

CS306 – Introduction to Perl  
Summer 2006  
Homework Assignment #1  
Due: June 14<sup>th</sup>, 11:20am

Please follow the guidelines on HW0 on how to submit the homework.

### Homework #1

**Question 1 (15 minutes, 15 pts.)**. List the three types of data in Perl and how they differ from one another.

**Question 2 (30 minutes, 35 pts.)**. Examining List Assignment. We know that we can use list assignment to initialize an array, like this:

```
@names = ("jack", "jill");
```

However, an often overlooked feature of Perl is the ability to assign one list to another, like this:

```
($boy, $girl) = ("jack", "jill");
```

The elements are copied over from the right to the left in the order they appear. In this manner, we can assign to several independent scalars from the same list. This is really handy with functions that return lists, such as `localtime()`.

```
($sec, $min, $hour, $mday, $mon, $year, $wday, $yday, $isdst) = localtime();
```

`localtime()` returns a 9-element list. By looking at the documentation (`perldoc -f localtime`) we determine that element 0 is the seconds, element 1 is the minutes, and so on. We can even use the slicing feature of lists to make this more precise:

```
($day, $month, $year) = localtime()[3-5];
```

Let's explore this feature further.

- a. What happens when there are fewer scalar variables on the left than there are values in the list on the right?
- b. What happens when there are more scalar variables on the left than there are values in

the list on the right?

- c. What happens when you have a list on the left side but a scalar value (not a list with one element, but a standalone scalar value!) on the right side?
- d. What happens when you have a single scalar variable on the left side and a list of values on the right side?

**Notes on programs: While writing these programs below, remember to comment thoroughly, including your name near the top as well as comments throughout explaining what you are doing in your program.**

Program 1: Guess The Number (45 minutes, 20 pts)

Write a program to play a guessing game with the user. The user has to guess a number from 1-100. After a guess is entered, tell the user whether they are too high or too low, and let them guess again, until they get the correct answer. When they get the right answer, tell them how many guesses it took for them to get there. This bit of code will be helpful:

```
# Get a random number in the range 1-100
$number = int(rand(100)) + 1;
```

Program 2: Exchange Rate (1 hour, 30 pts)

Write a program that converts from US dollars to various other currencies. Store the exchange rates in a hash. I suggest looking at XE.com for appropriate hash keys and values. When you run your program, it should ask the user for a US dollar amount and a target currency, and report the value of the US dollars in the other currency. Have at least five foreign currencies available. Make sure your program tells the user which foreign currency conversions are available. Have the program loop so that the user can do more than one conversion. The program should be case-insensitive – it should work whether I enter input in upper or lowercase letters. Hint: lc()