VIT®
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

EMBEDDED PROGRAMMING ECE4025 (L41+L42)

Allen Ben Philipose – 18BIS0043 – Task 6

Question 1:

10. Explain the shell script's debugging concept in detail

A script is simply a file containing a list of files of instructions/ commands. Instead of typing a series of commands, one by one, in the terminal, the system user can save all the commands in a file and the invoke the file to execute and re-execute the files multiple times.

Early script practice debugging is usually done by trial and error where we write the code, check the output to ensure they are working in the intended way, and if any bugs spotted, the programmer usually goes to that line and corrects the error manually.

However as the size of the scripts extend to thousands of commands, the scripts that change the system settings, perform vital backups, etc. we can discover that simply looking at the script's performance is not sufficient to find bugs. Trying to deduce the source of a problem based on performance alone, is insufficient for large shell scripts, and it is possible that by the time user gets the output, the script might have made incorrect and potentially destructive changes.

Fortunately, linux shell includes several built-in commands that allow various debugging nodes. When you need to add functionality to a big script that was written by someone else, the built-in debugging support can be extremely useful. It can ensure that the modifications don't affect the rest of the script

List of debugging options for shell scripts

i) No exec : -n

Primarily used for debugging syntax errors. Reads all commands, but does not execute them, so you have a safe way to test the scripts without altering any file.

ii) Verbose : -v

Displays all the lines as they are read by the executor.

iii) Execution trace (xtrace): -x

Displays commands and their arguments as they are executed. This is often called as shell tracing. We can see the values of variables and commands. In most cases this option provides the most useful info about a script

Methods of enabling shell script debugging mode

i) Invoking shell with debugging options

'$/bin/sh opt script arg1, arg2, arg3... argN' starts the shell with the debugging option defined by opt and tells it to run the script.

ii) Modifying the first line of the shell script

'#!/bin/sh' is used by UNIX to decide which shell you can use to run the script, implying that the script should be run using the /bin/sh shell.

'#!/bin/sh opt' is used to define a debugging option. These methods for activating debugging nodes are called as invocation activated debugging modes since they take effect when a script is invoked.

When more than one of the debugging options are selected, $- is assigned a letter that corresponds to that option.
eg. Letter 'v' is added to $- when the verbose option is used.

iii) Using set shell built in command:
set command

In invocation activated debugging modes, the debugging mode is activated at the start of the script and remains active till the end. Usually, you just need to debug one feature or a small portion of your script. Enabling debug mode for the entire script is unnecessary.

The debugging output can be very extensive and it is sometimes difficult to seperate the real errors from the noise. You can solve the noise problem by using the set command to allow debugging modes only in the sections of the script where you need them.

set opt : Can be used anywhere in the script and commonly used to change the debugging flags as part of normal execution. These are often referred to as programmer activated modes since they have to be manually set

set +opt :	Disable debugging mode
set +x :	Disable shell tracing
set - :	Disable any and all debugging modes that were enabled during the execution.

Question 2:

5. Write a shell script to find the factorial of the number

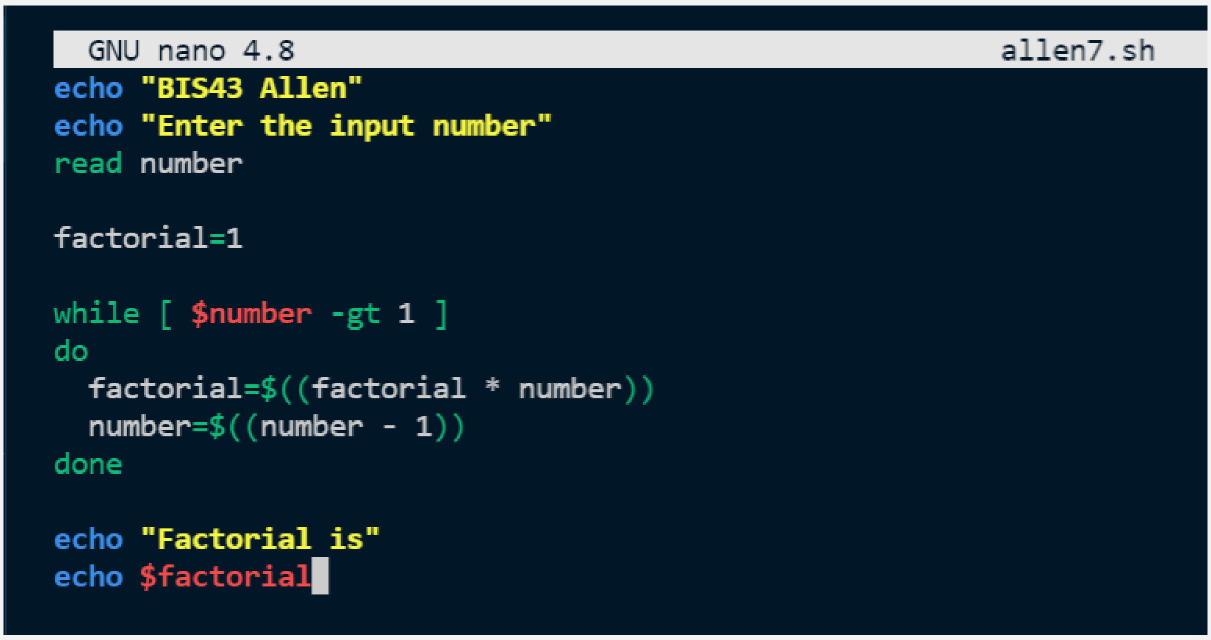
The program is possible by using an iterative process, such as a while loop, repeating a set of instructions multiple times, based on a condition given by the programmer.

