CSE 231 Fall 2017

Programming Project #4

Assignment Overview

In this assignment you will practice functions, strings and control.

This assignment is worth 40 points (4.0% of the course grade) and must be completed and turned in **before 11:59 on Monday, October 09**.

Background

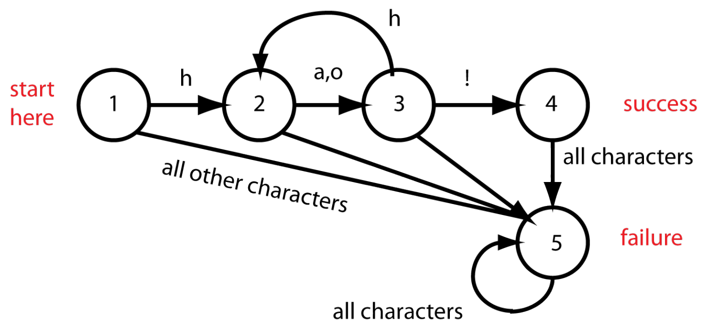
Finite State Machine or FSM is also called Finite State Automata. A Finite State Machine is a popular tool that is widely used in computer science. Why it is so popular? Because it is simple, and easy to understand and implement. Here is a brief introduction:
http://cs.union.edu/~striegnk/courses/nlp-with-prolog/html/node2.html

Your task is to implement an FSM that can recognize simple patterns of strings such as:

ha!
haha!
hahaha!
ho!
hoho!
hohoho!
hahohoho!
hahohahoha!

Basically, the pattern to be recognized is a sequence of "ha" or "ho", or a mixture of both. The sequence can be of any length (at least greater than three, i.e., the shortest sequence is "ha!" or "ho!"). There is no restriction on how many "ha"s or "ho"s the sequence can contain, and the "ha"s and "ho"s can appear in any order if the sequence is a mixture of both. The string must end with one "!".

The FSM that recognizes those exact string patterns can be drawn as follows:



This FSM has 5 *states* and 9 *transitions*. State 1 is the start state and state 4 is the successful end state while state 5 is the unsuccessful end state. The transitions indicate how one transits from one state to another. For example, if we are at state 1 (start state) and observe an "h", then we transit to state 2, but if we observe any other character we transition to state 5 (failure). Notice that if we land in state 5, any other character will transition us back into state 5, i.e. once in state 5 we never leave.

When using this FSM to recognize a string pattern (i.e., whether or not an input string conforms to the pattern described by the FSM), we scan the input string from left to right, and make transitions based on the observed character and the state of the FSM.

Consider some examples:

- Input “ha!”: starting in state 1 the “h” transitions to state 2, the “a” then transitions to state 3 and the “!” transitions to state 4. Since that was the last character we finish in state 4 which indicates success.
- Input “hax”: starting in state 1 the “h” transitions to state 2, the “a” to state 3, but the “x” transitions to state 5. Since that was the last character we finish in state 5 which indicates failure.
- Input “hahoha!”: starting in state 1 the “h” transitions to state 2, the first “a” to state 3, the next “h” sends us back to state 2, the “o” to state 3, the next “h” back again to state 2, the next “a” to state 3, and finally the “!” to state 4. Since that is the last character we finish in state 4 which is success.
- Input “y”: starting in state 1 the “y” transitions to state 5 which is failure.
- Input “yha!”: starting in state 1 the “y” transitions to state 5 and all subsequent characters, i.e. “ha!” in this case, transition us back to state 5 which is failure.
- Input “ha!x”: starting in state 1 the “h” transitions to state 2, the “a” to state 3, the “!” to state 4, but there is one more character, “x”, that transitions to state 5 which is failure.

Requirements:

1. Implement the above FSM to recognize whether a given string conforms to the specific pattern or not.
2. You have to use the 3 functions in the provided `proj04.py` skeleton. You have to implement and use them as specified:
    a. `get_ch()`: Does not have any parameters. Prompts for a character and returns it if it’s an empty string or a single character. Prints an error message and re-prompts if the input length is more than 1.
    b. `find_state(state, ch)`: Takes two parameters: current state and input character. Finds the next state and returns it.
    c. `main()`: It’s the main driver of the program. It doesn’t have any parameters. Calls the other two functions in a loop and keeps track of the entered character and the state. Prints the results when the returned value of `get_char` is empty string.
3. Input will come from the user one character at a time (in the `get_ch()` function). If the input has two or more characters, print an error message and prompt for another input.
4. Pressing only the Return key (Enter key) means the user input is finished.
5. Echo the string, then determine if it represents laughter or not, and print that. See the sample output below.

