

Programming in Perl  
CSCI-2230  
Final Exam

Rules and Information:

1. TURN OFF ALL CELULAR PHONES AND PAGERS!
2. Make sure you are seated at least one empty seat away from any other student.
3. Write your first name, last name, and RCS ID (not your RIN!) on the top of EACH page
4. Clear your desks/tables of all Calculators, PDAs, and Laptops
5. You may use any book (Camel, Llama, Mouse, or other) and any typed or hand-written notes.
6. You have until 6:00pm EDT to complete this exam.
7. If you have a question, raise your hand. Paul or one of the TAs will come to you.
8. In all of the questions, "write a program", and "write a script" mean to write a \*complete\* program, from beginning to end (including shebang); "write a piece of code", "write a chunk of code", "write a line" and "write a function" mean just the line(s) of code that satisfy the problem, not an entire program.
9. In any question that requires multiple steps, show all your work to be able to receive partial credit. If no work is given, and the final answer is wrong, you will lose all points for that question.
10. When you have completed the exam bring it to the table in the front of the room
11. Solutions to this exam will be posted on the course website just after 6pm today.
12. When the Exams have been graded, an announcement will be posted to the course website and to the class mailing list informing you where and when you may pick up your exams.
13. Good luck with the remainder of your classes and exams.
14. If you are interested in being an undergraduate Teaching Assistant for this course next semester (assuming it's still offered and Paul is still teaching it...), please email him at [lallip@cs.rpi.edu](mailto:lallip@cs.rpi.edu) for more information or to apply.

## REGULAR EXPRESSIONS (15%)

- 1) Write a regular expression to match a C-style comment. Save the contents of the comment in \$1. Do not save the delimiters or the leading or trailing whitespace. For example: (3pts)

```
/* This is a comment
that happens to
span multiple lines */
```

- 2) What is the output of this code? (3pts)

```
$string = "aaaab";
if ($string =~ /(a*)a(a|b)/){
    print "$1\n";
}
```

- 3) Write a regular expression to match the name of any Perl scalar variable, including the \$. (Don't concern yourself with matching 'special variables' like \$"). Recall that Perl variables must start with a letter or underscore, but the remaining characters may be letters, underscores, or numbers. (3 pts)

- 4) Why would this not be the best way of ending user input? (3 pts)

```
while ($in !~ /^quit/i){
    #do stuff...
    print "Enter an input, or the word \"quit\" to end\n";
}
```

- 5) The following code generates a syntax error. Why? How would it be fixed, without changing the look or readability of the code? (3 pts)

```
$pattern =~ m/          #start the regexp
    (\d*)              #search for 0 or more digits, and save them
    \s+                #at least one whitespace
    \1                 #the same digit pattern
    ([A-Za-z])        #search and save a single letter
    /;                 #end the regular expression
```

**SUBROUTINES (25%)**

- 6) Assuming I have declared a subroutine as `sub SciFi(\%\$\$@)`, which of the following function calls will NOT generate an error? (Check all that apply) (4 pts)

`SciFi(%Space, $the, "Final Frontier", @TheseAreTheVoyages);`  
 `SciFi(%A, "Long Time", "Ago", "In a Galaxy", "Far Far Away");`  
 `SciFi(%SoLong, $and, "Thanks", "For", "All", "The Fish");`  
 `SciFi(%Now, $DrBecket, $FindsHimself, $Leaping, "From Life to Life");`

- 7) Given that the following code is executed: (8 pts)

```
@info = ($stock, $price, $amount);
$buyer = "Paul Lalli";
@recent = (5.1, 5.3, 4.2, 5.0);
```

```
Buy(@info, "now", @recent, $buyer);
```

What are the contents of the following variables within the function `Buy`? (`Buy` does not have a prototype)

`$_[0]` \_\_\_\_\_ `$_[4]` \_\_\_\_\_  
`$_[($#_ - 1)]` \_\_\_\_\_ `$_[5]` \_\_\_\_\_

- 8) What is the output of the following code chunk? (7 pts)

```
$i = 20;
print "\$i = $i\n";
increment();
print "\$i = $i\n";
sub increment{
    my $i;
    $i++;
    print "\$i = $i\n";
}
```

- 9) I want to write a subroutine called `returns()` that will return the last member of the array `@retval` if the subroutine is called in scalar context, but if it is called in list context, it will return a list of the contents of `@retval` in opposite order. For example, assume `@retval` contains (1, 2, 3, 4, 5) just before the end of the subroutine.

```
$last = returns();      # $last should get the value 5
@opposite = returns();  # @opposite should get (5, 4, 3, 2, 1)
```

Write the return line of this subroutine (6 pts):

