Programming in Perl, Spring 98

Homework 1

Due : Monday, April 6 at midnight

Weight : 10% of course grade

Instructor : Louis Ziantz

This assignment consists of a series of small Perl programs. It should be submitted according to the *Submission Guidelines*. Don't forget to comment your program, providing a general description of the Perl script along with inline comments for important or complex portions of code.

Problem 1 Write a Perl script that finds the median(s) of a list. Informally, a median of a list is an item which is "halfway" down the list (that is, there are as many items above and below the median in the list). More precisely, if there are n items in a ordered list and n is odd, there is a single median at location $\frac{n+1}{2} - 1$. If n is even, there are two medians at locations $\frac{n}{2} - 1$ and $\frac{n}{2}$. Though the term median is normally applied to numerical data, your script median.plx should find the median of a list of words. The script should allow the user to enter a series of words (or other lines of text), each followed by a newline, with the entire list terminated by a Ctrl-D. It should read the lines into an array and sort them based on ASCII ordering. The program should then print the median or medians of the list. To make testing easier, you can generate a list of words using the editor of your choice. If the resulting file is named words.dat, the script may then be run as `median.plx < words.dat`.

Problem 2 Write a Perl script called histogram.plx that prints a histogram for a set of integers ranging from 0 to 99. The histogram will be divided into ten intervals, each with a ten point range. That is, the zeroth interval represents values from 0-9 while the ninth interval represents values from 90-99. An array can be used to maintain the number of values that fall in each interval of the histogram. The script should process integers until a -1 is encountered. As each value is read from STDIN, the appropriate location in the array should be incremented. Remember that in Perl division always yields a decimal value. If `$x` is a variable holding a floating-point value, it may be truncated to an integer value using the Perl builtin function `int` (that is, `int($x)`). Once all data have been input, the histogram should be printed. Histogram intervals should be output one to a line. Each line should begin with the interval number, followed by a space, a colon, and another space. An asterisk (`*`) should be printed for each value falling in an interval. Thus, if the array indicates that the zeroth interval contains 6 values, `0 : ******` should be printed as the first line. Given a file integers.dat with one number to a line (and -1 as the last value), the script may be run as `histogram.plx < integers.dat`.

Problem 3 A small mail-order compact disc company uses the price codes and corresponding prices as given below on the left and charges shipping and handling based on the total number of items ordered as indicated below on the right:

Code	Price
CL1	12.98
HR1	11.98
CL2	10.98
SR1	11.98
SR2	14.98
SR3	24.98

CDs	Charge
1–3	3.49
4–6	6.49
7–9	8.99
10+	10.99

Store the price information as a hash called `%code_to_price` using the price codes as keys and the prices as values. The shipping and handling charges should be stored in an array. A nested if structure can be used to select appropriate elements of the array when calculating shipping and handling.

Write a Perl script cd.plx that processes a customer's order. The script should first prompt for a customer's account number. You may assume any account number entered is valid. The program should then display the following menu:

```
P -- Place a CD on order.
R -- Remove a CD from on order.
T -- Total bill and list CDs ordered.
E -- Exit.
Type choice and hit enter:
```

The script should respond correctly to either a lower or uppercase character. The actions to be taken for each choice are as follows:

P
- prompt the user to enter a code for the CD to be purchased
- read the cd code input (you may assume any code entered is valid)
- prompt the user to enter a price code
- read the price code input
- use the price code entered to index the `%code_to_price` hash. If the corresponding value is defined, use the cd code as a key to a hash called `%purchase` and store the price as a value of `%purchase`. If there is no value in `%code_to_price` for the price code entered, print a message that an invalid price code was input and do not modify the `%purchase` hash (do not reprompt; display the main menu again).

R
- prompt the user to enter a code for the CD to be purchased
- read the cd code input (you may assume any code entered is valid)
- if there is no value corresponding to the cd code in `%purchase`, print an appropriate message (do not reprompt); otherwise, delete the location in `%purchase` associated with the cd code entered

T
- output a line indicating the customer's account number
- output a header for two columns that will be printed; it should consist of "CD code", two tabs, and "Price"
- print the cd code and price for each order stored in `%purchase` (separate them by two tabs)
- use each price to total the charges for the customer. In addition, count the number of items purchased.
- use the number of items purchased to determine shipping and handling charges and add this to the total bill (ignore possible taxes)
- print the number of items purchased and the total cost to the customer
- some example output:

```
Purchases for account number M21C:
CD code         Price
CL3456          $12.98
CL3367          $10.98
SR2345          $11.98
SR8889          $24.98
Total items on order: 4
Total due (includes shipping and handling): $67.41
```

other
- indicate that the choice entered was invalid

The script should continue prompting for choices until an 'E' or 'e' is input by the user.

Recall from the first lecture that in order to run a Perl script, you need to do the following:

1. place `#!/usr/local/bin/perl -w` at the beginning of the script file on a line by itself (assume the file is named script.plx)
2. execute the following command at a UNIX prompt: `chmod a+x script.plx`
3. run the script by typing the following at a UNIX prompt: `script.plx`