CSC3320 System Level Programming
Lab Assignment 5 - In-Lab

Part 1:

Question 1) : What did you see in the output of step 3?

The output for step 3 is –

```
[vsrikakulapu1@gsuad.gsu.edu@snowball ~]$ simple.sh
-bash: simple.sh: command not found
```

Question 2) : What did you see in the output of step 4?

The output for step 4 is –

```
[vsrikakulapu1@gsuad.gsu.edu@snowball ~]$ ./simple.sh
-bash: ./simple.sh: Permission denied
```

Question 3): Attach a screenshot of the output in step 6.

The output for step 6 is –

```
[vsrikakulapu1@gsuad.gsu.edu@snowball ~]$ chmod a+x simple.sh
[vsrikakulapu1@gsuad.gsu.edu@snowball ~]$ ./simple.sh
Congratulations! Now you know shell script!
The current time and date are: Fri Feb 12 00:04:21 EST 2021
```

Question 4): Describe the meaning of -n option in echo command.

By default, echo command takes the cursor to the new line after printing out. To stop this default behavior -n option is used.

```
[vsrikakulapu1@gsuad.gsu.edu@snowball ~]$ echo -n hello
hello[vsrikakulapu1@gsuad.gsu.edu@snowball ~]$
```

Question 5): Is "Simple Script" a comment? If not, what is the meaning of it or why we use it?

Yes, “Simple Script” is a comment as # (pound) symbol is used in front of it, and also “Simple Script” is not printed out in the output which emphasizes that it’s a comment. # (pound) symbol in the first line of script or # (pound) symbol present in anything of kind #!/bin/shellType doesn’t make it a comment and this is not the case here, so, in this case # (pound) symbol represents a comment.

Question 6): Is "#!/bin/bash" a comment? If not, what is the meaning of it or why we use it in first line?

No, "#!/bin/bash" is not a comment as it is used in "#!/bin/bash” in the first line which means that the shell specified in it i.e., here bash shell is used to interpret this shell script ‘simple.sh’.

Part 2:

Question 7) : How many directories you can find in the output?
Note: the directories are separated by colon.

There are 6 directories in the output.

```
[vsrikakulapu1@gsuad.gsu.edu@snowball ~]$ echo $PATH
/usr/local/bin:/usr/bin:/usr/local/sbin:/usr/sbin:/home/vsrikakulapu1/.local/bin:/home/vsrikakulapu1/bin
```

Question 8) : Can you find errors prompted in step 9 ? If not, please briefly describe why there is no need to put ./ before the file name.

There are no errors even when ./ is used before the file name because ./ is used when current working directory is not present in the PATH variable. However, in step 8, we add current working directory to the PATH and so the need to use ./ before file name no longer exists and the script executes perfectly without giving any errors.

```
[vsrikakulapu1@gsuad.gsu.edu@snowball ~]$ PATH=.:$PATH
[vsrikakulapu1@gsuad.gsu.edu@snowball ~]$ echo $PATH
.:/usr/local/bin:/usr/bin:/usr/local/sbin:/usr/sbin:/home/vsrikakulapu1/.local/bin:
/home/vsrikakulapu1/bin
[vsrikakulapu1@gsuad.gsu.edu@snowball ~]$ simple.sh
Congratulations! Now you know shell script!
The current time and date are: Fri Feb 12 00:12:59 EST 2021
[vsrikakulapu1@gsuad.gsu.edu@snowball ~]$
```

Question 9: Can you find the current working directory . in the PATH variable?

After terminating and reopening of the terminal in step 10, the current working directory . is not found in the PATH variable.

Question 10) : Can you find errors prompted in step 11 ? If yes, please explain why?

Yes, after step 11, we get the error as command not found because to execute the simple.sh script without ./ in front, current working directory should be present in the path and after the termination and re-logging in to server, we don't see the current working directory in the PATH and so either we need to use ./ in front of simple.sh to execute the script or add current working directory to PATH using $PATH=.:$PATH to execute the script without errors.

```
[vsrikakulapu1@gsuad.gsu.edu@snowball ~]$ simple.sh
-bash: simple.sh: command not found
```

Part 3 - Optional:
Can you find some errors when executing the command in step 4? If yes, please point out which lines contain errors. Think about the correction in your next lab.

Lines 1, 2, 7, 20 contain errors as the errors specify the line number in which errors occurred.

Lines 1 - $#/bin/bash

In line 1, $ symbol is used unnecessarily and ! (exclamation mark) is missing after # symbol.

Lines 2 - /* Check Error Script */

The second line is /* Check Error Script */ which is comment syntax for java but not for shell scripting and is not recognizable by shell.

