CSE/ISE 337 – Scripting Languages
Spring, 2023
Assignment 03 – UNIX/Bash Shell Scripting

Assigned: Wednesday, 03/22/2023
Due: **Monday, 04/12/2023, at 11:59 PM**
Total Problems: 5, Total Points: 70

Submission Instructions:

1. Collect and submit all of your scripts in a single zip archive named: `a03_<Last_name><First_name>.zip`
2. The zip file should contain the following:
    - p1.sh, p2.sh, p3.sh, p4.sh, p5.sh
3. Make sure you include your name, netid, and Student ID number as comments at the top of each script file.
4. Make sure you use the specified file, function, and class names (and any other names specified in the instructions). Grading may be partially automated (by importing your code to run against our test cases) and the use of incorrect names will interfere.
5. This is an individual programming assignment. Any collaboration on coding will be considered a violation of academic honesty.

Problem 1 (15 points): `p1.sh`

Imagine that we have written numerous C files (.c files) in a directory D and compiled then to object files (.o files). Now, we want to ship directory D to a client who wants to run the source code. However, we cannot release our source code to protect intellectual property. Hence, we will have to move our source files to another directory and keep the object files in the same directory. However, we need to ensure the following when we are moving source files.

1. While moving files to the destination directory, the directory structure should be maintained.

For example, consider that we are moving all C files from the directory `project` to the directory `project_src_bkup`. Let us say that the directory project has C source files in `project/`, `project/subProj1`, `project/subProj1/subsubProj1`, and `project/subProj2`.

The destination directory `project_src_bkup` should have the corresponding source files in the same directory structure, that is, `project_src_bkup/`, `project_src_bkup/subProj1`, `project_src_bkup/subProj1/subsubProj1`, and `project_src_bkup/subProj2`.

2. If the destination directory already exists, then it should not be recreated. However, if it does not exist, then it should be created. Subdirectories in the destination directory may be re-created even if they exist.

3. If the source directory does not exist, then exit with status 0 and display a message: `"<src_dir> not found"`

Task:

Write a script to generalize and automate the process described above. Your script should take 3 inputs from the command line that specify the extension of the files to be moved, the source directory and the destination directory. The script will move all files with the specified extension from the source directory (and its sub-directories) to the destination directory (and matching sub-directories). Any temporary files created during script execution should be deleted after the script finishes execution.

Parsing the command-line arguments:

The 1st input will be the extension of the type of files you want to move out of the source directory to the destination directory. The 2nd input will be the sourced directory and the 3rd input should be the destination directory. If any of the inputs are missing or if more than the required inputs are provided, then the script should display the following message: "USAGE: p1.sh <extension> <src_dir> <dst_dir>".

Script name:

Name your shell script: `p1.sh`

Example invocation: `$ p1.sh .py old_project new_project`

This will take all the Python source files (with .py extensions) from the `old_project` directory and place them in the `new_project` directory.

Problem 02 (15): `p2.sh`

Imagine that we are given a data file, where each line is a sequence of numbers. The numbers in each line are separated by 1 of the following 3 characters—comma(,), semicolon(;) or colon(:). Further, there is no limit to the number of numbers in each line. All the numbers in a line are separated by the same delimiter, but different lines may use different delimiters. Here is an example of a sample data file:

```
1;2;3;4;5
11:4:23:12
18,4,17,13,21,19,25
```

As can be seen from the example, you could imagine a data file to be arranged in rows and columns where the columns are separated by either comma, semicolon, or colon.

Task:

Write a script that will take a data file (as described above) as input and will compute the sum of each column in the data file. The sum of each column should be written to an output file which will also be provided as input to the script. All inputs to the script will be provided from the command line. The output file should be written in the format: `Col <n> : sum`

That is, each line in the output file should have the column and its corresponding sum. For example, based on the data file shown above, the output file should look like the following (don't include the comments):

```
Col 1 : 30 # 1 + 11 + 18
Col 2 : 10 # 2 + 4 + 4
Col 3 : 43 # 3 + 23 + 17
Col 4 : 29 # 4 + 12 + 13
```

```
Col 5 : 26 # 5 + 21
Col 6 : 19 # 19
Col 7 : 25 # 25
```

Parsing the arguments:

If a non-existent data file is provided as input to the script, then the script should report:
"<filename> not found"
If the output file provided as input does not exist, then it should be created. If an output file provided as input exists, then it should be over-written with the new output of the script. You can assume that a data file will always have the format specified above. If either the data file or the output file path is not provided, then display a message:
"data file or output file not found"

Script name:

Name your shell script: `p2.sh`

Problem 03 (15): `p3.sh`

Imagine a scenario where you have the scores of each student in an exam and you are supposed to assign an appropriate letter grade based on the score. Further, assume that you have collected the scores of each student in a file. Each such file will contain the student ID and a breakdown of the scores for each question in the exam. Here is an example of a sample file for a student:

```
ID,Q1,Q2,Q3,Q4,Q5
102,9,9,9,10,10
```

Each student will have a corresponding file in the format shown above. You can assume that an exam will always have 5 questions worth 10 points each.