**CLASSES/MODULES (15%)**

**10)** Write a constructor for a class `Person` that has data-members `Name` and `Age`. Your constructor should set `Name` to the null string, and `Age` to 0. (10 pts)

**11)** Given the following class definition, write a perl script to declare a member of that class, and call its one method on that member: (5 pts)

```
package MyClass.pm;

sub create {
    my $obj = {foo => 20, bar => -20};
    bless $obj, MyClass;
    return $obj;
}

sub change {
    $obj = shift @_;
    $obj{foo}++;
    $obj{bar}--;
}
```

## RANDOM BITS OF PERL (20%)

- 12)** Write a program to ask the user for a command. Find out if it is an actual unix command by running the program `man <command>` (without the brackets). If it is a command, run the command and display the output of the command. If it is not, print a message saying it's not a command. Hint: `man` returns the message `No manual entry for <command>`. if it cannot find an entry, and a description of the command if it can. (5 pts)

Here are two sample run-throughs of your program (user input is bold):

Enter a command:

**ls**

output of ls:

file1.txt

program.pl

page.html

Enter a command:

**blargh!**

Sorry, blargh! is not a command.

- 13)** What is the output of the following piece of code? (show your work for partial credit) (5 pts)

```
@numbers = (1..10);  
@num2 = map {$_ * 4} @numbers;  
@num3 = grep {$_ % 3 == 0} @num2;  
  
print "The contents of \@num3 are: @num3\n";
```

- 14)** What is the output of the following piece of code? (5 pts)

```
$message = "Hello World";  
$value = 42;  
  
print qq(I was told, "Give him the message: $message".\n);  
print q/"What's the answer"? Well, $value, of course!\n/;
```

**15)** Give the contents of each variable after the following chunk of code (show your work for partial credit)

(5pts):

```
$a = 10;
```

```
$b = 40;
```

```
$c = 15;
```

```
$x = -5;
```

```
$y = -10;
```

```
$z = 0;
```

```
($a, $b) = ($b, $a);
```

```
($y, $c, $z) = ($b*2, $x);
```

```
($x) = ($a, $b);
```

```
$c = ($y, $x);
```

## CGI (25%)

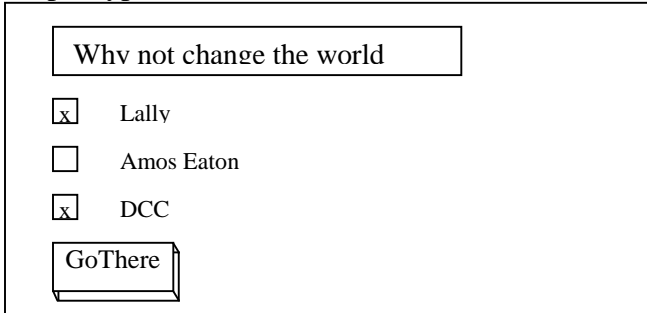
**16)** Why would this code not work properly? (5 pts)

```
#!/usr/local/bin/perl -w
use CGI qw(standard);
print <<HTML;
  <html><head><title>
  My Title
  </title></head>
  <body>
    la la la. Foo bar. B'bye!
  </body></html>
HTML
```

**17)** Assume some HTML form contains two submit buttons. One is named Submit1 (and labeled "Go!"), the other is named Submit2 (and labeled "Now!"). Write a piece of a CGI script to determine which button was pressed, and output a message indicating which was pressed. (You do not have to write the code to actually generate the form or the buttons) (5 pts)

18) The contents of the following form are displayed below. If the user has filled out the form as shown, and dump.cgi simply prints the header, starting HTML, the return value of Dump(), and ending HTML, show the output of dump.cgi: (5 pts)

```
<form method="post" action="dump.cgi">  
<input type="text" name="message"><br>  
<input type="checkbox" name="choice" value="Lally">Lally<br>  
<input type="checkbox" name="choice" value="AmosEaton">Amos Eaton<br>  
<input type="checkbox" name="choice" value="DCC">DCC<br>  
<input type="submit" name="Submit" value="GoThere">
```



19) Write a CGI script to display a form containing a textbox, two radio buttons, and a submit button. The radio buttons are marked "RPI" and "CS Dept". The user of your script enters his RCS user Id in the text box, and chooses one of the two radios. When the user presses the submit button, your script displays a page saying which radio was selected, and creates an HTML link to that user's webpage, on either the RPI or CS Dept system. For example, if I enter my user id and select CS Dept, your script generates:

```
You have chosen the CS Dept. Your home page is  
<a href="http://www.cs.rpi.edu/~lallip">here<a>
```

If I select RPI, the link instead goes to www.rpi.edu/~lallip

If the user does not enter a UserId, you should display an error message and nothing else. (10 pts)