For program improvement purpose, we will also add the code for user input and we will work on the user input to get the desired output

ans:

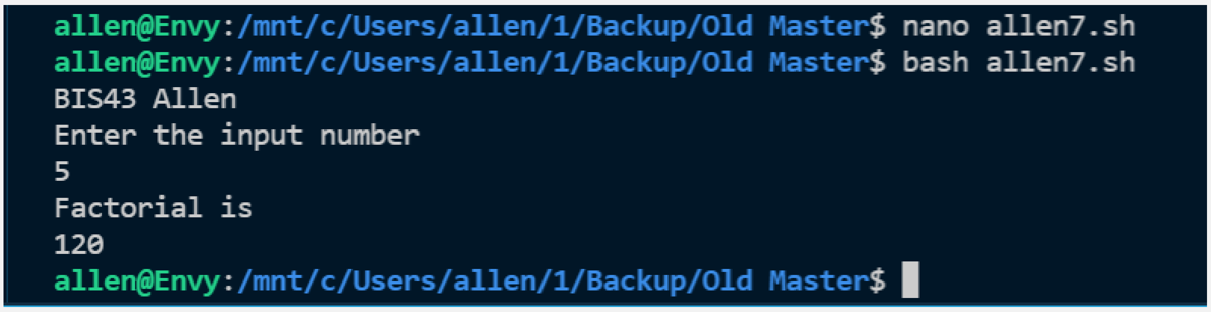
```
echo "BIS43 Allen"
echo " Enter the input number"
read number

factorial = 1

while [$number -gt 1]
do
    factorial = $((factorial * number))
    number = number - 1
done

echo "Factorial is "
echo $factorial
```

Nano editor – program saved in allen7.sh



```
  GNU nano 4.8                                      allen7.sh
echo "BIS43 Allen"
echo "Enter the input number"
read number

factorial=1

while [ $number -gt 1 ]
do
  factorial=$((factorial * number))
  number=$((number - 1))
done

echo "Factorial is"
echo $factorial
```

Output after executing bash command to run allen7.sh



```
allen@Envy:/mnt/c/Users/allen/1/Backup/Old Master$ nano allen7.sh
allen@Envy:/mnt/c/Users/allen/1/Backup/Old Master$ bash allen7.sh
BIS43 Allen
Enter the input number
5
Factorial is
120
allen@Envy:/mnt/c/Users/allen/1/Backup/Old Master$
```

Question 3:

Q4. Write a shell script command to find the smallest of the 3 numbers that are read from keyboard.

This can be achieved by multiple 'if' conditions which compares the variables which save the input received from the user.

