Project 1

|  |  |
| --- | --- |
| Due Date: | Wednesday, June 16, 2021 on GL |
| Assignment Type: | Individual (no external help and no groups!) |
| Assignment Title: | Project 1 – Bash and Regex |
| Style: | Submit all before due date to GL |
| External Sources: | You are allowed notes, books, and searches (Do not search for the actual solution – like do not Google “Project 1 – Bash and Regex”) |
| Description: | This lab covers Regex topics including quantifiers, anchors, and matching spaces. Please review the page below for all user input asked for in the problem set given. |
| Points | 80 |
| Starting Files | mkdir project1cd project1cp –r /afs/umbc.edu/users/j/d/jdixon/pub/cs433/proj1 . Submit all completed files to:  **/afs/umbc.edu/users/j/d/jdixon/pub/cmsc433/$USER/proj1** |

# Regex (20 points)

1. Create the Perl script “**proj1\_1.pl**” that print lines that contain an opening and closing XML heading tag from “**proj1\_data.txt**”. Valid tags are name, price, description, and calories regardless of the case. Tags may also have any arbitrary content inside of them. You may assume that a '<' or a '>' character will not appear inside of the attribute's value portion.

A few substitutions that are required:

* 1. The calories are too high – display 125 calories less than in the data.
  2. Description should be enclosed in " "
  3. Add a random adjective to the beginning of each name such as “Delicious” or “Heavenly” for all food items. There must be at least three adjectives available.
  4. The price is too low, add $0.50 to each price.

The output should look like this:

Delicious Belgian Waffles  
$5.95   
Two of our famous Belgian Waffles with plenty of real maple syrup  
525 calories

# Regex 2 (20 points)

1. This problem will be looking for and sanitizing credit card numbers from the 4 major credit card companies. For this problem you will be following the simple rules laid forth in the code below, whereas in the real world you would continue validation to make sure that it is a valid card number by performing the [Luhn formula](https://en.wikipedia.org/wiki/Luhn_algorithm) to compute the checksum of the card number and compare it against the check digit (the last digit on the card).

#!/usr/bin/perl

# Find and replace all instance of valid Visa, MasterCard, Discover or

# American Express card numbers with '...' followed by the last 4 numbers.

# The following table details the digits each card must begin with as well as

# the number of digits allowed for each card. In order to be a valid card

# number it must not surrounded by a "word" character (in the regex sense)

#

# Card Type Starts With Number Digits

# --------------------------------------------------------------------

# American Express 34 or 37 15

# Discover 65 or 6011 16

# MasterCard 51 through 55 16

# Visa 4 13 or 16

#

# For example, the following lines...

#

# 'not my amex: 341234567890123'

# '412345678901234'

# '6011012345678901 is a discover card'

#

# ...would get converted to...

#

# 'not my amex: ...0123'

# '412345678901234'

# '...8901 is a discover card'

# Bash (40 points)

1. In this problem, you are going to update a script that pulls stock data from the Yahoo finance website so that the user can include a starting date and ending date and the data will be filtered based on the dates entered.

Yahoo! Finance maintains an archive of historic stock data that can be easily accessed. For this project, a starting script has been provided from here: <http://github.com/bradlucas/get-yahoo-quotes>

Go ahead and download the script and test it. For example, to download all historical stock prices for Apple (AAPL) we would execute **./get-yahoo-quotes.sh AAPL**

You can read about Unix time here: <https://en.wikipedia.org/wiki/Unix_time>

This program will produce a CSV file with all the data. You will need to modify this script (make a copy of the script, and start editing your own version) to make sure that your script sets a start and end dates based on user input. You will ask the user for a starting date and an ending date in the format of 5/15/2021.

Make sure to automatically use the correct end day depending on the user-selected month (otherwise when you use date to convert it to seconds for START\_DATE and END\_DATE you will have a problem). We must automatically specify the end day depending on the user-selected month. Special care is necessary for the month of February in leap years.

Hint 1: A case conditional structure predicated on the user-selected month may be useful. Think about nesting a second case conditional block to handle February.

Hint 2: Pseudo-code to determine leap years: <http://en.wikipedia.org/wiki/Leap_year#Algorithm>

Hint 3: The special bash variable $? holds the return code for the most recent call made in your script. It will hold a value of 0 if the call was successful and will hold a non-zero value if the call failed. You may wish to use this special symbol in an if loop (e.g., if [ $? -ne 0 ] ; then) to determine whether or not your call to wget succeeded.