CSE 337: Homework Assignment 2

Instructions

Please read the the following instructions carefully before coding. You may lose points if you fail to follow these instructions.

- Deadline for this assignment is **March 24, 11:59PM. No late submissions accepted**.
- Stick with built-in Perl modules unless otherwise specified
- Keep your answers for each question in a separate file, and specify the file name clearly (*e.g.,* q1.pl for question 1). For the questions with multiple parts, you can use a filename like q1_p1.pl
- Make sure your programs work in other machines. One way to test this is to test your program in our UNIX servers, and see if it works there. You will lose marks if your program fails to run in our machines.
- Put all of your Perl files, and necessary input files in a single folder (Do not make a sub-folder for each question).
- **The questions will be graded using a script unless otherwise specified. Hence, please strictly adhere to the sample input-output formats provided.**

  Note: Please rename your folder in the right way before zipping! For example, if your name is Trung Nguyen and your student ID is 111345678, put all submission files in a folder named Trung_Nguyen_1112345678, then compress that folder to Trung_Nguyen_1112345678.zip.

1 User Account Management in Unix/Linux [20 pts]

On unix/linux machines, the file /etc/passwd contains a list of all the users on a system. Each line looks like:

```
root:*:0:0:System Administrator:/var/root:/bin/sh
daemon:*:1:1:System Services:/var/root:/usr/bin/
nobody:*:-2:-2:Unprivileged User:/var/empty:/usr/bin/false
```

Each line contains a user name, password (all * here), user id, group id, name, home directory, and shell. These fields are separated by colons. A sample passwd file is provided at "passwd.txt".
Part 1. Write a program that prints out the usernames where the user id is an even number. [5 pts]

Part 2. Print all usernames that are sorted by their user id decreasingly. [5 pts]

Part 3. Print the number of users in each group. [5 pts]

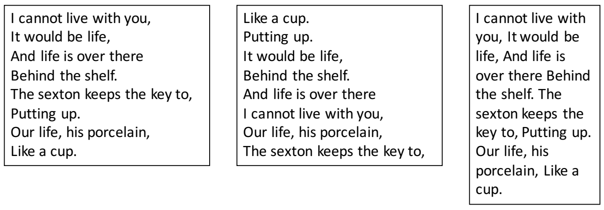
Part 4. Write a subroutine that adds a given user at the end of the list and save to 'passwd.txt' file. You need to assign an idle user id and an existing group id to the new user, meaning the no two users share the same user id. Please find the maximum user id in current list, increase it by 1 and use that to be the user id for the new user. Assign an existing group id randomly. You can specify other entries freely, e.g., user name, home directory, etc. [5 pts]

Sample outputs for this question are provided in sample_output_q1_q2.txt.

2 Text Reformatter [15 pts]

Suppose you are given a piece of text (a sequence of lines):
Part 1. Write a subroutine to sort this piece of text in increasing order of the length (number of total characters including whitespace) of each line. Ties can be ignored: Order between lines with the same length does not matter. [7 pts]

Part 2. Write a subroutine that takes an integer parameter specifying the maximum width of each line, and reformat the given text to fit the given line width as much as possible. The program CANNOT break words using hyphen. [8 pts]
Sample input and output:



```
I cannot live with you,
It would be life,
And life is over there
Behind the shelf.
The sexton keeps the key to,
Putting up.
Our life, his porcelain,
Like a cup.
```

```
Like a cup.
Putting up.
It would be life,
Behind the shelf.
And life is over there
I cannot live with you,
Our life, his porcelain,
The sexton keeps the key to,
```

```
I cannot live with
you, It would be
life, And life is
over there Behind
the shelf. The
sexton keeps the
key to, Putting up.
Our life, his
porcelain, Like a
cup.
```

Figure 1: Text formatter example. On the left is the original text, on the center is the output of first subroutine, on the right is the output of the second subroutine.

Sample outputs for this question are provided in sample_output_q1_q2.txt. Also, a sample input is provided in sample_input_q2.txt.

3 List Operators and Hashes [20 pts]

Part 1. Write a program that takes a text file as an input (as a command line argument). The program should print out the total number of lines, words, and characters in the text file, and the content in the reverse order, both in terms of lines and in terms of characters in each and every line. [10 pts]
Note: Except the new line character, everyother type of characters should be counted.
Sample input and output:

```
$ cat input21.txt
Q1.1. List Operators
Write a program that takes a text file as an input.

$ perl q3_p1.pl input21.txt
lines: 2, words: 14, characters: 72
reversed:
.tupni na sa elif txet a sekat taht margorp a etirW
srotarepO tsiL .1.1Q
```

Part 2.
Below is the table for currency exchange **from** USD **to** 7 other currencies:

	EUR	CAD	INR	JPY	VND	KRW	BTC
USD	0.81	1.29	65.2	105.75	22750	1079.43	0.000088

Write a program that allows the user to input an amount of money in one of the currencies provided above and calculate the corresponding amount in the target currency. The program **must use a hash** of currency - ratio pairs and be able to handle invalid currency input. [10 pts]
Sample input and output:

```
$ perl q3_p2.pl
Exchangeable currency: usd, eur, cad, inr, jpy, vnd, krw, btc
Enter the current currency: btc
Enter the target currency: jpy
Enter the amount of money: 1
1 btc is 1201704.55 jpy.

$ perl q3_p2.pl
```

```
Exchangeable currency: usd, eur, cad, inr, jpy, vnd, krw, btc
Enter the current currency: zwl
We do not trade zwl!
Re-enter the current currency: usd
Enter your target currency: xyz
We do not trade xyz!
Re-enter the target currency: usd
Enter the amount of money: 1000
1000 usd is 1000 usd.
```

4 Files and Directories [20 pts]

Part 1.

