HOMEWORK - 1

# Problem 1: (1.5 Points)

For the first problem, let's keep things simple, the objective is to get you more familiar with bash syntax, by building a simple game that you may be already familiar with known as Rock, Paper, Scissors.

The bash script should meet the following requirements:

1. The script should allow two players to play, player 1 is the user and player 2 is the computer.
2. The script should greet the players when it starts
3. It should explain the rules of Rock Paper Scissors: 3a. Rock beats scissors (by crushing them). 3b. Scissors beat paper (by cutting it).

3c. paper beats rock (by covering it).

1. The script should allow the player to enter in their selection (use read command). You can assume the user will only type in rock, paper, or scissors (all lower-case).
2. It should print out the computer’s selection, which should be random (hint: you can use the RANDOM environment variable along with '%' to limit the range).
3. Should declare a winner, or a tie.
4. If there's a tie, the script should rematch until there's a winner

# Problem-2: (2.5 points)

Write a Bash script that performs the following steps:

1. Takes the name of a file or directory as an argument. If no argument is provided, prompt the user to enter the name.
2. Determines whether the input is a file or a directory.
3. If it's a directory:

3a. Check if the user has read and execute permissions on the directory. If not, display a warning message.

3b. If the user has the necessary permissions , count the number of files and subdirectories within it.

3c. If the user has the necessary permissions, display the list of all files and subdirectories along with their respective sizes.

1. If it's a file:

4a. Check if the user has read permission on the file. If not, display a warning message.

4b. If the user has read permission, determine the word count, line count, and character

count of the file. Based on the character count, categorize the file as follows: 4c-1. "Text Heavy" if the character count is greater than 10,000.

4c-2. "Moderately Sized" if the character count is between 1,000 and 10,000. 4c-3. “Light File" if the character count is less than 1,000.

# Problem-3: (2.5 points)

Write a bash script that performs the following steps:

1. Accepts a list of filenames (at least three) as command-line arguments. If no filenames are provided, exit the script with an error message.
2. Verify if each file exists. If a file does not exist, print a warning message and skip to the next file.
3. Create a timestamped directory (e.g., "backup\_YYYYMMDD") in the current directory to store the backups.
4. For each specified file:

4a. Copy the file into the backup directory.

4b. Rename the copied file in the backup directory to include a timestamp in its filename,

for example, "file.txt" should become "file.txt\_YYYYMMDD".

4c. After all files are processed, create a simple log file in the backup directory. This log should list the original filenames and their new names in the backup directory.

1. At the end of the script, display a message indicating the completion of the backup process and the location of the backup directory.

# Problem-4 (3.5 points)

Write a bash script that performs the following steps:

1. Prompt the user for a directory name, verify that the directory exists, and navigate into that directory. If the directory does not exist, allow the user to create it
2. List all the files and subdirectories in the directory and display a menu to the user to select a file or a subdirectory.
3. If a file is selected:

3a. Display the first ten lines of the file.

3b. Ask the user if they want to display more. If "yes", display the next ten lines. 3c. Continue this process until the user responds with "no" or the end of the file is reached.

1. If a subdirectory is selected:

4a. navigate into that subdirectory.

4b. Perform a recursive search to find all files that have been modified in the last 24 hours.

4c. Present a list of these files to the user and ask them to select one.

4d. For the selected file, repeat the file display process as described in step 3.

1. After the operation is completed for either a file or subdirectory, return to the main menu to allow the user to select another file or subdirectory, or to exit the script.

NOTE:

For the above problems, Though there is an easier way to achieve the goal with the using **find**

command, the find command is not allowed to appear in your bash scripts.