

## COMP 322 Internet Systems Fall 2008 Exam 2—Solutions

50 points total

1 (16 pts.). The listing (with gaps) on the next page shows function `getCodes()`, with a loop that repeatedly prompts for strings with the following pattern (whose components occur in the order listed)

- Two uppercase letters
- A dash (-)
- Two or three digits
- One or more asterisks (\*)

Examples of valid strings are

MP-74\*

AA-648\*\*

The string without the trailing \*'s is considered a code. If the string ends with two or more \*'s, the code in the string is considered a special code; otherwise (i.e., the string ends with just one \*), the code is just an ordinary code. For example, the above strings represent an ordinary code, MP-74, and a special code, AA-648. The regular expression, `reg`, representing this pattern must cause the code to be remembered and the string of \*'s to be remembered. Also, for a match, the entire string must match the regular expression. (Hint: use anchors.)

The loop prompts for a code. To exit the loop, the user clicks the **Cancel** button (or strikes the **ESC** key). If the user does not exit, the string entered is assigned to the variable `str`. If `str` matches `reg`, then we check whether it's a special or ordinary code. If the string has more than one \*, then the code is special, and we add it to the end of the array `specCodes`. Otherwise, we add it to the end of array `codes`. (Recall that the string attribute `length` is the length of the string.) If `str` does not match `reg`, we raise an alert box telling the user the string is in the wrong format.

At the end, the function displays the contents of arrays `codes` and `specCodes` in alert boxes.

The gaps in the following listing are labeled with Greek letters. These letters are repeated after the listing with descriptions of what go in the corresponding gaps. You there supply the missing code.

```

function getCodes()
{
  var codes = [],
      specCodes = [],
      reg = α _____,
      str;
  while ( str = window.prompt( "Enter a code",
                               "XX-123*" ) )
  {
    if ( str.match( reg ) )
      β _____
      _____
      _____
    else
      window.alert( "Wrong format" );
    window.alert( "The ordinary codes:\n" + codes );
    window.alert( "The special codes:\n" + specCodes );
  }
}

```

$\alpha$  (9 pts.): The regular expression as described above

**Answer**

```

/^[A-Z]{2}-\d{2,3})(\*+)/

```

$\beta$  (7 pts., multiple lines in the listing): If the matched string ends with more than one \*, add the code to the end of array specCodes; otherwise, add the code to the end of array codes.

**Answer**

```

if ( RegExp.$2.length > 1 )
  specCodes.push( RegExp.$1 );
else
  codes.push( RegExp.$1 )

```

2 (15 pts.) The HTML document listed below allows the user to click on the paragraph with id “pClick” to add the name of a friend as the content of a li element that is the child of the ul element with id “names”. The list of friends is initially empty. The screen shot at right shows the case after the paragraph has been clicked twice.



```
<html>
<head>
  <title>Exam 2, Problem 2</title>
  <script type="text/javascript" src="prob2.js"</script>
</head>
<body onload="start()">
  <h4>Some Friends</h4>
  <ul id="names"></ul>
  <p id="pClick">Click here to add a name.</p>
</body>
</html>
```

The listing (with gaps) on the next page is of file `prob1.js`, referenced in the script element of the above HTML. It defines two functions. Function `start()` registers `addName()` as a listener for `click` events in the paragraph with id “pClick”. Function `addName()` prompts for a name and assigns the returned string to variable `name`, assigns to `ls` a reference to the element (a `ul`) with id “names”, and creates a new `li` element and assigns its reference to `newName`. It then makes the value of `name` the content of the `li` element (`newName`) and adds that element at the end of the children of the `ul` (`ls`).

The gaps in the following listing are labeled with Greek letters. These letters are repeated after the listing with descriptions of what go in the corresponding gaps. You there supply the missing code.

```
function start()
{
    α
}

function addName()
{
    var name = window.prompt( "New name", "John" ),
        ls = β,
        newName = γ;

    newName.innerHTML = name;
    δ;
}
```

α (4 pts., multiple lines in the listing): Supply the body of function `start()`, which makes function `addName()` a listener for `click` events in the paragraph with id “pClick”.

**Answer**

```
var pc = document.getElementById( "pClick" );
pc.addEventListener( "click", addName, false );
```

β (3 pts.): Get a reference to the `ul` element with id “names”.

