Assignment 1 - Implement an NFA

Implement a nondeterministic finite automata that reads an input file which describes the NFA and outputs if it ends in an accept or reject state.

1. File Input
Read the filename which describes the input NFA as a command line argument to the nfa program. Executing the program would look like this:

```
./nfa example1
```

where example1 is the input file name.

The input file has the following format.

```
<alphabet-size> <alphabet-list>
<number-of-states> <state-list>
<starting-state>
<accept-state>
<length-input-string> <input-string>
<number-of-transitions>
<transition-1>
<transition-2>
  ...
<transition-n>
```

Example 1: A sample file for a simple NFA:

```
2   0 1            -two strings in alphabet, the strings are 0 and 1
2   q1 q2          -two states, the states are q1 and q2
q1                 -the starting state is q1
q2                 -the accept state is q2
3 0 1 0            -there are three input elements, the strings 0, 1, and 0
4                  -number of transitions (arrows) in the NFA is 4
q1 0 q1            -transition 1, if state q1 is active and input 0, new state is q1
q1 1 q2            -transition 2, if state q1 is active and input 1, new state is q2
q2 0 q1            -transition 3, if state q2 is active and input 0, new state is q1
q2 1 q2            -transition 4, if state q2 is active and input 1, new state is q2
```

Example 2: A second sample input file:

```
2    0 1             -two strings in the alphabet, the strings are 0 and 1
3    q1 q2 q3        -three states, the states are q1, q2, and q3
q1                   -starting state is q1
```

q3 -accept state is q3
2 0 1 -there are two input elements, strings 0 and 1
3 -number of transitions (arrows) is 3
q1 1 q2 -transition 1, if state q1 is active and input 1, new state is q2
q2 e q3 -transition 2, an epsilon transition leading from q2 to q3
q1 0 q1 -transition 3, if state q1 is active and input 0, new state is q1

There can be any number of whitespace characters between the text in the file.
Input strings can be longer than one character.

2. Implementation
There will not be more than 100 strings in the alphabet. Each string could be up to 10 characters long. The state names will not be longer than 10 characters. The input sequence will not be longer than 100 strings.
The code will be written in C.

3. Output
The program will step through each input string and update the list of active states. The input value and the list of states will be printed.
Active states are displayed as 1 and inactive states as 0.
At the end of the input the program prints if the NFA is in an accept or reject state.

For example 1 the output would look like this:
0 1 0
1 0 1
0 1 0
reject

The above output means:

0 1 0 -the input 0 with the starting state q1 leaves state q1 active and q2 not active
1 0 1 -the input 1 with the above states leads to state q1 being inactive and q2 being active
0 1 0 -the input 0 with the above state leads to state q1 being active and q2 being inactive
reject -after the last input string, only q1 is active, the accept state is q2 is not active so print reject

For example 2 the output would look like this:
0 1 0 0
1 0 1 1
accept

The above output means:
0 1 0 0 -the input 0 with the starting state q1 leaves q1 active
1 0 1 1 -input 1 with the above state leads to state q2 being active, when q2 activates it follows the epsilon transition and activates q3
accept -at the end of the input the accept state q3 is active so print accept