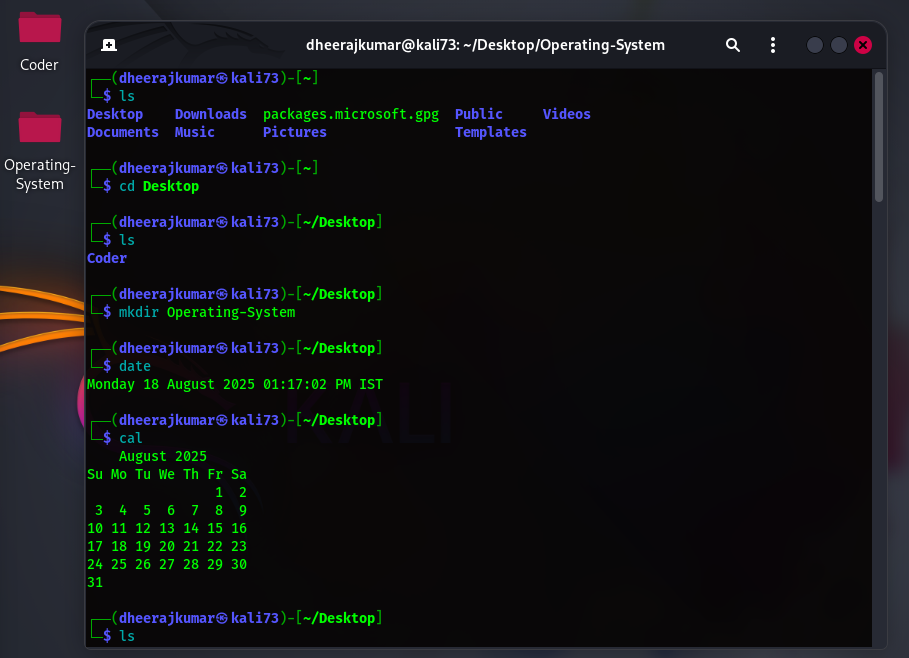
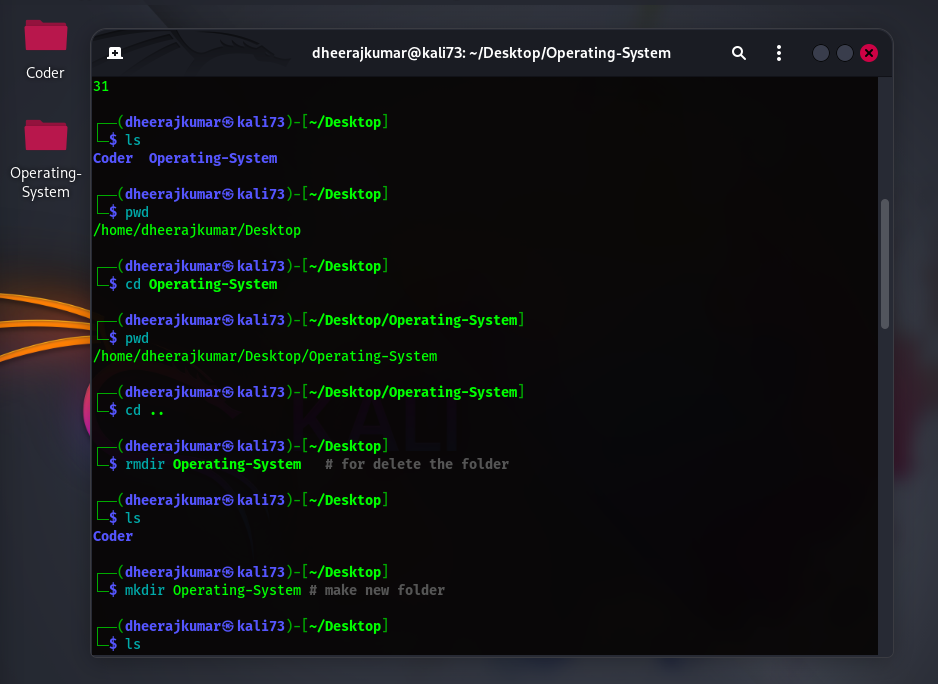
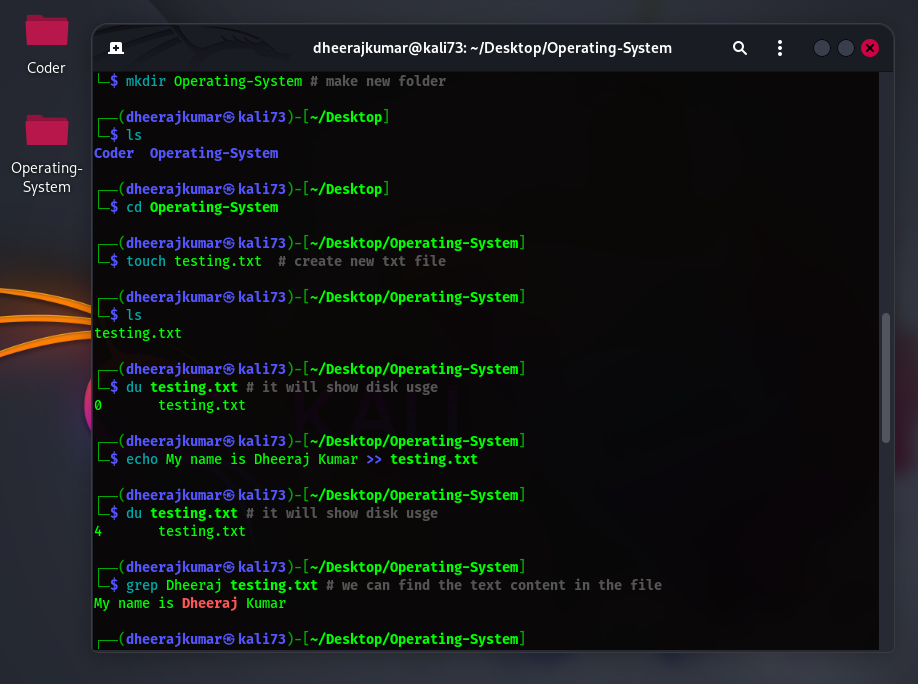
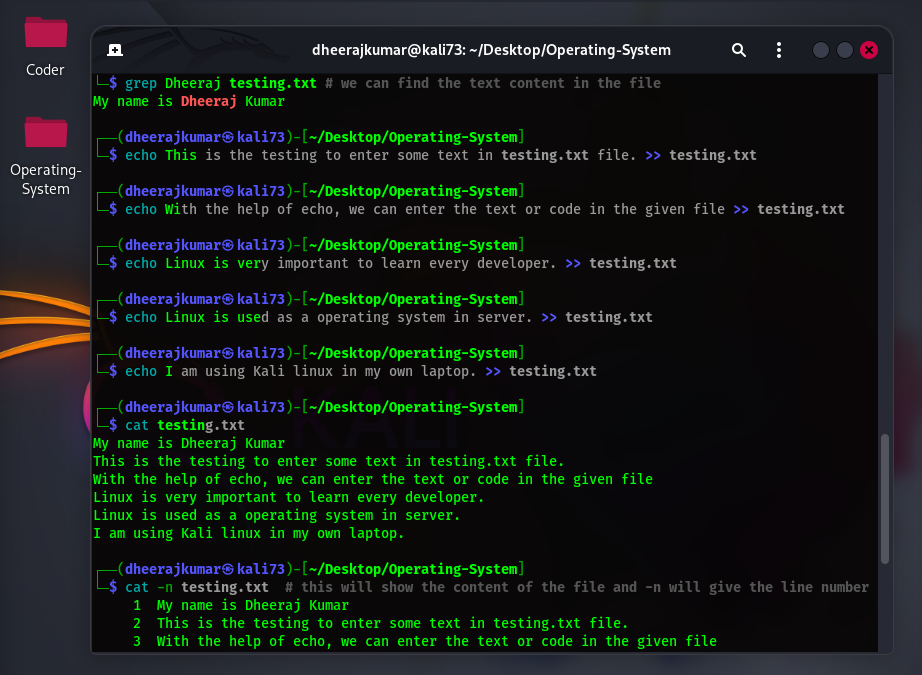
**Operating System (MCA)**

