

CSCI-2230 Programming in Perl

Final Exam Review Questions

The final exam will cover everything we have done in the 2nd half of the class, beginning with Regular Expressions. This includes: RegExps, Subroutines, Objects, Modules, "Random Bits of Perl" (eval, quoting operators, backticks, map, grep, each, \$!, list assignment), References, and CGI programming.

The exam will be open-book, open-note. You may use the Camel, the Llama, another Perl text, your own notes, and/or class notes. You may not use Calculators, PDAs, Laptops, or any other student.

Following are some sample questions for the final exam:

Regular Expressions

1) Write a Regular expression that matches the following:

a) at least one a followed by any number of b's

```
/a+b*/
```

b) three consecutive copies of the value stored in \$foo

```
/$foo{3}/
```

c) any five characters, including the newline

```
/.{5}/s
```

d) any 'word' (as defined by perl) two times, separated by any other different word (with any amount of whitespace in between the words). For example:

```
Hello world5 Hello
```

```
/(\w+)\s*(\w+)\s*\1/
```

2) Write one line of code that will search a string \$email for an email address, and store the user name in \$user, and the domain in \$domain. For example, if \$email contains lallip@cs.rpi.edu, after your one line, \$user will contain lallip and \$domain will contain cs.rpi.edu

```
($user, $domain) = ($email =~ /(.*)\@(.*)/);
```

3) What is printed in the following example?

```
$string = "Name=Paul;RIN=660066466;email=lallip@cs.rpi.edu";
```

```
$string =~ /Name=(.*/);
```

```
print "Name: $1\n";
```

Name: Name=Paul;RIN=660066466

4) What about this one?

```
$phrase = "Do not go gently into that dark night\nRage, Rage\nAgainst the dying of the light";
```

```
@firsts = $phrase =~ /^(\w+)/gm;
```

```
print "@firsts\n";
```

```
@lasts = $phrase =~ /(\w+)$/g;
```

```
print "@lasts\n";
```

Do Rage Against

light

User subroutines

5) What is the output of this code?

```
sub fun1{
    $i=0;
    foreach (@_){
        print "$i: $_\n";
        $i++;
    }
}
$name = Paul;
fun1 "hello", "world", "and $name!";
```

0: hello
1: world
2: and Paul!

6) Write a function that will accept a string and a list. The function will return a string consisting of each member of the array prepended by the string. For example, if the function is called by

```
prepend("good", ("morning", "evening", "boy"));
```

it will return the string

```
"good morning good evening good boy"
```

Perl should give an error if incorrect arguments are passed (ie, first argument is not a scalar, second is not a list, or incorrect number of arguments)).

```
sub prepend($@){
    $attach = shift;
    foreach (@_){
        $ret_val .= "$attach $_";
    }
    chop $ret_val;
    return $ret_val;
}
```

7) What is the difference between the following two styles of calling a function?

```
my_funct ("Hello", 3);
```

```
my_funct "Hello", 3;
```

In the first, my_funct does not have to be declared before being called.

In the second, it does.

Objects

8) 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.

```

package Person;
sub new {
    my $ref = {Name => "", Age => 0};
    bless ($ref, Person);
    return $ref;
}

```

9) Below are the entire contents of a file called MyClass.pm Why won't the Perl interpreter accept it? (ie, it won't compile. Why?)

```

package MyClass;

sub new {
    my @students = ();
    my $MyObj = {Day => "Wednesday", Room => "DCC330", Time => "4pm-6pm",
Enrollment=>0, students => \@students};
    bless $MyObj, MyClass;

    return $MyObj;
}

sub register{
    my ($ref, $new_student) = @_;
    $ref->{Enrollment} ++;
    push @{$ref->{students}}, $new_student;
}

sub print{
    my $ref = shift;
    print "Students in this class are: @{$ref->{students}} \n";
    print "And the Enrollment is $ref->{Enrollment}\n";
}

```

The file doesn't end with the statement
1;

Random Bits of Perl

10) Re-write the following statement without using any \ characters. Note that the statement you write must be functionally equivalent to this statement:

```
print "$name said, \"What do you mean?\", and I answered, \"Well, $father told me, 'Do it!'\"";
```

```
print qq($name said, "What do you mean?", and I answered, "Well $father told me, 'Do it!'");
```

11a) Give the values of each of the following variables at the end of this code: \$a, \$b, \$c, \$d, \$e

```

$a = 5;
$b = 10;
$c = "foo";
($d, $a) = ($a, $b);
($e, $b, $c) = (5.1, "hello");

```

```

$a = 10, $b = "hello", $c = undef,
$d = 5, $e = 5.1

```

11b) Write one statement of code that does the equivalent of the following three statements (the value of \$temp is irrelevant at the end of your statement):

```
$temp = $a;  
$a = $b;  
$b = $temp;
```

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

CGI Programming

12) What should be the first two lines of every single CGI script you write?

```
#!/usr/local/bin/perl -w  
use CGI ":standard" #or some variant of this
```

13) If more than one form element has the name "foo", what does param('foo') return in list context? In scalar context?

list = list of all values of parameters with name foo; scalar = value of first parameter with name foo

14) Give the CGI 'shortcut' for the following piece of HTML:

```
<input type="text" name="Age" value=18>  
textfield(-name=>"Age", -value=>18);
```

15) Write a small CGI program that will create a field for the user to enter the name of a file on the user's machine. When the user presses submit, the program will print the contents of that file to the screen in reverse order (ie, last line first, second-to-last line second, ..., first line last)

```
#!/usr/local/bin/perl -w  
use CGI ":standard";  
print header;  
print start_html;  
if (!param){  
    print start_multipart_form;  
    print filefield(-name=>'file');  
    print submit(-name=>'Submit', -value=>'Submitted');  
    print end_form;  
} else {  
    $file = param('file');  
    @contents = <$file>;  
    @contents = reverse @contents;  
    foreach (@contents){  
        print;  
        print br;  
    }  
}
```