Write a program that writes a string “Perl is cool!” to a file as follows:

1. First, ask the user to input the name of the file to write the string to.
2. - If the input file exists, check if there is a subdirectory named "backup". If there is, print "Checking backup directory... already exists", if not, create the "backup" subdirectory and print "Checking backup directory... backup directory created". Note: You can create a new directory using mkdir function.
   - If the input file exists and the file has more than 10 lines, give the user 2 choices:
     – Type ’c’ to copy first 10 lines of the file under "backup" folder with the same name, and then overwrite the existing file
     – Type ’o’ to proceed without creating a backup, and overwrite the existing file
   - If the input file exists and it has not more than 10 lines, backup everything under the "backup" folder with the same name, and overwrite the existing file. Note: Copying should be performed with Perl’s file read/write functions, rather than using the operating system’s file operations functionality
   - Otherwise, write the string to the input file normally. [10 pts]

Sample input and output:

```
$ perl q4_p1.pl
Enter file name:
randomFile.txt
Wrote to file randomFile.txt

$ perl q4_p1.pl
Enter file name:
randomFile.txt
randomFile.txt already exists. Checking backup directory... backup directory created
randomFile.txt has more than 10 lines. What to do next?
Enter ’c’ to backup the first 10 lines, ’o’ to overwrite without creating a backup
c
Ok, old file backed up under backup directory
Wrote to file randomFile.txt

$ perl q4_p1.pl
Enter file name:
randomFile.txt
randomFile.txt already exists. Checking backup directory... already exists
randomFile.txt has more than 10 lines. What to do next?
Enter ’c’ to backup the first 10 lines, ’o’ to overwrite without creating a backup
o
Wrote to file randomFile.txt

$ perl q4_p1.pl
Enter file name:
randomFile.txt
randomFile.txt already exists. Checking backup directory... already exists
randomFile.txt has no more than 10 lines.
Ok, old file backed up under backup directory
Wrote to file randomFile.txt
```

Part 2.

Write a program asking the user to input a string. Print the names and statuses (e: if exist, r: if readable, w: if writeable, x: if executable, T: if it is a text file) of all files in the current directory that contain the input string. If the substring appears multiple times in a file then print the file name only once. There's no need to check in subdirectories. Hint: readdir will be helpful: http://perldoc.perl.org/functions/readdir.html [10 pts]

Sample input and output:

```
$ perl q4_p2.pl
What string do you want to search for?
use warnings
Found "use warnings" in file q1_p1.pl............erwT
Found "use warnings" in file q1_p2.pl............erwT
```

5 Regular Expressions [25 pts + 5 pts bonus]

In this question, each subsection contains different types of regular expression questions.

5.1 Matching strings[4 pts each]

For each item below, write a regular expression that matches the strings on the left, while not matching the strings on the right. **Note:** The regular expression should match entire strings, not just the beginning/end of it.

•

very agile fluffy cat	cat
very very agile fluffy cat	agile cat
very very big cat	fluffy cat
very very fluffy cat	agile fluffy cat
very big cat	agile big cat

•

cat.	hello
dog?	world!!
computer.	hello!?
world!	perl?!

For submission, create a text file named "q5_p1.txt", and provide your regular expression solution for each item in a new line(*i.e.,* 2 lines in total).

5.2 More Regular Expression Questions[4 pts each]

- Write a regular expression that matches the following format: Month day, year(For example: "January 25, 2018" or "June 8, 1992", but not "June 54, 1992"). The regular expression you wrote should capture month, day and year information as separate groups. To keep things simple, you can assume there are 30 days in every month.
- Write a regular expression that matches a line in which every word is surrounded by spaces. An empty line always matches.

For submission, create a text file named "q5_p2.txt", and provide your regular expression solution for each section in a new line.

5.3 Finding examples and counter-examples[3 pts each]

For the regular expressions presented below, provide **three** strings that matches the regular expression, and **three** strings that do not match the regular expression.

- `[-+]?\d+(\.\d*)?F\b`
- `s/(#?)1|one/[\1January]/ig`
- `/((.*?)\d)\s\2/`

For submission, create a text file named "q5_p3.txt", and provide your examples by indicating whether it matches or not. This part will not be graded by a script, so you are allowed to use whichever format you want, as long as you make your answers clear.

5.4 Bonus: IP Matcher[5 pts]

Complete the IP-matching regular expression we have seen in the class by capturing all edge cases listed below(For IPv4 only, and all IP blocks can be from 0 to 255 inclusive)

- Each block in the IP address can have 1 to 3 characters
- IP addresses need to be between 0-255(Inclusive)
- Each block needs to be captured as a separate group(*i.e.,* encapsulated in parentheses)

For submission, create a text file named "q5_p4.txt", and provide your regular expression solution as a single line in this file.