:00 p.m.

Submission Instructions: Each question below requires you to design a finite automaton. You must type a description of your automaton (using the file format described on page 3). Use separate files for each automaton and name the files `q1.txt` and `q2.txt`. You can submit the files either using the web portal at `https://webapp.eecs.yorku.ca/submit` or using the following unix `submit` command from one of the departmental machines

```
submit 2001C a2 q1.txt q2.txt
```

You may resubmit files up until the deadline (the new version will replace the old version). You might want to test out submitting a file prior to the deadline, just to make sure you know how to do it.

The files you submit should be plain text files. Do not submit any other format, like .doc or .rtf or .pdf. To create a text file, you can

- use your favourite text editor like vi or emacs on a unix machine (or in a Mac terminal window)
- use Notepad in Windows
- use WordPad in Windows and select the option to save the file as a text document (rather than Rich TextFormat)
- use TextEdit on a Mac and choose the Make Plain Text option from the menu (or create the file after setting the default format to plain text in Preferences).

If you have options about what kind of text file to save, use UTF-8.

Note: there is a Java programme called `DFA.java` posted on the course web page that allows you to test your solutions before you submit them. Read the comments in `DFA.java` to see how the programme works.

1. Consider the input alphabet $\Sigma = \{\texttt{0}, \texttt{1}\}$. Design a deterministic finite automaton that accepts a string if and only if it is the binary representation of a multiple of 5.

For example, the string `1111` should be accepted because it is the binary representation of fifteen, which is a multiple of 5, and `1110` should be rejected because it is the binary representation of fourteen, which is not a multiple of 5. Leading `0`'s are allowed, so `001111` should be accepted, but the empty string should be rejected.

In the comments after the description of your automaton, explain in English what each state of your automaton represents (that is, describe which strings take your automaton to that state).

2. The Leutonian alphabet consists of 4 letters: $\Sigma = \{\texttt{a, r, m, o}\}$. The characters `a` and `o` are vowels, while `r` and `m` are consonants. There are three spelling rules that all Leutonian words must follow:

1. there cannot be more than two consecutive consonants
2. there cannot be two consecutive vowels
3. a `r` can never appear later in the word than an `m`.

E.g., `roma` and `rorramma` satisfy Leutonian spelling rules, but `roora`, `mora` and `rorrma` do not.

Design a deterministic finite automaton that accepts a string if and only if it follows the three rules of Leutonian spelling.

In the comments after the description of your automaton, explain in English what each state of your automaton represents.

York University Finite Automaton File Format (YUFAFF)

The YUFAFF is a way to describe a finite automaton in a text file. It can be used for deterministic or non-deterministic automata.

Suppose you wish to describe an automaton $(Q, \Sigma, \delta, q_0, F)$. We shall assume the state names are $q_0, q_1, \ldots, q_{n-1}$, where q_0 is the starting state. (If this is not the case, just rename the states.)

The first line of the file contains four positive integers n, m, k and t separated by single spaces, where $n = |Q|, m = |\Sigma|, k = |F|$ and t is the number of lines in the file that will be used to describe transitions of the automaton.

The second line of the file contains m strings $\sigma_0, \sigma_1, \ldots, \sigma_{m-1}$ separated by single spaces. These strings are the names of the characters of the input alphabet. These strings should be distinct and should not contain any spaces. Also, the string `empty` is not allowed as a character name. (Ordinarily, you will use single-character strings to name the characters, but we allow strings more generally in case you want to talk about an automaton that uses an alphabet of non-ASCII input characters. For example, if your input alphabet was $\Sigma = \{\alpha, \beta, \gamma\}$, you might use the names `alpha beta gamma` for the characters.)

The third line contains k natural numbers separated by single spaces. These numbers are the indices of the accepting states. For example, if the accepting states are q_0, q_6 and q_9 then the third line of the file should contain `0 6 9`.

The following t lines each contain the description of one transition defined by δ.

For *deterministic* finite automata, each of these t lines contains a natural number i, a character name x and another natural number j, separated by single spaces. There should be one line containing $i\ x\ j$ if and only if $\delta(q_i, x) = q_j$. The values should satisfy $0 \le i < n$, $0 \le j < n$, $x \in \{\sigma_0, \ldots, \sigma_{m-1}\}$. (For deterministic finite automata, we should have $t = mn$.)

For *non-deterministic* finite automata, each of the t lines contains a natural number i, a string x and another natural number j, separated by single spaces. The string x should either be a character name in $\{\sigma_0, \ldots, \sigma_{m-1}\}$ or should be the string `empty` to indicate a ε-transition. There should be one line containing $i\ x\ j$ if and only if $q_j \in \delta(q_i, x)$.

Anything after these t lines will be ignored, so you can use them to write comments or explanations.

Example: The deterministic automaton shown in Figure 1.14 of the textbook could be described as follows in YUFAFF:

```
3 4 1 12
<RESET> 0 1 2
0
0 <RESET> 0
0 0 0
0 1 1
0 2 2
1 <RESET> 0
1 0 1
1 1 2
1 2 0
2 <RESET> 0
2 0 2
2 1 0
2 2 1
Comment: This file describes the automaton in Figure 1.14 of the textbook
```