==========**READ BEFORE YOU START SCRIPTING**========

SETUP For the LAB

un-tar **lab4_solution.tar.gz** into a folder named after your roll number. You can use the following command for this:

```
mkdir    <roll_no_lab4>
tar   -xvzf    lab4_solution.tar.gz    -C    <roll_no_lab4>
```

Change permissions to this folder as follows:

```
chmod  -R 700  <roll_no_lab4>
```

After you are done, check it with script: **structure_check.sh** which is located inside the untarred folder.

If it gives status OK, then you can now proceed to do the assignments as follows:

Solving Instructions

1. Solution for each assignment is to be typed into the .sh file already kept inside the corresponding folder.

2. Remember, you need to change the permission of the .sh file before executing it:

   ```
   chmod u+x    <scriptfile>
   ```

1. Do not change the structure and the names.. As automatic checker will give you a “not-submitted” grade otherwise if it does not find the names.

2. Remember to follow the exact output formats specified. An extra characters of changes to it will lead to incorrect evaluation by automatic checker.

3. Inside the script if you use any temporary file, write command to remove it after the operation within that script only.

Submission Instructions

1. create **<roll_no_lab4>.tar.gz** file. Upload this file.

==========**PROBLEM STATEMENTs and their respective folder names** ========

Write a bash script to …

Script1

take a number N as argument, calculate sum of numbers from 1 to N. Print the sum. Don't print any other characters except the sum.

For example

```
>>./script1.sh 5
>>15
```

Script2

take a path as argument, count number of folders in that specified path and print only that number.

You may use the command **grep** to find directories.

For example if test is the directory on which the script is to be tested, and if test has 2 folders then

```
>>./script2.sh test/
>>2
```

Script3

take a filename as an argument, read the all the list of names in that file, and make as many folders with those names. Print message “created <name>” after each successful creation. Print "could not create <name>" if a folder could not be created.

For example: If file namelist contains:

```
test1
test2
test1
```

After running the script, output can be had as follows if the creation is successful for first two but not for the last one (since test1 already existed!):

```
>>./script3.sh namelist
>>created test1
>>created test2
>>could not create test1
```

Script4

take two file extensions as argument and convert all the files in current directory having first extension to second extension keeping the filename same. Sample files are provided inside test folder.

Example:
```
>>./script4.sh png eps
```
All the files having extension .png should be converted to .eps.

Script5

take a file extension as argument, find all the files with that extension and append today's date with the filename keeping the extension same. Format of the date should be dd-mm-yyyy. Don't hard code the date, it should work on any day.
Example:
```
>>./script5.sh jpg
```
A file like trek.jpg will be renamed as trek10-08-2016.jpg as today's date is 10-08-2016. If you run the script tomorrow, it should be renamed to trek11-08-2016.jpg

Script6

list all the files and folders of current directory in following manner. First line should be the heading. Dump the list in "ls_sort" file.

File name<tab>Time<tab>Permission

Script7

Copy the file created by the above script (i.e. ls_sort) into your current directory (i.e. script7). Write a script which will take a column number 1 or 2 or 3 as an argument and sort the file in reverse order of the given column. Print the sorted file back to the same file keeping the first line (heading) intact.

Script8

Merge the two tables tab1 and tab2 to another table tab3 sorted according to the column number given as the argument to the shell script.

Example:

```
>>./script8.sh  tab1 tab2 tab3 3
```

It creates a new file tab3 and merges the content of tab1 and tab 2 and sorted based on the 3rd column.

Script9

Count the no. of lines in all the files in the given directory and prints only the number.
Example: if **test** is the directory on which the script is to be tested and its contains 3 files.

```
>>./script9.sh testcase/
>>18
```

Script10:

count the total number of those lines where each line contain **atleast** one occurrence of ‘if’ in all the files in the current directory.

print only the number.

Note that if there are multiple occurrences of ‘if’ in a single line you need to count that as one.

Example: if **test** is the directory on which the script is to be tested and its contains 3 files where there are 2,2,1 lines in respective files which contains atleast one ‘if’.

```
>>./script10.sh testcase/
>>5
```

Script11

count the total number of lines where each line contain **atleast** one occurrence of ‘for’ in all the files in the current directory and through all the internal nested directories

print the number..

Note that if there are multiple occurrences of ‘for’ in a single line you need to count that as one.

Example:

```
>>./script11.sh test/
>>8
```