```
i 3
q 1    # prints "found: root"
q 2    # prints "found: r"
q 3    # prints "found: r r"
q 4    # prints "not found"
```

3. **rpn.py:** Write an interpreter for a reverse Polish notation (RPN) calculator. Each line of the input corresponds to a single operand or operator. Operands should be pushed onto a stack, and popped as needed when an operator is encountered (and the result pushed back on the stack). At each step, you should print the top-most element on the stack. If an operator requires more elements than are available on the stack, you should print "invalid operation" and ignore the operator. Operands will be non-negative integers. You should support these operators:

   | | |
   |---|---|
   | `+` | Add two numbers |
   | `-` | Subtract two numbers |
   | `*` | Multiply two numbers |
   | `/` | Divide two numbers |
   | `~` | Negate one number |

   Example:

   ```
   23   # prints 23
   5    # prints 5
   +    # pops 23 and 5, pushes and prints 28
   ```

   Another example:

   ```
   1    # prints 1
   2    # prints 2
   +    # pops 1 and 2, pushes and prints 3
   5    # prints 5
   1    # prints 1
   -    # pops 5 and 1, pushes and prints 4
   +    # pops 3 and 4, pushes and prints 7
   ```

4. **dfa.py:** Write a program that reads a description of a deterministic finite automaton (DFA) and then classifies input strings as accepted or rejected by the DFA.

   DFAs are characterized by the following 5-tuple: $(Q, \Sigma, \delta, q_0, F)$, where $Q$ denotes the set of states, $\Sigma$ is the alphabet of possible input symbols, $\delta$ is the set of transition rules, $q_0$ is the start state, and $F$ is the set of final (accepting) states.