Go through Webminal_ScriptingIntro.pdf Download Webminal_ScriptingIntro.
These are labs taken from Webminal.
Note: The solutions are given in the pdf above. But you must learn to do it yourself!

⇨ Webminal Scripting Intro:

a) Script of my name and course:

⇨ Here is the output:

```
19706@ip-172-26-2-101:~$ vi name.sh
19706@ip-172-26-2-101:~$ chmod +x  name.sh
19706@ip-172-26-2-101:~$ ./name.sh
Yunisha
I study in Computer Science.
```

⇨ Here is the script is:

```
19706@ip-172-26-2-101:~$ cat name.sh
#!/bin/bash

echo "Yunisha"
echo "I study in Computer Science."
19706@ip-172-26-2-101:~$
```

b) The script in sha-bang shell:

⇨

```
19706@ip-172-26-2-101:~$ cat name.sh
#!/bin/bash

echo "Yunisha"
echo "I study in Computer Science."
19706@ip-172-26-2-101:~$
```

c) The script in Korn shell:

⇨

```
19706@ip-172-26-2-101:~$ cat name.sh
#!/bin/ksh

echo "Yunisha"
echo "I study in Computer Science."
19706@ip-172-26-2-101:~$
```

d) The script with a two-variable:

⇨

```
19706@ip-172-26-2-101:~$ vi variable.sh
19706@ip-172-26-2-101:~$ chmod +x variable.sh
19706@ip-172-26-2-101:~$ ./variable.sh
Variable 1: 7
Variable 2: Hey!
19706@ip-172-26-2-101:~$
```

e) The previous script does not influence your current shell (the variables do not exist outside of the script). Now run the script so that it influences your current shell.

⇨

```
19706@ip-172-26-2-101:~$ source variable.sh
Variable 1: 7
Variable 2: Hey!
19706@ip-172-26-2-101:~$ echo $variable_1
7
19706@ip-172-26-2-101:~$
```

f) Is there a shorter way to source the script?

⇨

```
19706@ip-172-26-2-101:~$ . ./variable.sh
Variable 1: 7
Variable 2: Hey!
19706@ip-172-26-2-101:~$ echo $variable_2
Hey!
19706@ip-172-26-2-101:~$
```

g) Comment on your scripts so that you know what they are doing.

⇨

```
19706@ip-172-26-2-101:~$ cat variable.sh
#!/bin/bash

# Define the variables
variable_1="7"
variable_2="Hey!"

# Output their values
echo "Variable 1: $variable_1"
echo "Variable 2: $variable_2"
19706@ip-172-26-2-101:~$
```

⇨ <u>Webminal Scripting Inputs</u>

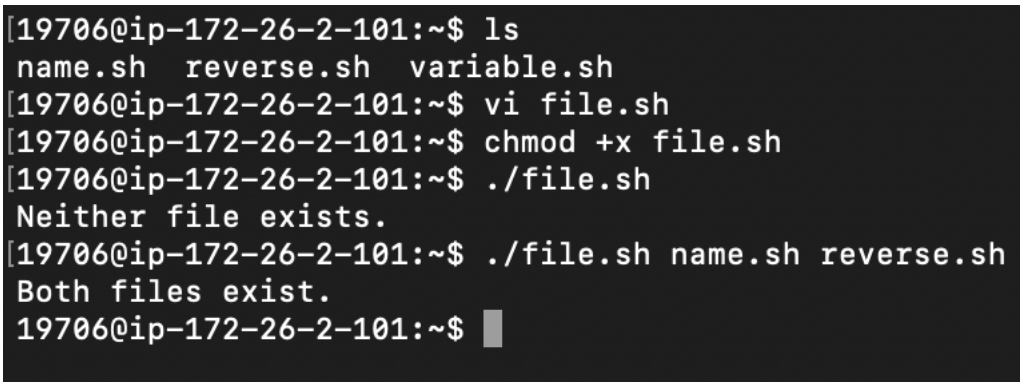
a) Write a script that receives four parameters, and outputs in reverse order.

⇨

```
19706@ip-172-26-2-101:~$ vi reverse.sh
19706@ip-172-26-2-101:~$ chmod +x reverse.sh
19706@ip-172-26-2-101:~$ ./reverse.sh
Please enter four parameters:
i love my life
life my love i
19706@ip-172-26-2-101:~$
```

b) Write a script that receives two parameters (two filenames) and outputs whether those files.

⇨




Question no.2
Create a makefile that has 3 options:
make type1, make type2, or make random.

The greeting statements are:
Type 1: "Hello World! How are you doing today? How may I help you?"
Type 2: "I am in a bad mood today. What do you want? Speak now or forever hold your peace."

- If you type make random, one of these 2 greetings would be printed out
Type 1: "Hello World! How are you doing today? How may I help you?"

Type 2: "I am in a bad mood today. What do you want? Speak now or forever hold your peace.”

- The greeting statements that are printed out depend on a random number generator.
- If the number generated is odd, a type 1 greeting is generated. If the number is even type 2 greeting is generated.

Also, for each of the targets, please use the touch command to create the file named type1, type2, and random respectively.

Now use the same make file as above.

⇨ Here is the output:

```
-sh-4.2$ make type1
Hello world! How are you doing today? How may I help you?
-sh-4.2$ make type2
I'm in a bad mood today. What do you want? Speak now or forever hold your peace.
-sh-4.2$ make random
I'm in a bad mood today. What do you want? Speak now or forever hold your peace.
-sh-4.2$ make random
make: `random' is up to date.
-sh-4.2$ ls
Makefile  random  type1  type2
-sh-4.2$
```

⇨ Here is the code:

```
-sh-4.2$ cat Makefile
type1:
        @echo "Hello world! How are you doing today? How may I help you?"
        @touch type1
type2:
        @echo "I'm in a bad mood today. What do you want? Speak now or forever hold your peace."
        @touch type2
random:
        @if [ $$((RANDOM % 2)) -eq 0 ]; then \
           echo "I'm in a bad mood today. What do you want? Speak now or forever hold your peace.";\
        else \
           echo "Hello world! How are you doing today? How may I help you?";\
        fi
        @touch random
-sh-4.2$
```

⇨ The $$((RANDOM % 2)) expression generates a random number between 0 and 1. If the number is O(even), we display the type2 greeting message, else we display the type greeting message if the number is odd (1).

In each of the targets, the touch command is used to create files named type1, type2, and random respectively.

Create another target called complete.

When complete is being made, it prints out, "that's the end!"

It is dependent on the target random, meaning that complete is only created after random is done.

⇨ Here is the output:

```
-sh-4.2$ make complete
I'm in a bad mood today. What do you want? Speak now or forever hold your peace.
That's the end!
-sh-4.2$ ls
complete  Makefile  random
-sh-4.2$
```

⇨ Here is the code:

```
-sh-4.2$ cat Makefile
type1:
	@echo "Hello world! How are you doing today? How may I help you?"
	@touch type1
type2:
	@echo "I'm in a bad mood today. What do you want? Speak now or forever hold your peace."
	@touch type2
random:
	@if [ $$((RANDOM % 2)) -eq 0 ]; then \
	   echo "I'm in a bad mood today. What do you want? Speak now or forever hold your peace.";\
	else \
	   echo "Hello world! How are you doing today? How may I help you?";\
	fi
	@touch random
complete:random
	@echo "That's the end!"
	@touch complete
-sh-4.2$
```