Assignment - 1: Bash Scripting

Software Systems Development

Deadline: 11:55 pm, 14 September 2020

Submission Instructions:

- Name each submitted file as 'q<number>.sh'. For example, submitted version of script file for question 5 must be named as q5.sh.
- A README file for the assignment is mandatory (it will be given some points). The README must include brief explanation for each solution.
- Put all the scripts in a folder, name it as your 'RollNumber' and compress it as 'Rollnumber_Assignment1.zip'.
- Any/all questions/clarifications regarding Assignment 1 must be Via Moodle. Please use the Assignment 1 thread on the News Forum for the same.

Warning: **Plagiarism is strict NO! In all likelihood you will end up failing this course if you plagiarize on any assignments or activities in this class. Note that automatic plagiarism detection scripts will be run on all your submissions.**

QUESTIONS:

1. Following needs to be implemented in a single script file ‘q1.sh’:
    a. Create a directory ‘Assignment1’ and cd into it.
    b. Create 5 empty files: lab1.txt, lab2.txt, lab3.txt, lab4.txt, lab5.txt in a single command.
    c. Rename the above files to lab1.c, lab2.c, lab3.c, lab4.c, lab5.c in a single command.
    d. List the content of your current directory in long list format sorted in increasing order of file size.
    e. Display all files and folders inside **Home directory (~)** up to two levels of depth such that full path is displayed for each file/folder.
    f. Display all ‘.txt’ files inside directory ‘Assignment1’ such that full path is displayed for each file/folder.

2. Given a scrambled word (read input from user), check if it exists as an executable command. If yes, print ‘Yes’ followed by the command name in the next line. Else print ‘No’. If there are multiple valid commands for the scrambled word, you may print any one of them.

    Format:

    - Input =>
        - <Word>

    - Output =>
        - Yes/No
        - <Command_Name> (if yes)

    Example:

    ```
    ➜ bash q2.sh
    Input => tac
    Output =>
    Yes
    cat

    ➜ bash q2.sh
    Input => hy
    Output =>
    No
    ```

3. List the last 10 commands that have been used on your terminal in the descending order of usage. The result should contain command name and the count. If some commands have same count, print any order.

```
Format:

Input => None
Output => Let suppose you use cat 2 times, gedit 3 times, echo
5 times

➜ bash q3.sh
Output =>
echo 5
gedit 3
cat 2
```

4. Write a script that takes a list as input from user and flattens the data in the list. Any type of data can be inside list example doubles, numbers etc.

```
Example:

➜ bash q4.sh
(Input => Output)
Ex 1. (1 (2) ((3)) (5) (((7 8 9)) ()) => (1 2 3 5 7 8 9)
Ex 2. ((1 2)(3 (4))) => (1 2 3 4)
Ex 3. ((1 1.2)()(3.5 4)) => (1 1.2 3.5 4)
```

5. Take a string as input from user and check if the value is a palindrome. If there is some uppercase letter in the original string then it must be converted into lowercase before checking. Output Yes / No.

```
Example:

➜ bash q5.sh
(Input => Output)
1. madam => Yes
2. Madam => Yes (M is upper case but still palindrome)
3. max => No
```

6. Write a program that can take any number of **arguments** and performs an exponential value of that number. Note that here you don't take input from user, instead the numbers are provided in the command line itself via arguments.

```
➜ bash q6.sh 2 3 4
4096
Explanation => 2^3^4 => 8^4 => 4096

➜ bash q6.sh 2 3 4 5
1152921504606846976
```

7. Get process IDs of all running processes and save it in file 'pid.txt'. Now take an integer N as input from user and output the N smallest process IDs. If N is more than the total number of running processes then update N to that value.
   After the execution of the script, 'pid.txt' file should be present locally, containing all the running process IDs and stdout should contain N smallest of them.

```
Example: Let's say running PIDs are 9843, 4208, 2313 and 4398.

➜ bash q5.sh
Input => 2
Output =>
2313
4208
➜ cat pid.txt
9843
4208
2313
4398
```

8. Write a script that checks if Crontab is properly formatted. Crontab will be written in a file named 'crontab_file'. You have to take it as argument to the bash script and then output Yes/No.
   Format: Min HR Dom Mon Dow Command

```
Example:

-> cat crontab_file
30 08 10 06 * ls
-> bash q8.sh crontab_file
Yes
```

```
-> cat crontab_file
30 08 10 cat * ls
-> bash q8.sh crontab_file
No
```

No need to use ‘cat’ in the bash script, it’s just for reference.

9. Given a number (read from input) determine whether or not it is valid per the Luhn formula. The Luhn algorithm is a simple checksum formula used to validate a variety of identification numbers, such as credit card numbers and Canadian Social Insurance Numbers.
   https://www.geeksforgeeks.org/luhn-algorithm/
   The task is to check if a given string is valid or not. Output valid/invalid in respective cases.

   Strings of length 1 or less are not valid. Spaces are allowed in the input, but they should be stripped before checking. All other non-digit characters are disallowed.

   Example:

   Valid case

   ```
   4539 3195 0343 6467
   ```
   The first step of the Luhn algorithm is to double every second digit, starting from the right. We will be doubling
   ```
   4_3_ 3_9_ 0_4_ 6_6_
   ```
   If doubling the number results in a number greater than 9 then subtract 9 from the product. The results of our doubling:
   ```
   8569 6195 0383 3437
   ```
   Then sum all of the digits:
   ```
   8+5+6+9+6+1+9+5+0+3+8+3+3+4+3+7 = 80
   ```
   If the sum is evenly divisible by 10, then the number is valid. This number is valid!

   Invalid case

   ```
   8273 1232 7352 0569
   ```
   Double the second digits, starting from the right
   ```
   7253 2262 5312 0539
   ```
   Sum the digits
   ```
   7+2+5+3+2+2+6+2+5+3+1+2+0+5+3+9 = 57
   ```
   57 is not evenly divisible by 10, so this number is not valid.

```
➜ bash q9.sh
(Input => Output)
1. 4539 3195 0343 6467 => Valid
2. 8273 1232 7352 0569 => Invalid
```

10. Build a Basic calculator that takes operator and “n” operands for basic arithmetic operations. There will be 1 operation and multiple operands. Allowed Operations are +, -, *, /. All operations are from left to right. Use 4 precision for decimal numbers. n is always greater than 2. See examples for more clarity.

```
Format:

<Operation Type>
<Number of operands>
<Operand1>
<Operand2>
<Operand3>
...

If <Operation Type> is * and <Number of operands> is 3, then
above format implies <Operand1> * <Operand2> * <Operand3>.
Similarly others.

Example:

Input:
+
5
2
20
35
7
12
Output:
76

Input:
*
4
2
3
4
5
Output:
120

Input:
/
3
4
2
3
Output:
0.6667

Input:
-
3
4
2
7
Output:
-5
```

Guidelines regarding Input/Output format:

- There are two kind of input formats that you'll learn through this assignment, one is directly from the command line and other is taking input from the user. Each of them are specifically mentioned in the questions and you are required to follow it.

- You should not print anything like – "Please enter input:", "Input:", "Output:", "Program outputs …" etc. It is just for reference in the examples. Just take the input from command line/user and output whatever is being asked for, without anything extra.

- Take care of whitespaces, case (lower or upper), tabs, newlines while printing the output. Take reference from the examples, in case of any doubts.