

CS306 – Perl Programming
Spring 2009, Homework Assignment #1
Due: Monday, January 26th, 4pm

Guidelines for Submitting Homework

All homework should be submitted in the form of a single zip file which contains one program for each question. Your zipfile should be named *lastname-hw1.zip* and should contain a single text file called *answers.txt* for the written answers, and then files named *hw1p1.pl*, *hw1p2.pl*, *hw1p3.pl* *hw1p8.pl* for the programs. **Make sure your name appears in all source code files!** Email this zip file to cs306@cis.uab.edu. **When emailing this file email a copy to yourself so you know the email went through ok.**

Homework #1 (150 total points)

Question 1 (15 points). List at least three ways in which Perl is different from Java.

Question 2 (8 points). Explain and illustrate the concept of the default variable (`$_`) in Perl. You should include a code sample other than what appeared in class or the book.

Question 3 (12 points). Context. Indicate whether each of the following operations are in scalar or list context.

- a) `$count = 16 + 4;`
- b) `($num1, $num2) = ($num2, $num1);`
- c) `$last = $#array;`
- d) `@array = qw/red yellow green/;`
- e) `@newarray = reverse(@array);`
- f) `$value = @newarray;`

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

Also, turn on warnings in all of your programs. Here's a sample program:

```
#!/usr/bin/perl -w
# the above line must always be the very first line
# in your program!
# Joe Student
# CS306, SP09, HW1

print "Hello World!\n";
```

Read each problem description carefully, and be sure to address everything that is asked for in the problem.

A NOTE ON DEBUGGING: WHEN YOU GET STUCK, PRINT OUT YOUR VARIABLE VALUES AT VARIOUS POINTS IN YOUR PROGRAM. THIS IS OFTEN THE EASIEST WAY TO FIND WHERE THE CODE IS BROKEN.

Program 1: Guess the Secret Password (10 points)

Write a program which asks a user to guess a secret password. If they don't get it right in three tries, exit the program. The secret word should be "cheeseburger" (case-sensitive!) to make it easy to test, and this can be hard-coded into your program.

Sample Output:

```
Guess the secret word: jello
Sorry, that is incorrect.
Guess the secret word: pizza
Sorry, that is incorrect.
Guess the secret word: cheeseburger
You're right!
```

Program 2: Guess the Number (15 points)

Write a program which makes the user guess a number between 1-100. Here's how to generate a different random number between 1-100 (inclusive) each time you run the code:

```
$num = int(rand(101));
```

You should then write a loop which asks the user to guess a number until they get it right. After each guess, tell them to guess higher or lower to help them out. When the user guesses the right number, tell them how many guesses it took them, and then exit.

Sample Output:

```
Guess the secret number: 50
You are too low.
Guess the secret number: 75
You are too low.
Guess the secret number: 87
You are too high.
Guess the secret number: 81
You are too high.
Guess the secret number: 78
You are too high.
Guess the secret number: 77
You are too high.
Guess the secret number: 76
You win! It took you 7 guesses.
```

Program 3: Geometry (10 points)

Write a program to ask the user for a number from 1-10, and then compute and display the following:

- The area of a circle which has a radius of that value
- The circumference of a circle which has a radius of that value

- The area of a square which has a side length of that value
- The area of an isosceles right triangle with legs of that length
- The length of the hypotenuse of that isosceles right triangle (hint: `perldoc -f sqrt`)

Sample Output:

```
Please give me a number between 1-10: 5
The area of a circle with radius 5 is 78.53975.
The circumference of a circle with radius 5 is 31.4159.
The area of a square which has a side length of 5 is 25.
The area of an isosceles right triangle with legs of length 5 is
12.5.
The length of the hypotenuse of that triangle would be
7.07106781186548.
```

Program 4: Football Scout (10 points)

You've been asked to evaluate the performance of potential athletes for your football team, and you're looking for the right combination of speed and strength. You'll give a scholarship to anyone who can meet one of the following conditions:

- Bench press at least 300 pounds
- Run 100 meters in 10.1 seconds or less
- Bench press at least 220 pounds AND run 100 meters in 10.5 seconds or less

Write a program which takes as input the number of pounds the athlete can bench press and the speed in which the athlete runs the 100 yard dash, and determines whether or not you should offer that person a scholarship based on whether they meet the criteria.

There's a catch! Your program may only contain `-one-` `if` statement and no `elsif` clauses.

Sample Output:

```
How much can you bench press in pounds? 170
How fast can you run 100 meters in seconds? 10.4
Keep working in the gym.
fran$ ./hw1p4.pl
How much can you bench press in pounds? 230
How fast can you run 100 meters in seconds? 10.3
Welcome to campus!
fran$ ./hw1p4.pl
How much can you bench press in pounds? 310
How fast can you run 100 meters in seconds? 12.5
Welcome to campus!
```

Program 5: Sum the Elements of a List (15 points)

Write a program which prompts the user to enter a list of 5 numbers, one number per line. Place those numbers into an array. Then, iterate over the array and produce the sum of all of the elements in the list.

