COMP1562 Logbook (Week 3)

Basic Information

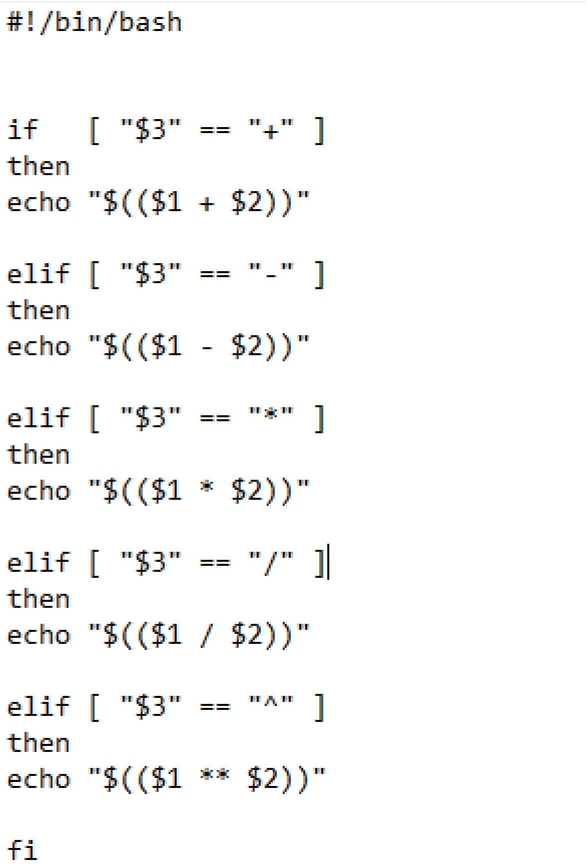
1.1 Student name	**Trevor Kiggundu (001001720)**
1.2 Who did you work with? Name and/or id	**Maruf Hoque (001006731)**
1.3 Which lab topic does this document relate to?	Shell Programming
1.4 How well do you feel you have done?	I have completed the exercise and am totally satisfied with my work.
1.5 Briefly explain your answer to question 1.4	My group and I were able to successfully follow and complete the tasks. Proof of that is shown below.

Annotated screenshots demonstrating what you have achieved:

TASK 3.1:

Figure 1 showing screenshot showing the code used to complete task 3.1:

-The value in $3 is the operator, and the value in both $1 and $2 are the user inputted numbers. The program uses a series of ‘if’ and ‘else-if’ statements to check which operator has been typed in. The ‘echo’ statement prints the desired operation.

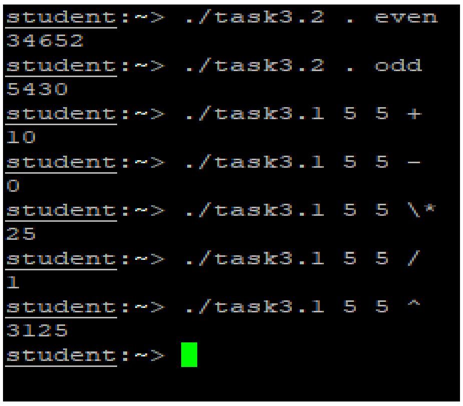


```
#!/bin/bash


if   [ "$3" == "+" ]
then
echo "$(($1 + $2))"

elif [ "$3" == "-" ]
then
echo "$(($1 - $2))"

elif [ "$3" == "*" ]
then
echo "$(($1 * $2))"

elif [ "$3" == "/" ]
then
echo "$(($1 / $2))"

elif [ "$3" == "^" ]
then
echo "$(($1 ** $2))"

fi
```

Figure 2 showing successful completion of task 3.1, which was to run the code to perform operations “+”, “-“, “*”, “/” “^”:

-The value in $1 is 5 and the value in $2 is 5. Expected results are as follows:

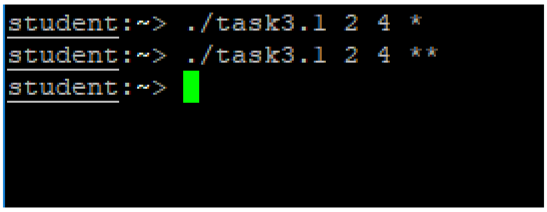
5+5 = 10
5-5 = 0
5x5 = 25
5/5 = 1
5^5 = 3125



```
student:~> ./task3.2 . even
34652
student:~> ./task3.2 . odd
5430
student:~> ./task3.1 5 5 +
10
student:~> ./task3.1 5 5 -
0
student:~> ./task3.1 5 5 \*
25
student:~> ./task3.1 5 5 /
1
student:~> ./task3.1 5 5 ^
3125
student:~>
```

Figure 3 showing possible errors that could be inputted:

