Homework 2

Start Assignment

Due Feb 3 by 11:59pm **Points** 100 **Submitting** a file upload
File Types sh and jpg **Available** Jan 27 at 12am - Feb 5 at 11:59pm 10 days

Submission Instructions:

1. For each problem, submit **1)** your program, and **2)** one or two screenshots in **jpg** format showing that your program really works.

2. Name your scripts using the pattern Problem#.sh , and name your screenshots using the pattern Problem#_Index.jpg. Problem# is the problem number (e.g., 1, 2, 3, 4, etc). Index is the index number showing the order of the screenshots (e.g., 1 or 2).

3. Submit individual files. **DO NOT SUBMIT A ZIP FILE**.

Problem 1:

Write a Bash script. The script takes a pathname as an argument and checks the file/directory corresponding to the pathname.

1) The script first reports whether it is a directory or a file corresponding to the pathname.

2) If it is a file, the script reports the size of the file and the size category (large, medium, or small) of the file. The category is determined as follows: If the file size is greater than 1MB (1048576B), the file is a large file; if the file size is less than or equal to 1MB (1048576B) and the file size is greater than 100KB (102400B), the file is a medium file; otherwise, it is a small file.

Combine a sequence of *if* statements on the file name and file size. To get the file size, use command *du -b* and command *cut*.

Testing: Try your script with the following input: */home*, */bin/ping*, */bin/grep*, and */bin/bash*. Your script should print out messages like "directory", "small file", "medium file", and "large file".

Problem 2:

Write a Bash script, which prints out the entries (including files and sub-directories) in a directory in a reversed order as what obtained using filename expansion. The pathname of the directory is provided as an argument. Assuming that with filename expansion you see /my/dir contains "cache cron games lib log run tmp," your program "reverse.sh /my/dir" will print "tmp run log lib games cron cache." You can print the names on one line, or each name on a line.

Use filename expansion, an array, and at least one function reverse(). Your script should be able to correctly handle the entries with spaces in their names. For example, if a file is named "bash scripting.ppt", make sure that your script does not break the name into two parts: "bash" and "scripting.ppt".

DO NOT use the built-in command "sort -r" or "ls -r".

Problem 3:

Write a Bash script, which sorts a list of command line parameters in ascending order. For example, your command will look something like:

$ insert-sort.sh 7 2 3 9 -1

and type enter.
Your program will return: -1 2 3 7 9

Use only basic commands and array. Do not use any built-in commands that sort array.

Problem 4:

Write a Bash script, which builds a table of counters for the commands under /bin in alphabetical order. For example, there are no commands starting with "a", 23 commands starting with "b", ..., and 10 commands starting with "z" (zcat zcmp zdiff zegrep zfgrep zforce zgrep zless zmore znew). Exclude commands starting with non-alphabets such as "[." Your script will print

```
a 0
b 23
...
...
...
z 10
```

Use loop to iterate through "a" - "z", and use an array to save the counts. After all the counts are obtained, print out the result.

Note that your script must report a number for every lower case letter from 'a' to 'z'. There are no requirements on reporting numbers for other characters (symbols, digits, or upper case letters).

Problem 5:

Write a Bash script that removes all zero length ordinary files in the directory (including those in the sub-directories at all levels, i.e., your program needs to traverse the directory tree) passed as an optional argument. If you do not specify the directory argument, the script uses the current working directory as the default argument.

This problem is for practicing bash programming skills. Though there is an easier way to achieve the goal with the *find* command, **the *find* command is not allowed to appear in your bash script.**

Testing: Run the script **create_dir.sh** **(https://njit.instructure.com/courses/21117/files/3245435/download?download_frd=1)** to create a directory named testdir. Then run your script against ./testdir. After that, run the following commands:

```
find testdir -name file?
find testdir -name file1?
find testdir -name file2?
find testdir -name file3?
```

You should not see any file names printed out by the first two commands. But you should find some file names printed out by the last two commands.