**QUIZ 2**

*Ask a user to input department number (1-10), employee ID number (1-5000), and years of service (1-40). After data is entered, display total years of service for all employees in that department.*

* **Preparation**

Like most Unix programming files, it must first be created and opened using some type of text editor -

***vi employee\_service.pl***

The shebang line of the program will determine the correct interpreter for use in executing the file -

***#!/usr/bin/perl***

In order to catch and prevent very many errors in our program, certain “use” commands can be put in place, by employing the programmer to use certain keywords for performing functions -

***use strict; use warnings;***

Each input (Dept. Number, ID Number, and Service) has a set range of numbers, which we can specify using constants -

***use constants {***

***DEPT\_MIN => 1,***

***DEPT\_MAX => 10,***

***ID\_MIN => 1,***

***ID\_MAX => 5000,***

***YEARS\_MIN => 1,***

***YEARS \_MAX => 40,***

***};***

* **Prompting user input**

Now we need to prompt the user to input all 3 values as a worker. First we start in our main function ( **sub main {** ). For each value awaiting assignment, we call a separate function ( **input\_valid** ), assigning that calling to a variable of a corresponding name -

***my $department = input\_valid(“Please enter your department number”, DEPT\_MIN, DEPT\_MAX);***

***my $employee\_id = input\_valid(“Please enter your employee ID”, ID\_MIN, ID\_MAX);***

***my $years = input\_valid(“Please enter your years of service”, YEARS\_MIN, YEARS\_MAX);***

Each of these will send the variable to the input\_valid function ( **sub input\_valid {** ). Firstly, we can assign our prompt, minimum, and maximum arguments sent into the array to three new variables. We can also create a variable for storing user input -

***my ($prompt, $min, $max) = @\_;***

***my $input;***

The @\_ variable contains the arguments passed into the function.

The function will then prompt the user for input, via a do function -

***do {***

***print “$prompt ($min-$max) - “;***

***#OUTPUT EXAMPLE - “Please enter your department number (1-10) - “***

Next to actually allow for user input, using <STDIN> -

***$input = <STDIN>;***

Pressing enter after giving our text input will add a character to our string. This ENTER character can be removed by a “chomp” function -

***chomp($input); }***

Now to manage our input into data. We can use a **while** to continue looping our do function until certain arguments are met -

***while ($input !~ /^/d+$/ || $input < $min || $input > $max);***

This while clarifies that if the input is a non-positive integer (/^/d+$/), the input is less than its minimum value, or is greater than its maximum value, the prompt will keep looping until a correct input is typed in. Once this happens, we can send this input back to our main function, ending our input\_valid function -

***return $input;***

This function will apply to all three calls, for each of the three user inputs.

For clarity’s sake, we can display our inputs with various print lines -

***print “Dept. Number - $department \n”;***

***print “ID Number - $employee\_id \n”;***

***print “Years of Service - $years \n”;***

* **Displaying total years of service for dept.**

I am under the assumption that my goal here is to display the collective total of every single employee within my department as a single number, rather than the number of years for each individual employee.

To accomplish this, we need to keep a unit-wide count of everyones hours, which can be kept in a hash variable -

***my %dept\_years;***

After all the calls to input\_valid, we update our dept\_years variable, based on our selected department, and updates that departments year count based on the inputted years of service, then we can later on print the total years for that particular unit -

***$dept\_years{$department} += $years;***

***Print “Total Department Years - $department\_years{department} \n”;***

This is the end of our main function, but in order to actually run the code within our main function, we must call it at the very end of our program, unlike programming languages like C++

***main();***

**#!/usr/bin/perl**

**# employee\_service.pl - Joseph Opolka**

**use strict;**

**use warnings;**

**use constant {**

**DEPT\_MIN => 1,**

**DEBT\_MAX => 10,**

**ID\_MIN\_ID => 1,**

**ID\_MAX => 5000,**

**YEARS\_MIN => 1,**

**YEARS\_MAX => 40,**

**};**

**sub input\_valid {**

**my ($prompt, $min, $max) = @\_;**

**my $input;**

**do {**

**print "$prompt ($min-$max): ";**

**$input = <STDIN>;**

**chomp($input);**

**}**

**while ($input !~ /^\d+$/ || $input < $min || $input > $max);**

**return $input;**

**}**

**sub main {**

**my %dept\_years;**

**my $department = input\_valid("Please enter your department number", DEPT\_MIN, DEBT\_MAX);**

**my $employee\_id = input\_valid("Please enter your employee ID", ID\_MIN, ID\_MAX);**

**my $years = input\_valid("Please enter your years of service", YEARS\_MIN, YEARS\_MAX);**

**$dept\_years{$department} += $years;**

**print "\nDepartment Number - $department\n";**

**print "ID Number - $employee\_id\n";**

**print "Years of Service - $years\_of\_service\n";**

**print "\nTotal Department Years (department $department) - $dept\_years{$department}\n";**

**}**

**main();**