COMP 722 E-commerce Fall 2006 Programming Assignment 2

Due in the Digital Drop Box on Wednesday, Sept. 13 at 11:00 PM

1. Write an HTML document that uses JavaScript to input two strings. (Use the `prompt` method of the `window` object to raise prompt dialog boxes.) It then counts the number of times the second string occurs in the first string. It outputs this count. (Use the `alert` method of the window object to raise an alert box that displays the count.) The matching should be case-insensitive. (You can use the string instance method `toLowerCase` to get lowercase versions of both strings.) For example, if the first string is

Inside incapable bins singly

and the second string is

in

then the program should output 4.

One way to approach this problem is to use the string instance method `indexOf` with two arguments. Where the value of variable `s` is a string, *str* is a string, and *n* is a non-negative integer,

`s.indexOf(` str, *n* `)`

searches `s` from left to right for *str*. It starts the search at position *n* in `s` and returns the index of the first occurrence of *str* it finds. It returns –1 if it doesn't find an occurrence of *str*. For example, if `s` is the string "inside incapable bins singly", then

`s.indexOf( "in", 0 )` returns 0 (the first "in" begins at position 0),
`s.indexOf( "in", 1 )` returns 7,
`s.indexOf( "in", 8 )` returns 18,
`s.indexOf( "in", 19 )` returns 23, and
`s.indexOf( "in", 24 )` returns -1.

So the idea is to initialize a variable to the value returned by `first.indexOf(second,0)`, where `first` is the first string read in and `second` is the second. In a loop, this variable is incremented, used as the second argument in a similar call of `indexOf`, and updated to the value returned by this call. When –1 is returned, the loop exits. The count is the number of times `indexOf` was invoked and didn't return –1.

2. The following code (with fragments missing) prompts the user for a time and date in the format

hh:mm mm/dd/yyyy

representing two-digit representations of the hour ("hh"), the minutes (the first "mm"), the month (the second "mm"), and the day ("dd"), followed by the four-digit year ("yyyy"). The time is military time (hours range from 0 to 23). Note that there is one or more white-space characters between the time and the date. Valid years range from 1900 to 2006. Some examples of valid inputs are

01:00 01/03/2001
23:59 12/31/1901

Some examples of invalid inputs are

1:30 10/03/1999 – The hour has only one digit.
01:30 2/12/1984 – The month has only one digit.
24:45 04/23/2002 – The hour is greater than 23.

The code attempts to match the string input from the prompt against a regular expression that defines the legal format for the input. If the match fails, an illegal-format message is sent to the browser and the program terminates. If the match succeeds, the program checks that all five values are within the valid ranges. This requires that, in the regular expression, the subpatterns corresponding to the five values be enclosed within parentheses so that the implicit variables `RegExp.$1` to `RegExp.$5` may be defined after a successful match. The valid range for hours is 0-23, for minutes is 0-59, for months is 1-12, for days is 1-31, and for years is 1900-2006. If one or more of the values is out of range, a message to this effect is sent to the browser. If all the values are valid, then the last thing the program does is send a message of the following form to the browser:

⟨hh⟩ hours, ⟨mm⟩ minutes in day ⟨dd⟩ of the ⟨mm⟩th month of ⟨yyyy⟩

Here the place-holders of the form ⟨__⟩ represent the values from the input string; they are available at this point in the program in the variables `RegExp.$1` to `RegExp.$5`.

For this problem, we require that the input string not contain any characters other than those that are matched by the regular expression – the regular expression must match the entire input string. So the regular expression must begin with an anchor (that forces it to match the beginning of the string) and end with an anchor (that forces it to match the end of the string).

Each missing fragment is labeled with a Greek letter. Each Greek letter is repeated after the code with a description of the missing code at the indicated position. You can download this file (with the fragments missing) from the link just below where you got the assignment..

```
<html>
<title>Assignment 2, Problem 2</title>
<script type = "text/javascript">
  var  str,
       reg = _α_____________________________________________,
       wrong;

  str = prompt( "Enter the time and date in the the format\n" +
                "hh:mm mm/dd/yyy",
                "00:00 01/01/2000" );
  if ( str.match( reg ) )  {
    if ( _β_______ < 0 || 23 < _β_______ )  {
      alert( "The hour is out of range." );
      wrong = true;
    }

    if ( _γ_____________________________ )  {
      alert( "The minute is out of range." );
      wrong = true;
    }
```

Continued

Continued from previous page

```
    if ( _δ_____________________________ )  {
      alert( "The month is out of range." );
      wrong = true;
    }

    if ( _ε_____________________________ )  {
      alert( "The day is out of range." );
      wrong = true;
    }

    if ( _ζ_____________________________ )  {
      alert( "The year is out of range." );
      wrong = true;
    }

    if ( wrong )
      document.writeln( "The format is correct but one " +
                        "or more values is out of range." );
    else
      document.writeln( _η_____________________________
                        _____________________________
                        _____________________________________
                        _________________ );
  }
  else
    alert( "The format is illegal." );
</script>
</head>
</html>
```

α: The required regular expression

β (occurs twice): The implicit variable containing what was matched by the first parenthesized expression in the regular expression. The condition in which this occurs is true if the hour is out of range.

γ: The condition that is true if the minute is out of range

δ: The condition that is true if the month is out of range

ε: The condition that is true if the day is out of range

ζ: The condition that is true if the year is out of range

η: The string that is output (echoing the five values extracted from the input string) if the input string is valid.