Lines 7 – grep '^[^a]*ce$' << END >> Result

In line 13 -ENDHERE is mentioned.

In line 17- semi-colon missing between ls and mail

Output in terminal is :

./checkError.sh: line 1: 1/bin/bash: No such file or directory

./checkError.sh: line 2: /1: Permission denied

Try to find out some errors!!!

./checkError.sh: line 20: warning: here-document at line 7 delimited by end-of-file (wanted `END')

Corrections:

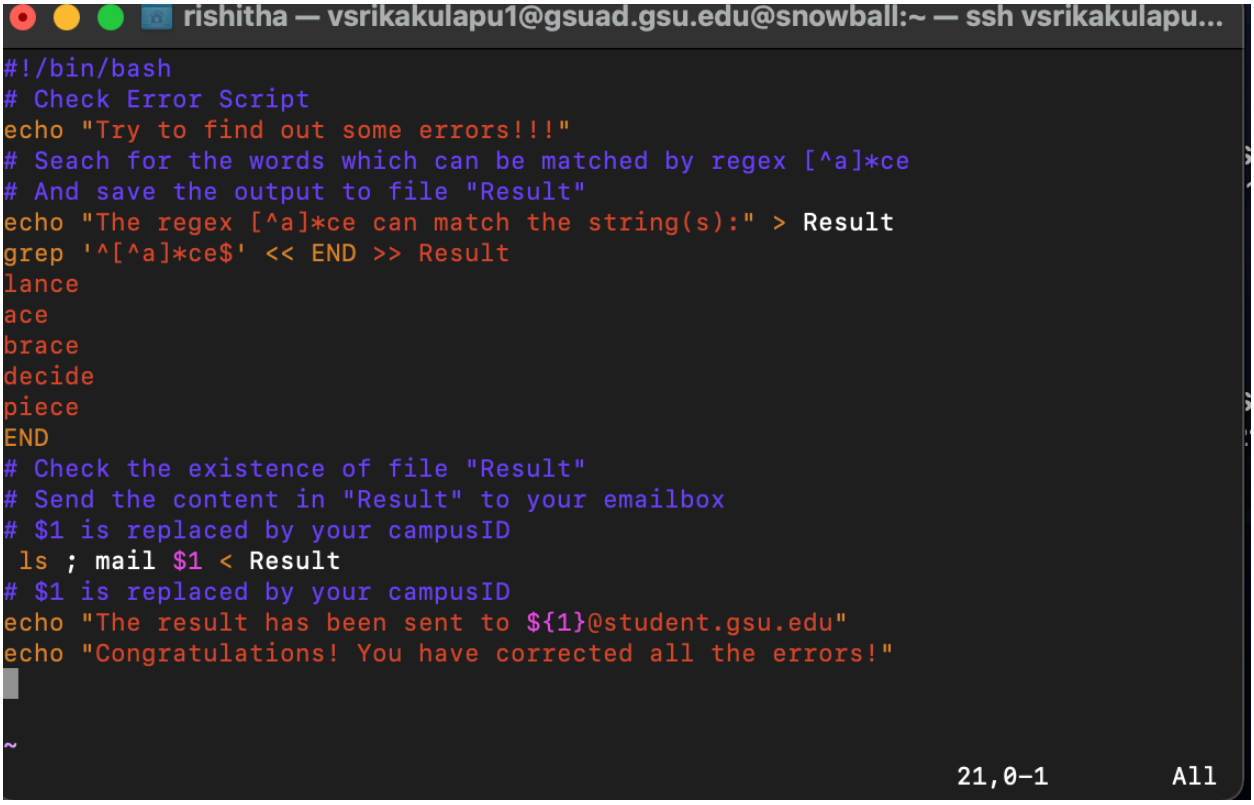
Lines 1 should be changed to - #!/bin/bash

Lines 2 should be changed to - # Check Error Script

Line 13 should be changed to - END

Line 17 should be changed to – ls ; mail $1 < Result.

Corrected file :



Output after correcting the file :
Here ls shows many files as my home directory contains all these files.

```
[vsrikakulapu1@gsuad.gsu.edu@snowball ~]$ vi checkError.sh
[vsrikakulapu1@gsuad.gsu.edu@snowball ~]$ ./checkError.sh vsrikakulapu1
Try to find out some errors!!!
checkError.sh  ft.txt                    homeworks   public     test
csc2720        h1.awk                    Lab2_2.txt  Result     test.txt
csc3320        h2.awk                    Lab3        simple.sh
float          homework_instructions.txt Lab4        s.txt
The result has been sent to vsrikakulapu1@student.gsu.edu
Congratulations! You have corrected all the errors!
```