Assignment 2: Bash Scripting
IT Workshop - 1

- Deadline for submission is: **9:00PM, 9 October 2017**

- Submission of this assignment will be via a Git repository on **GitLab**. GitLab is similar to GitHub, except it allows you to create private repositories.

- Submission details:
  - Create an account on **gitlab.com** if you don't already have one.
  - Create a **private** repository on gitlab, so that other's can't see your code, with the name "**Assignment2-ITWS**".
  - Give the TA user **itws.ta** "**master**" access to this repository.
    How to: (https://stackoverflow.com/questions/31908222/giving-access-to-private-gitlab-repository)
  - In your repository, there should be at most 15 files:
    - Each question should have a script file with name format: q<n>.sh
      "q1.sh" for question 1, for example.
    - There should also be a readme file in your repository, with any additional information.
    - Do not add any other files or directories in the repository.

- This assignment will be evaluated automatically. Make sure you submit your scripts in the format specified, otherwise they will not be evaluated.

- Make sure to give the **TA user account access to your repository**. Otherwise, we will not be able to access it and evaluate your work, leading to a 0 in this assignment for you.

- **Plagiarism will earn you a straight ZERO in this assignment. Automatic plagiarism detection scripts will be run on all your submissions.**

Q1. Write a bash program which take two variables (i.e. warning level and critical level percentage) and check disk space left. Finally print the output as shown below:

```
$ bash answer.sh 75 90
OK, /dev/sda1, 40%
WARNING, /dev/sda2, 78%
CRITICAL, /dev/sda3, 99%

$ bash answer.sh 80 95
OK, /dev/sda1, 40%
OK, /dev/sda2, 78%
CRITICAL, /dev/sda3, 99%
```

Explanation:

All the disks with percentage between 75% and 90% are printed as WARNING, while greater than 90% CRITICAL.

Q2. Write a bash program which print all the directories present in current folder twice. Once sorted by size of folder, and once by files count.

```
$ bash answer.sh
folder1         20G
folder2         10G
folder3         1G

folder2         212 files
folder1         100 files
folder3         2 files
```

NOTE:

- Script should consider all the files recursively and not the files just immediately inside the folder.
- Usage of only “find” and “ls” command is allowed.

Q3. Write a bash program which take a string input, find all .txt files (in current directory) containing this string - along with count of lines string is found. It also exit with exit code as follows:

Exit Code	When
0	Success; Find at least one file.
1	Fail; No file containing the string.

```
$ cat 1.txt
This is 1st text file.
This is a sentence.
How are you?

$ cat 2.txt
```

```
This is 2nd text file.
I am learning bash scripting.
All the best.

$ bash answer.sh “This”
2 lines in 1.txt
1 lines in 2.txt

$ echo $?
0

$ bash answer.sh “3rd”
$ echo $?
1
```

Q4. Write a bash program which take file name as parameter, and print the file with counter on every new line as shown below.

```
$ cat file.txt
Welcome to IIIT-H.
My name is ITWS I.

This is an assignment.
Did you notice blank lines?

$ bash answer.sh file.txt
1) Welcome to IIIT-H.
2) My name is ITWS I.
3) This is an assignment.
4) Did you notice blank lines?
```

Q5. Write one line commands (using pipes) to use in following scenario:

```
$ ls
to_replace.txt
replace/

$ cat to_replace.txt
a b f g

$ ls replace
a.txt b.txt c.txt d.txt e.txt f.txt g.txt h.txt i.txt … x.txt y.txt z.txt

$ <Your Command Here>

$ ls replace
aa.txt bb.txt c.txt d.txt e.txt ff.txt gg.txt h.txt i.txt … x.txt y.txt z.txt
```

Explanation:

Delete all the files from **replace folder**, which are listed in **to_replace.txt**. For ex. **f** is present in **to_replace.txt**, hence we renamed **f.txt** inside replace folder to **ff.txt**

Q6. Ask the user for their monthly salary. Calculate whether they have to pay tax, and if so, how much is that amount. Print the result. (Tax is to be paid if annual salary is more than 3 lakh, and tax is 30% of total income)

Example:

If person doesnt have to pay tax, output “NO TAX PAYMENT REQUIRED”
Otherwise, output “TAX PAYMENT REQUIRED: XX”, where XX is the tax amount.

Q7. Write a script that will take a filename as an argument and adds execute permission to the file for the user, but only if the file is a regular file (Directories, device files, etc. not allowed). Name the script makeExec.sh

Your script must check to see that there is exactly one argument. If there are no arguments or more than one arguments, your script must produce a "usage" message that tells how to use the script. Your shell script must check to see if the file exists; if it doesn’t, give an error message.

Q8. Write a script that will do an ls -l command on each of its arguments only if the argument is the name of a regular file (Directories, device files, etc. not allowed). You should print nothing if the file does not exist or it’s not a regular file. Note that there can be multiple arguments to the script.

Example:

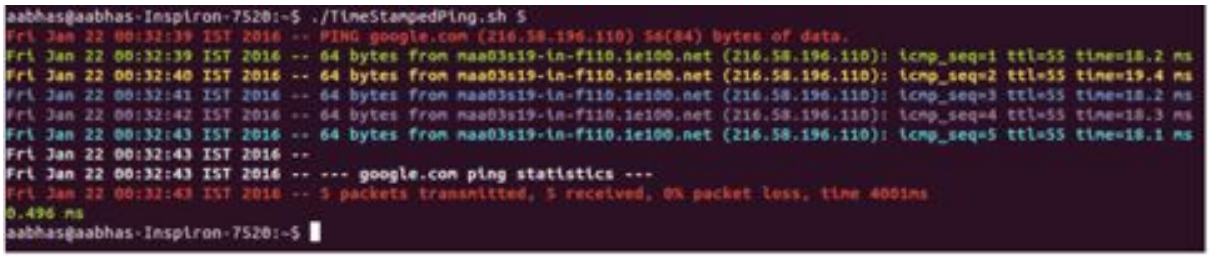
```
./myscript.sh abc file2 file4 testfile
```

Q9. Ask the user for their name and birthdate. If it’s their birthday today, output “Happy Birthday, ABC. You are XX years old today!”, where ABC is user’s name and XX is their age.

Q10. Write a script that outputs “X OUT OF Y USERS ARE ONLINE RIGHT NOW”, where X is number of users who are logged in on the local system, and Y is the number of total users on the system.

Q11. Print squares of numbers from 1 to 10. Then, print n^n for number n going from 1 to 10. Finally, print all the ‘odd’ fibonacci numbers <=100.

Q12. Write a shell script, time_ping.sh, to ping google.com n times only. Also, the output of ping should be timestamped. n will be given as a command line argument. The output at each consecutive lines should have a different color (make sure you use more than 5 different colours and consecutive colours are not same).



Here you have pinged google 5 times as given in the command-line argument and also each output line of ping has a timestamp with it.

For example, in the first line:

```
Fri Jan 22 00:32:39 IST 2016
```

Q13. Write a Bash script, roman.sh, to convert a number to Roman number. The function takes the input number as a command line argument. You can assume the range of the input to be 0-150, however bonus points will be awarded for having larger range.

The command to run will look like:

```
$ roman.sh 77
LXXVIII
```

Q14. Write a Bash script, num_sort.sh, that can sort a list of command line parameters in ascending order. There is no limit to the number of arguments passed as list parameters.

For example, your command will look something like:

```
$ num_sort.sh 8 27 9 -2 7 92 -9 0 and type enter.
-9 -2 0 7 8 9 27 92
```

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