Assignment 2
Nondeterministic Finite Automaton

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

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 NFA and *naccepting* is the number of accepting states in the NFA. 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), one for each state in the NFA.

Recall that an NFA state can have zero or more outgoing edges for each symbol in the alphabet Σ. Accordingly, the δ function for NFA is defined as $\delta : QX\Sigma \rightarrow P(Q)$. Unlike the δ functions for deterministic finite automata (DFA) that always return a single state, δ functions for NFA return a *set of states*, including possibly the empty set.

The line for each state will start with the number of number of outgoing transitions, *ntransitions*, for the first symbol in the alphabet, a. That will be immediately followed by *ntransitions* integers each identifying the state transition to when reading a from the NFA input tape. Immediately following those states (i.e., on the same line in the input file) you will read an integer, again *ntransitions*, which represents the number of outgoing transitions for second symbol in the alphabet, b. That will be immediately followed *ntransitions* integers each identifying the state transition to when reading b from the NFA input tape.

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:

```
5 2
2
4
2 0 1 2 0 3
1 2 0
1 2 1 2
0 1 4
1 4 1 4
5
a
ab
aba
abaa
abaab
```

It describes a NFA with (assumed) $\Sigma = \{a, b\}$, $Q = \{0, 1, 2, 3, 4\}$, (assumed) $q_0 = 0$, $T = \{2, 4\}$, and $\delta : Qx\Sigma \rightarrow P(Q)$ as follows:

δ	a	b
0	{0,1}	{0,3}
1	{2}	{}
2	{2}	{2}
3	{}	{4}
4	{4}	{4}

and concludes with the fives words each $\in \Sigma^*$; a, ab, aba, $abaa$, and $abaab$.

2 Output

The output first contains the NFA's transition table δ annotating each of the accepting states with an asterisks "*".

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 NFA.

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

```
    | a b
----+----
  0 | { 0 1} { 0 3}
  1 | { 2} {}
* 2 | { 2} { 2}
  3 | {} { 4}
* 4 | { 4} { 4}
a rejected
ab rejected
aba rejected
abaa accepted
abaab accepted
```

Note the "*" just before states 2 and 4 to indicate that those are accepting states.

3 Files I Give You

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

To be eligible to receive full credit your program must execute on execute on `turing.cs.niu.edu` and produce output that results in an empty `diff` when tested `turing` using any of the input files in the following manner (assumes your program executable is named `nfa`).

```
% nfa.key < nfa-in.1 > nfa-in.1.key
% nfa < nfa-in.1 > nfa-in.1.out
% diff -bitw nfa-in.1.key nfa-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 nfa` that will build your program and produce an executable file that must have the filename `nfa`. If you write your program in Python, then the name of the file you submit must be `nfa.py`.

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