**Answer**

```
document.getElementById( "names" )
```

γ (4 pts.): Create a new `li` element, returning a reference to it.

**Answer**

```
document.createElement( "li" )
```

δ (4 pts.): Add the new `li` element at the end of the children of the `ul` element.

**Answer**

```
ls.appendChild( newName )
```

3 (19 pts.). The HTML document listed below (with gaps) initially is rendered as shown at right. The form element has id “frm”. The field set contains two radio buttons, both with name “gender”. The value of each is the same as the text displayed with it (i.e., the first button has value “Males”, and the second has value “Female”). The name of the textbox is “honorific”. The user clicks a radio button then the submit button.

File prob3.js, referenced in the script element of the HTML document, is listed (with gaps) on the next page. It contains two function definitions. Function start() registers function formSubmit() as a listener for submit events in the form. Function formSubmit() finds the value of the checked button. If this is “Male”, it assigns “Mr.” as the value of the textbox; otherwise, it assigns “Ms.” as its value. The screenshot at right show the case where the Male button was checked before Submit was clicked.

Function formSubmit() works as follows. It assigns to variable fm a reference to the form then assigns to genders the array of objects representing the radio buttons (with name “gender”). (Each of these objects has a checked and a value attribute.) It then turns off the default behavior, the submission. (The function is passed an Event instance, event. We assume the document is rendered by Mozilla Firefox.) Next, it loops over the elements in genders and sets variable gend to the value (“Male” or “Female”) of the button that was checked. Finally, it sets the value of the textbox according to whether or not gend is “Male”.

```
<html>
<head>
  <title>Exam 2, Problem 3</html>
  <script type="text/javascript" src="prob3.js"></script>
</head>
<body onload="start()">
  <form id="frm">
```

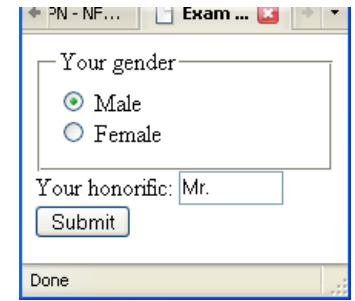
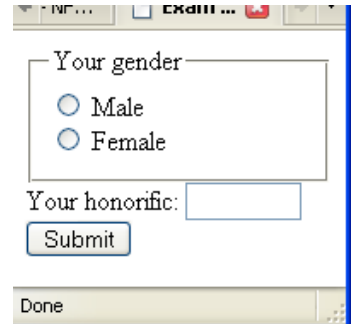
α

```
    Your honorific: <input type="text" size="7" name="honorific"></br>
    <input type="submit" value="Submit">
  </form>
</body>
</html>
```

α (5 pts., multiple lines in the listing): Write the missing fieldset element. The two radio buttons have name “gender”. The one has value “Male”, the other value “Female”.

**Answer**

```
<fieldset>
  <legend>Your gender</legend>
  <input type="radio" name="gender" value="Male"> Male</br>
  <input type="radio" name="gender" value="Female"> Female
</fieldset>
```



```

function start()
{
    _____
}

function formSubmit( event )
{
    var fm = _____,
        genders = fm.genders,
        gend;

    _____;

    _____

    fm.honorific.value = gend == "Male" ? "Mr." : "Ms.";
}

```

$\alpha$  (3 pts., multiple lines in the listing): Supply the body of function `start()`, which makes function `formSubmit()` a listener for submit events in the form element (with id “frm”).

**Answer**

```

var fm =document.getElementById( "frm" );
fm.addEventListener( "submit", formSubmit, false );

```

$\beta$  (3 pts.): Get a reference to the form element (with id “frm”).

**Answer**

```

document.getElementById( "frm" )

```

$\gamma$  (3 pts.): Turn off the default behavior (submission). We assume the document is being rendered with Mozilla Firefox.

**Answer**

```

event.preventDefault()

```

$\delta$  (5 pts., multiple lines in the listing): Loop over the elements in `genders` and set variable `gend` to the value of the button that was checked.

**Answer**

```

for ( var i in genders )
    if ( genders[i].checked )
        gend = genders[i].value;

```