Note: This program must first place all of the numbers into an array, and then iterate over the array to produce the sum. Simply producing the sum as the user enters the numbers will not receive credit.

Sample Output:

```
Enter 5 numbers, one per line.  
5  
7  
2  
4  
6  
The sum of the numbers is 24.
```

Program 6: School Pride (15 points)

Create two arrays, @schools and @nicknames. Populate these with at least 5 school-nickname pairs. e.g. if \$schools[0] is "UAB" then \$nicknames[0] should be "Blazers". These pairs should be hard-coded into your program as list literals. Then write a loop to print out sentences like "The nickname for UAB is the Blazers.", one per line, for each pair.

Sample Output:

```
The nickname for UAB is the Blazers.  
The nickname for Georgia is the Bulldogs.  
The nickname for Auburn is the Tigers.  
The nickname for Florida is the Gators.  
The nickname for Tennessee is the Volunteers.
```

Program 7: Outputting Sorted Lists (15 points)

Write a program that reads in a list of words, one per line, from the user. The list may have any number of words – the user will signal the end of the list by sending the EOF character (Ctrl-D on Linux). Then, output the list of words, one per line, in ASCIIbetical sorted order.

```
Enter a series of words, one per line. Hit Ctrl-D when finished.  
red  
green  
blue  
orange  
yellow  
white  
black  
brown  
red  
^D  
Those same words in sorted order are:  
black
```

blue
brown
green
orange
red
red
white
yellow

Program 8: The Ticket Line (25 points)

Write a simple, menu-driven program that represents a movie theater line as a FIFO queue. The menu has four options: (a)dd someone to the back of the line, (c)all the next person from the front of the line, (p)rint the list of people in line, and (q)uit. If you add someone, the program prompts for their name. If you call someone, the program reports the name of the person at the front of the line. Print simply prints the list of people in line. Quit exits the program.

Sample Output:

Welcome to Fran's Ticket Emporium!

Choose an option:

(a)dd person to end of line
(c)all next person in line
(p)rint the list of people in line
(q)uit

Choice: a

Name: Fran

Fran added to back of line.

Choose an option:

(a)dd person to end of line
(c)all next person in line
(p)rint the list of people in line
(q)uit

Choice: a

Name: Jill

Jill added to back of line.

Choose an option:

(a)dd person to end of line
(c)all next person in line
(p)rint the list of people in line
(q)uit

Choice: p

The line is: Fran Jill

Choose an option:

- (a)dd person to end of line
- (c)all next person in line
- (p)rint the list of people in line
- (q)uit

Choice: c

Fran is next.

Choose an option:

- (a)dd person to end of line
- (c)all next person in line
- (p)rint the list of people in line
- (q)uit

Choice: c

Jill is next.

Choose an option:

- (a)dd person to end of line
- (c)all next person in line
- (p)rint the list of people in line
- (q)uit

Choice: c

No one in line.

Choose an option:

- (a)dd person to end of line
- (c)all next person in line
- (p)rint the list of people in line
- (q)uit

Choice: a

Name: Tony

Tony added to back of line.

Choose an option:

- (a)dd person to end of line
- (c)all next person in line
- (p)rint the list of people in line
- (q)uit

Choice: p

The line is: Tony

Choose an option:

- (a)dd person to end of line

(c)all next person in line
(p)rint the list of people in line
(q)uit

Choice: c
Tony is next.

Choose an option:
(a)dd person to end of line
(c)all next person in line
(p)rint the list of people in line
(q)uit

Choice: q
Goodbye.

Optional: Challenge Problems (Bonus Points)

There is no partial credit for the Challenge Problems. Your program must work for ALL test cases, and we will purposefully pick data which tests the corner cases and special case scenarios.

Program 9: Scout Advice (10 points)

Write an extension to Program 4, the Football Scout program so that if they don't meet any of the criteria, print out helpful advice like “You need to bench press at least 140 more pounds, reduce your 100 meter time by 1.2 seconds, or bench 60 more pounds and reduce your 100 meter time by 0.8 seconds to receive a scholarship.” Note that if they've already met one of the criteria for the combined case (third bullet point), your advice changes to something like “You need to bench 60 more pounds or reduce your 100 meter time by 0.3 seconds to receive a scholarship.”

Program 10: Number Sorter (10 points)

Write a program which will take in a list of numbers (which each must be between 1-1000), one per line, from the user, and print them out in numerically sorted order, highest to lowest, **using only the material we learned in Ch. 1-3 of Learning Perl.**