Task:

Write a script that will take a directory of score files as input, compute the score received by a student as a percentage, and display the student ID and her corresponding letter-grade in the console in the format:
ID:<letter-grade>
The table below shows the percentage range and the corresponding letter grade:

Percentage Score Range	Letter Grade
93 <= x <= 100	A
80 <= x < 93	B
65 <= x < 80	C
50 <= x < 65	D
< 50	F

Parsing the arguments:

The scores directory is the only argument to the script. If it is not provided, then display a message:
"score directory missing"

If the argument provided is not a valid directory, then display:
"<dirname> is not a directory"

Script name:

Name your shell script: `p4.sh`

Example:

Assume that the scores directory provided as input has the following files:

```
==> data/prob4-score1.txt <==
ID,Q1,Q2,Q3,Q4,Q5
101,8,6,9,4,10

==> data/prob4-score2.txt <==
ID,Q1,Q2,Q3,Q4,Q5
102,9,9,9,10,10

==> data/prob4-score3.txt <==
ID,Q1,Q2,Q3,Q4,Q5
103,5,6,2,4,6

==> data/prob4-score4.txt <==
ID,Q1,Q2,Q3,Q4,Q5
104,10,10,10,10,10

The script should display the following in the console:
101:C
102:A
103:F
104:A
```

Problem 04 (15): `p4.sh`

An automated spell checker is a piece of software that can automatically detect spelling mistakes in a file. Imagine that you must develop a specific kind of spell checker that can be used to verify the spelling of 5 letter words. Your spell checker will rely on a dictionary that contains all possible five-letter words.

Task:

Write a script that takes a target file as first input and a dictionary file as second input and displays all 5-letter words in the target file that have been spelled incorrectly in the console. Assume that the dictionary file is a newline separated list of all possible 5 letter words; that is, each word will be on a new line.

Parsing the arguments:

The 1st argument is the file whose spelling needs to be checked and the 2nd argument is a dictionary. If neither is provided, then display a message:
"input file and dictionary missing"
If an invalid filename is provided as 1st argument, then display a message:

"<filename> is not a file"
Do the same for the 2nd argument.

Script name:
Name your shell script: `p4.sh`

Example:
Assume that we want to check the spelling of 5-letter words in a file called `p4-sample.txt`. This file has the following text:

Python is an easy to laern, powerful programming language. It has efficient highlevel data structures and a simple but effective approach to object-oriented programming. Python's elegant syntax and dynamic typing, together with its interpreted nature, make it an ideal language for scripting and rapid application development in many araes on most platforms.
The Python interpreter and the extensive standard library are freely available in source or binary form for all mojar platforms from the Python Web site, https://www.python.org/, and may be freely distributed. The same site also contains distributions of and pointers to many free trhid party Python modules, programs and tools, and additional documentation.

In the above text the words 'laern', 'araes', 'mojar', and 'trhid' should have been spelled as 'learn', 'areas', 'major', and 'third'. Hence, on executing the script as follows:

```
$ ./p4.sh p4-sample.txt dictionary.txt
```

The script should write to the console:

```
laern
araes
mojar
trhid
```

Problem 5 (10): p5.sh

Task:
Write a script using simple shell commands. The script takes a command line argument that specifies a directory `dir`. The script first changes directory to `dir`, then prints the following in sequence:

(a) A line starting: "`Current date and time: `"
Then on the same line, the current time and date.
(b) A line starting: "`Current directory is: `"
Then, on the same line, the absolute pathname of the current working directory.
(c) An empty line
(d) The line: "`--- 10 most recently modified directories ---`"
(e) the 10 most recently modified subdirectories of the current directory in long listing format (most

recent first)
(f) An empty line
(g) The line: `"--- 6 largest files ---"`
(h) The 6 largest files in the current directory in long listing format (largest file last)
(i) An empty line, followed by a line of 70 equal signs

Note that you may assume that there are at least 6 files in an input directory. However, there may be any number of directories in the input directory.

Parsing the arguments:

The 1st (and only) command-line argument is the name of the target directory for the script. If the directory does not exist, then print the following message:
`"the directory does not exist"`
If the directory argument is missing, then print the following message:
`"input directory is missing"`

Script name:

Name your shell script: `p5.sh`