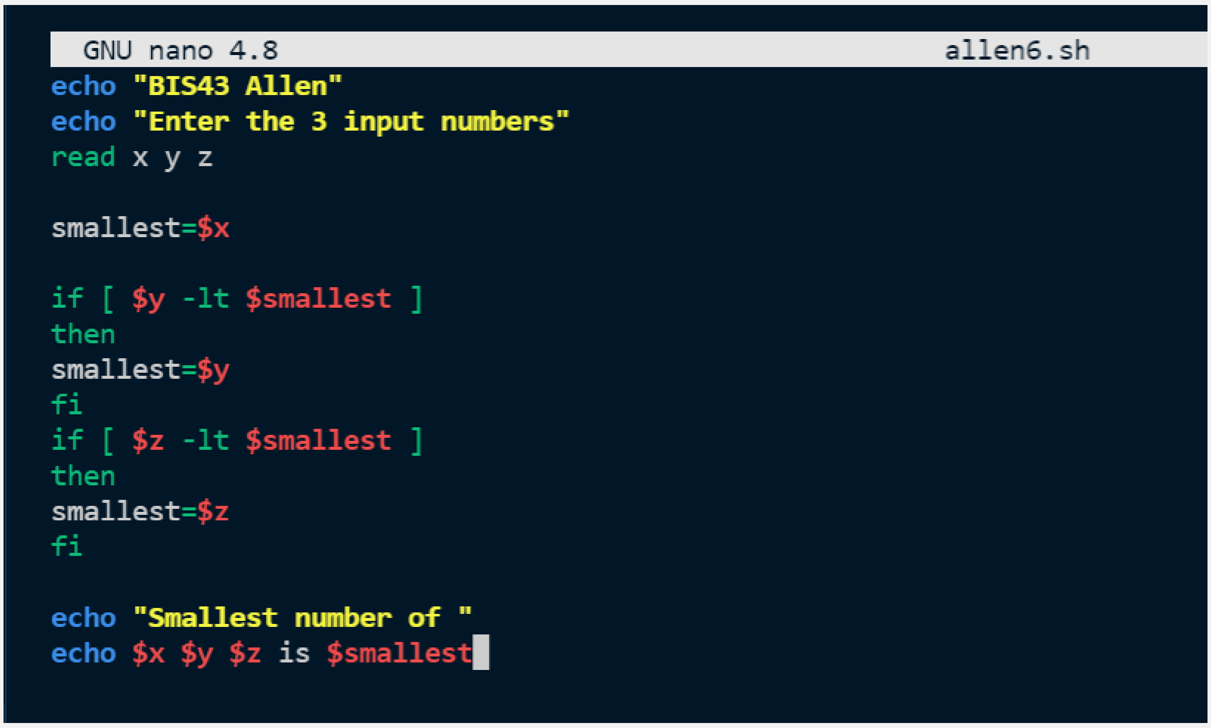
ans:

```
echo "BIS43 Allen"
echo " Enter the 3 input numbers "
read x y z

smallest = $x

if [$y -lt $smallest]
then
  smallest = $y
fi
if [$z -lt $smallest]
then
  smallest = $z
fi

echo "Smallest number of "

echo $x $y $z is $ smallest
```

First 'if' condition

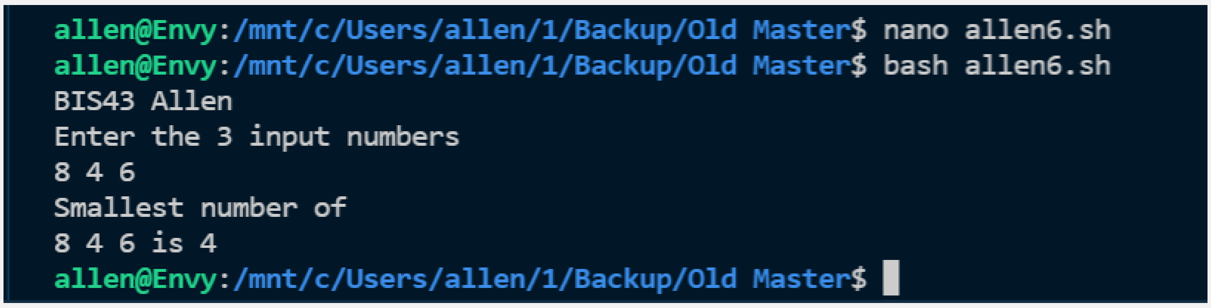
Second 'if' condition

Nano editor – program saved in allen6.sh



```
  GNU nano 4.8                                           allen6.sh
echo "BIS43 Allen"
echo "Enter the 3 input numbers"
read x y z

smallest=$x

if [ $y -lt $smallest ]
then
smallest=$y
fi
if [ $z -lt $smallest ]
then
smallest=$z
fi

echo "Smallest number of "
echo $x $y $z is $smallest
```

Output after executing bash command to run allen6.sh



```
allen@Envy:/mnt/c/Users/allen/1/Backup/Old Master$ nano allen6.sh
allen@Envy:/mnt/c/Users/allen/1/Backup/Old Master$ bash allen6.sh
BIS43 Allen
Enter the 3 input numbers
8 4 6
Smallest number of
8 4 6 is 4
allen@Envy:/mnt/c/Users/allen/1/Backup/Old Master$
```