-"*" and "**" are not operators so they will not display any answer

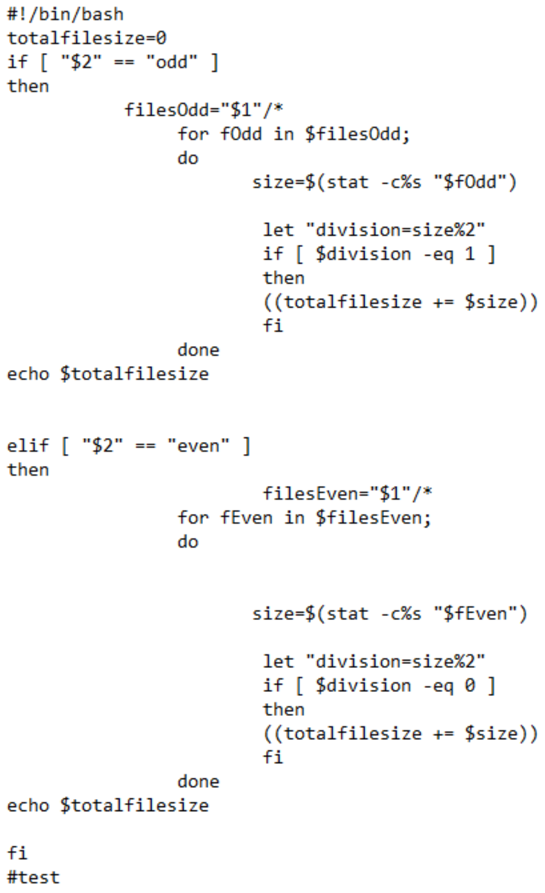


```
student:~> ./task3.1 2 4 *
student:~> ./task3.1 2 4 **
student:~>
```

TASK 3.2:

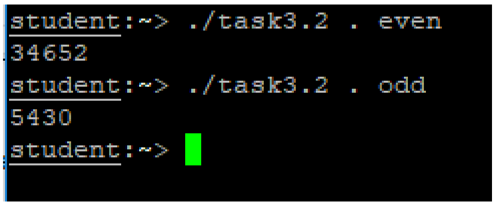
Figure 4: completed code

We used a for loop for task 3.2 to open up the files and get the file sizes. Then, by using modulus to work out the remainder, we can find out if the file size is a odd or even number. This allows the script to process the file as odd or even. The line of code "((totalfilesize += $size))" adds the current file size to the total file size which is represented by the variable $size.



```
#!/bin/bash
totalfilesize=0
if [ "$2" == "odd" ]
then
           filesOdd="$1"/*
                for fOdd in $filesOdd;
                do
                       size=$(stat -c%s "$fOdd")

                        let "division=size%2"
                        if [ $division -eq 1 ]
                        then
                        ((totalfilesize += $size))
                        fi
                done
echo $totalfilesize


elif [ "$2" == "even" ]
then
                        filesEven="$1"/*
                for fEven in $filesEven;
                do


                       size=$(stat -c%s "$fEven")

                        let "division=size%2"
                        if [ $division -eq 0 ]
                        then
                        ((totalfilesize += $size))
                        fi
                done
echo $totalfilesize

fi
#test
```

Figure 5: Script running in putty

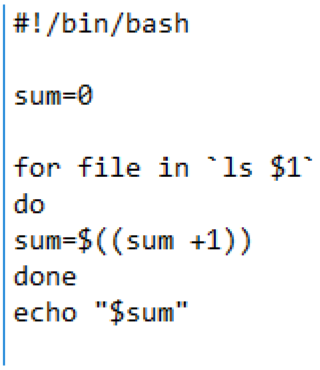


```
student:~> ./task3.2 . even
34652
student:~> ./task3.2 . odd
5430
student:~>
```

TASK 3.3:

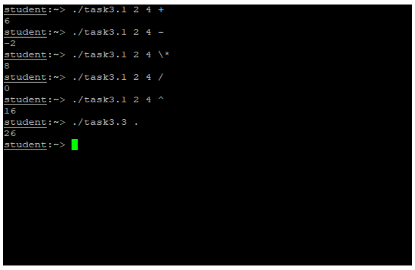
Figure 6: completed code

In this script, we set the register to 0. Then by using the command `ls $1` we can open up the folder to display the files. By using a for loop, we can add in increments of 1 for each file present. “Echo $sum” will display the total number of files in a directory.



```
#!/bin/bash

sum=0

for file in `ls $1`
do
sum=$((sum +1))
done
echo "$sum"
```

Figure 7: Script running in putty:



```
student:~> ./task3.1 2 4 +
6
student:~> ./task3.1 2 4 -
-2
student:~> ./task3.1 2 4 \*
8
student:~> ./task3.1 2 4 /
0
student:~> ./task3.1 2 4 ^
16
student:~> ./task3.3 .
26
student:~>
```

Personal Reflection:

This weeks' lab required us to use the Putty application once more to create and edit scripts just like last week. We were also more familiar with shell programming so the first task was not as difficult. We, however, ran into issues with the first task as we used the wrong operations in the Putty application, using "*" and "**" instead of "*" and "^" for multiplication and powers. These corrections and error tests are done and shown in figures 2 and 3. Regarding task 3.2, we ran into problems using script check initially after getting our code to run in Putty, as we did last week. After searching it up, we came to a conclusion that we were supposed fi for both task 3.1 and 3.2, as we were using if statements. We also added "#test" as a comment to make our code run in script check, as it was cutting off the last few lines of code when uploaded. Task 3.3 was fairly straight forward and did not give us any real problems that we hadn't previously encountered. I am glad that I experienced this lab session and 100% believe that is has made me a better programmer and given me a better understanding of system shells.