Deliverables

The deliverable for this assignment is the following file:

`proj04.py` -- your source code solution

Be sure to use the specified file name and to submit it for grading via Mimir before the project deadline.

Getting Started

- *Solve the problem using pencil and paper first.* You cannot write a program until you have figured out how to solve the problem. This first step can be done collaboratively with another student. However, once the discussion turns to Python specifics and the subsequent writing of Python, you must work on your own.

- Use Anaconda Spyder to open the provided file (`proj04.py`).

- Write a simple version of the program, e.g. implement the `get_ch` function and test it.

- Use the **Mimir** system to turn in the first version of your program. All tests will fail initially because we aren't far enough to pass the first test.

- Next implement `main` and call the `get_ch` function to get characters repeatedly and quit if user has entered empty string.

- Cycle through the steps to incrementally develop your program:

    - Edit your program to add new capabilities.
    - Run the program and fix any errors.

- Use the **Mimir** system to submit your final version.

Hints:

States and their transitions can be handled as follows:

a) Use a variable named state to keep track of what state you are in.
b) Since you always start in state one, begin with state = 1
c) You can handle each transition with a series of if and elif statements. Here is pseudo code to represent that:

```
if you are in state X:
    if character == 'z' or character == 'w':  # transition
        state = A   # next state (you could return A here if you wish)
    else:
        state = B   # next state (you could return B here if you wish)
```

Example Interaction

Test 1:

```
I can recognize if you are laughing or not.
Please enter one character at a time.

Enter a character or press the Return key to finish: h
Enter a character or press the Return key to finish: a
Enter a character or press the Return key to finish: !
Enter a character or press the Return key to finish:

You entered ha!
You are laughing.
```

Test 2:

```
I can recognize if you are laughing or not.
Please enter one character at a time.

Enter a character or press the Return key to finish: h
Enter a character or press the Return key to finish: a
Enter a character or press the Return key to finish: h
Enter a character or press the Return key to finish: o
Enter a character or press the Return key to finish: h
Enter a character or press the Return key to finish: a
Enter a character or press the Return key to finish: !
Enter a character or press the Return key to finish:

You entered hahoha!
You are laughing.
```

Test 3:

```
I can recognize if you are laughing or not.
Please enter one character at a time.

Enter a character or press the Return key to finish: ha!
Invalid input, please try again.

Enter a character or press the Return key to finish: a

Enter a character or press the Return key to finish:

You entered a
You are not laughing.
```

Test 4:

```
I can recognize if you are laughing or not.
Please enter one character at a time.

Enter a character or press the Return key to finish: h
Enter a character or press the Return key to finish: a
Enter a character or press the Return key to finish: !
Enter a character or press the Return key to finish: a
Enter a character or press the Return key to finish:

You entered ha!a
You are not laughing.
```

Test 5:

```
I can recognize if you are laughing or not.
Please enter one character at a time.

Enter a character or press the Return key to finish: h
Enter a character or press the Return key to finish: a
Enter a character or press the Return key to finish:

You entered ha
You are not laughing.
```

Test 6:

```
I can recognize if you are laughing or not.
Please enter one character at a time.

Enter a character or press the Return key to finish: h
Enter a character or press the Return key to finish: a
Enter a character or press the Return key to finish: h
Enter a character or press the Return key to finish: e
Enter a character or press the Return key to finish: e
Enter a character or press the Return key to finish: !
Enter a character or press the Return key to finish:

You entered hahee!
```

You are not laughing.

Scoring Rubric

```
Computer Project #4 Scoring Summary

General Requirements

  __0__   (5 pts) Coding standard

Program Implementation

  __0__   (7 pts) Test case 1

  __0__   (8 pts) Test case 2

  __0__   (5 pts) Test case 3

  __0__   (5 pts) Test case 4

  __0__   (5 pts) Test case 5

  __0__   (5 pts) Test case 6
```