# Assignment 1
## Deterministic Finite Automaton

Write a program that accepts as input a description of a deterministic finite automaton (DFA) over the alphabet $\Sigma = \{a, b\}$ followed by a sequence of one or more words. Your program should simulate the DFA and report whether each word was accepted or rejected by the DFA.

# 1 Input

Your program will read its input from `stdin`.

The first line will have two integers, *nstates* and *naccepting*, where *nstates* is the number of states in the DFA and *naccepting* is the number of accepting states in the DFA. States are labeled with integers and are numbered sequentially starting with zero. You may assume that $0 < nstates \leq 100$ and $0 \leq naccepting \leq nstates$. State 0 will always be the start state.

There will then be *naccepting* line(s) each with a single integer that identifies one of the accepting states. Note that *naccepting* might be zero, in which case the input file will proceed from the first line with two integers described above directly to the lines described in the next paragraph.

There will then be *nstates* line(s) each with two integers. The first line corresponds to the DFA's $\delta(0, a)$ and $\delta(0, b)$, the second line to $\delta(1, a)$ and $\delta(1, b)$, and so on.

Following that there will be a line with a single integer, $nwords > 0$.

That will be followed by *nwords* line(s) where each line will have a single word $w \in \Sigma^*$ where, again, $\Sigma = \{a, b\}$. You may assume that $w \neq \Lambda$.

Here is an example input file:

```
2 1
0
0 1
1 1
3
a
b
ab
```

It describes a DFA with (assumed) $\Sigma = \{a, b\}$, $Q = \{0, 1\}$, (assumed) $q_0 = 0$, $T = \{0\}$, and $\delta : Qx\Sigma \rightarrow Q$ as follows:

| $\delta$ | a | b |
|---|---|---|
| 0 | 0 | 1 |
| 1 | 1 | 1 |

and concludes with the three words each $\in \Sigma^*$; $a, b$ and $ab$.

# 2 Output

The output first contains the DFA's transition table $\delta$ annotating each of the accepting states with an astericks "*".

Immediately thereafter there must be one line for each of the words $w \in \Sigma^*$ from the input file. Each line should print the word $w$ followed by either the word `accepted` or `rejected` to indicate whether $w$ was accepted or rejected by the DFA.

Using the example input above, the output should look exactly like this:

```
    |  a  b
----+------
* 0 |  0  1
  1 |  1  1
a accepted
b rejected
ab rejected
```

Note the "*" just before state 0 to indicate that state 0 is an accepting state.

# 3 Files I Give You

In `/home/turing/karonis/Classes/ForStudents/Handout/21fall/DFA` on `turing.cs.niu.edu` you will find a small collection input files (e.g., `dfa-in.1`), an executable solution to this assignment, `dfa.key`, to help you develop and test your program, and a submission script, `submit_dfa`, described below.

To be eligible to receive full credit your program must execute on `turing.cs.niu.edu` and produce output that results in an empty `diff` when compared to the output generated by the answer key `dfa.key` using any of the input files. You can test that your program produces output that is indentical to the answer key's output (i.e., empty `diff`) like this on `turing` (assumes your program executbale is named `dfa`).

```
% dfa.key < dfa-in.1 > dfa-in.1.key
% dfa < dfa-in.1 > dfa-in.1.out
% diff -bitw dfa-in.1.key dfa-in.1.out
%
```

# 4 File You Must Write

You will submit your source code file(s) and a `makefile` that will allow (a) `make clean` which will remove all binary files and (b) `make dfa` that will build your program and produce an executable file that must have the filename `dfa`. If you write your program in Python, then the name of the file you submit must be `dfa.py`.

Use `submit_dfa` to submit each file one at a time. To submit your `makefile`, for example, simply type `submit_dfa makefile`.
Please *do not* submit binary files (e.g., *.o or an executbale).