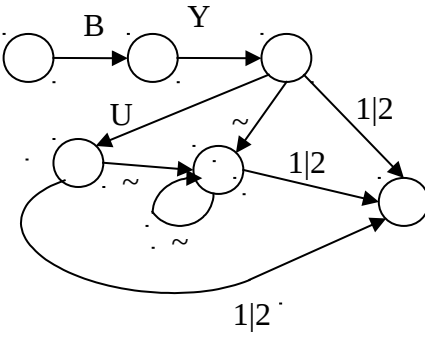


Questions:	Answers:
<p>1. Consider the following finite state machine (FSM).</p>  <p>a) Does the FSM recognize: “BYU~1” (i.e., does it recognize the sequence of the five characters within the quotes)?</p> <p>b) Does the FSM recognize: “BY~1”?</p> <p>c) Does the FSM recognize: “BYU1”?</p> <p>d) Give all shortest strings recognized by the FSM.</p> <p>e) Give all strings with exactly two tildes (~) recognized by the FSM.</p> <p>f) How many strings does the FSM recognize?</p>	<p>a) Yes</p> <p>b) Yes</p> <p>c) Yes</p> <p>d) Assuming the FSM MUST end on the final node:</p> <ul style="list-style-type: none"> <li>- BY1</li> <li>- BY2</li> </ul> <p>e)</p> <ul style="list-style-type: none"> <li>- BY~~1</li> <li>- BY~~2</li> <li>- BYU~~1</li> <li>- BYU~~2</li> </ul> <p>f) Because a node can loop back adding a '~' infinitely, there is an infinite number of recognizable strings.</p>
<p>2. Give a regular expression that recognizes the same set of strings as the FSM in Question 1.</p>	<p><math display="block">^BY((1 2) (\sim+(1 2)) (U(\sim+(1 2)) (1 2)) (1 2)))\\$</math></p>

3. Draw a finite state machine (FSM) for a string representing a single time of day. Your FSM should accept times such as:

12:36 pm    1:59 am    4:00 pm    2:45 am

Note that there are no leading zeros. Also, am and pm are lowercase and have no periods. A terminal error state is not necessary.

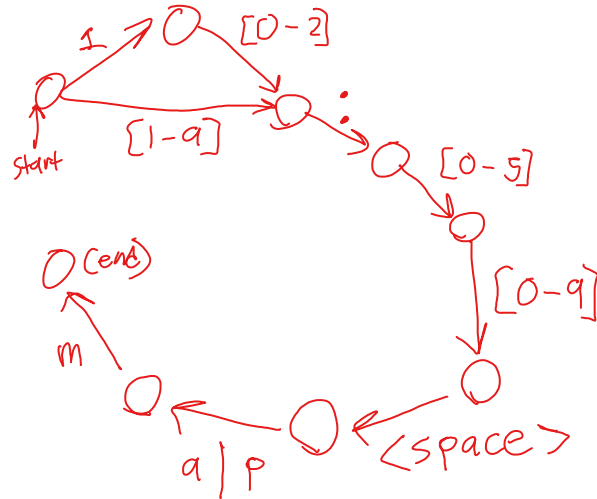
To make the task easier, you can make up any grouping you want. For example, you may wish to let <digit> represent any digit, let <1> represent the digit 1, let <012> represent any digit 0 through 2, and let <space> represent a space.

Time statements are formatted in many ways, but we want your FSM to only accept the above time format—not military time or any other time format.

Square brackets [ ] represent a numerical range, i.e. [0-9] could be any number between 0 and 9 (0, 1, 2, 3, 4, 5, 6, 7, 8, 9), or [0-2] could be between 0 and 2 (0, 1, 2).

Vertical bar | represents an OR, i.e. 'a|p' could be 'a' or 'p'.

<space> represents a space character.



4. Give a regular expression for the FSM described in Question 3.

$^(1[0-2]|[1-9]):[0-5][0-9]\s(a|p)m\$$

5. Using the regular-expression tester at <http://regexpal.com>, copy the four time statements in Question 3 above into the space marked “Enter test data here.” Then enter your regular expression for Question 4 into the space marked “Enter your regular expression here.” (If necessary debug your regular expression so that it works correctly—correctly highlights all four time statements.)

Type in five more “good” test cases---four of which should fail to satisfy your regular expression.

As your answer for this question, provide a screenshot of your work or, if this is inconvenient, write your five “good” test cases as your answer and circle the one (the only one) that satisfies your regular expression in Question 4.

1:21 pm (correct)

1:61 pm (char after : is above 5)

1:21pm (no space after last digit)

01:21 pm (leading 0)

6. Copy the following child list into the text area of the regular-expression tester at <http://gskinner.com/RegExr/>. (This child list comes from an OCR'd page in *The Ely Ancestry*. Note, for example the OCR error in the year i860.)

1. Charles Halstead, b. 1857, d. 1861.
2. William Gerard, b. 1858, d. 1861.
3. Theodore Andruss, b. i860.
4. Emma Goble, b. 1862.

a) Write a single regular expression that recognizes each and all of the records. (You may use any of the advanced regular-expression patterns listed in the RegExr quick reference guide. Example: `[A-Za-z]{5,8}` matches any string of letters whose length is between 5 and 8 characters long.)

b) If not already in your regular expression for (a), add parentheses around the subexpression for the name, the birth date, and the death date. Give the new expression if it is different from (a). (Every parenthesized expression denotes a capture group. When used in programming, capture groups return recognized substrings when the regular expression matches a text string, and thus a programmer can obtain desired substrings. The RegExr site lists capture groups below the text area.)

c) List the capture groups (including the group number) for the name, birth date, and death date.

a)

```
^d\.[A-Z,a-z]+\s[A-Z,a-z]+,\sb.\s[a-z,0-9]{4}
(\.|\,sd\.[0-9]{4}\.)$
```

b)

```
^d\.[A-Z,a-z]+\s[A-Z,a-z]+,\sb.\s[a-z,0-9]{4}
(\.|\,sd\.[0-9]{4}\.))$
```

c)

Capture Group #1 (Name)  
`([A-Z,a-z]+\s[A-Z,a-z]+)`

Capture Group #2 (Birth Date)  
`(b.\s[a-z,0-9]{4})`

Capture Group #4 (Death Date)  
`(,\sd\.[0-9]{4})`

(there is a 3rd capture group, which includes an OR to handle lines ending with '.' after the birth year, OR continuing with ',' to include a death year.)

7. Draw an FSM for comments as defined in Project 1. (You may use <EOF> for the end-of-file character and <EOL> for the end-of-line character.)

Note: This question should illustrate that the specification in Project 1 is incomplete, as several cases are not specified. In this situation the programmer either has to make reasonable assumptions or ask for more direction. For this question, make reasonable assumptions (all reasonable solutions are acceptable).