

COMP 322 Internet Systems Fall 2006 Quiz 3—Solutions

Total points: 20 (10 per problem)

1. The following HTML document executes a function `go` when it is loaded. This function prompts for a model identifier and checks whether the string the user enters is a valid model identifier. A model identifier consists of the following (and nothing more—**hint**: use some anchors):

- an uppercase letter in the range A-D,
- three digits,
- an uppercase letter in the range V-Z,
- two digits,
- then optionally a string consisting of
 - a period (.),
 - an uppercase letter in the range M-N,
 - then one digit.

The three-digit and two-digit substrings are remembered. For example, the following are some valid model identifiers:

B123X98

C555W22.M3

If the string supplied by the user is a valid model identifier, then the three-digit and two-digit substrings are displayed as the content of a `div` element with `id` “display”. The three digits are identified as the “Group”, and the two digits are identified on a separate line as the “Model”. For example, for the first of the two example valid identifiers shown above, the content of the `div` element is rendered as

Group: 123

Model: 98

The following is a listing of the document with gaps identified by Greek letters. After the listing, the Greek letters are repeated with descriptions of the corresponding code. You then supply the missing code.

```

<html>
<head>
  <title>Quiz 3, Problem 1</title>
  <script type="text/javascript">
    function go()
    {
      var ref = α _____,
          regexp = β _____,
          str = window.prompt( "Enter the model identifier",
                               "A999V99" );

      if ( str.match( regexp ) )
        ref.innerHTML =
          γ _____;
      else
        ref.innerHTML = "Invalid model identifier.";
    }
  </script>
</head>
<body onload="go()">
  <div id="display"></div>
</body>
</html>

```

α (2.5 pts.): a reference to the object corresponding to the `div` element with id "display"

Answer

```
document.getElementById( "display" )
```

β (5.5 pts.): the regular expression for matching valid model numbers (as defined above)

Answer

```
/^[A-D](\d{3})[V-Z](\d{2})(\.[M-N]\d)?$/
```

γ (2 pts.): the string, as described above, that becomes the content of the `div` element if the regular expression matches what the user entered. Note that this uses the two strings remembered from the match.

Answer

```
"Group: " + RegExp.$1 + "<br>Model: " + RegExp.$2
```

2. The following Perl program (with gaps) reads in a sequence of number sorted in non-decreasing order, one per line, from a file whose name is given as a command-line argument. It removes the input record separator from each number and pushes it onto the array `@nums`. It also keeps a running sum of the numbers in `$total`. Just after the input loop, the program calculates and outputs the average of the numbers.

The program then finds the median of the numbers in `@nums` (which are already sorted) and outputs the median, the list of numbers in `@nums` that are less than the median (the initial part of `@nums`), and the list of numbers in `@nums` greater than the median (the final part of `@nums`). If there is an odd number of numbers in `@nums`, then (since `@nums` is sorted), finding the median amounts to finding the index midway between the first index (0) and the last index (the length of `@nums` minus 1); that is, the index of the median is the length of `@nums` divided by 2. In the program, for the odd-numbered case, a single statement modifies `@nums` so that it contains only the numbers less than the median, sets `$median` to the median, and sets `@high_nums` to the list of numbers greater than the median.

If there is an even number of numbers, then we follow the convention that the median is the average of the two numbers nearest the middle. For example, if our sequence of numbers is

```
2 4 6 8 10 12
```

then the two numbers nearest the middle are 6 and 8, so we take 7 as the median. In the program, for the even-numbered case, we assign to `$high_mid_index` the index of the second of the two numbers nearest the middle. Then we assign to `$median_ave` the median as just described. Next, in a single statement, we modify `@nums` so that it contains only the lower half of the sequence number (originally in `@nums`) and set `@high_nums` to the upper half of the sequence of numbers.

The following is a listing of the document with gaps identified by Greek letters. After the listing, the Greek letters are repeated with descriptions of the corresponding code. You then supply the missing code.

```

while ( chomp( $num = <> ) ) {
    push @nums, $num;
    $total += $num;
}

print "The average is ", α, "\n";

if ( @nums % 2 == 1 ) {
    β;
    print "The median is $median\n",
          "The numbers below the median are @nums\n",
          "The numbers above the median are @high_nums\n";
}
else {
    $high_mid_index = @nums / 2;
    $median_ave =
        ( $nums[ $high_mid_index-1 ] +
          $nums[ $high_mid_index ] ) / 2;
    γ;
    print "The median is $median_ave\n",
          "The numbers below the median are @nums\n",
          "The numbers above the median are @high_nums\n";
}

```

α (2 pts.): Calculate the average of the numbers.

Answer

```
$total / @nums
```

β (5 pts.): In a single statement, assign to `$median` the middle element in the array `@nums`, assign to `@high_nums` the list of numbers in `@nums` after `$median`, and modify `@nums` so that it contains only the list of numbers before `$median`. **Hint:** Use an assignment statement that has a `splice` expression on the right and a list with two elements on the left.

Answer

```
( $median, @high_nums ) = splice @nums, @nums / 2;
```

γ (3 pts.): In a single statement, assign to `@high_nums` the numbers in `@nums` from `$high_mid_index` to the end and modify `@nums` so it contains only the numbers before `$high_mid_index`. **Hint:** Use `splice`.

Answer

```
@high_nums = splice @nums, $high_mid_index;
```