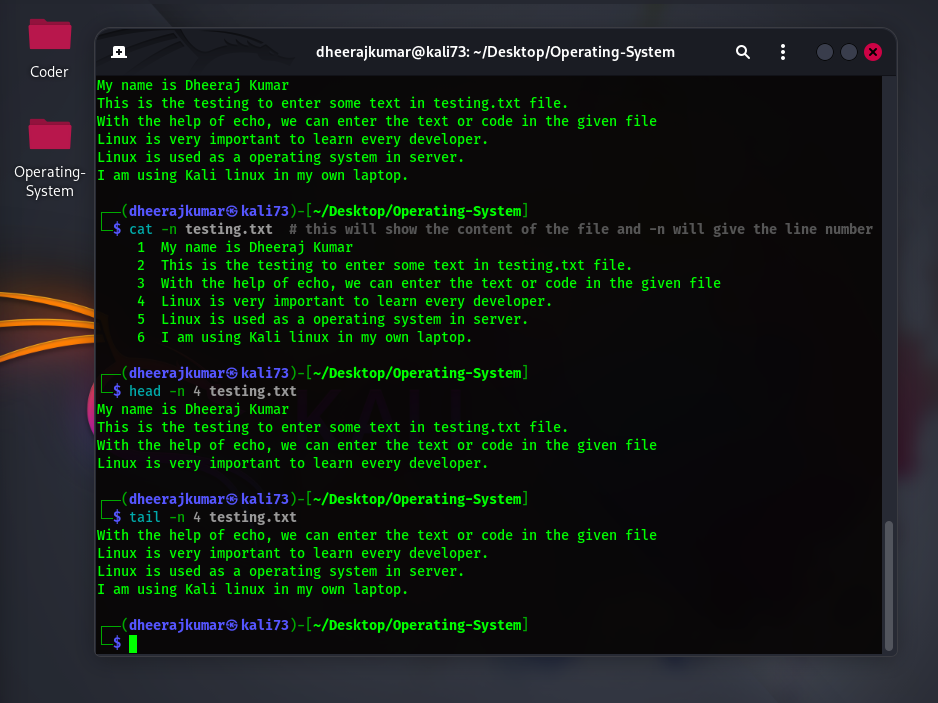
**Date: - 11/08/2025**











**Commands**

* **ls: -** List all the file and folder inside the path.
* **cd: -** Change directory.
* **cd .. : -** Change directory one step back
* **cd ~ :**- the command cd ~ is used to change the current working directory to the user's home directory. If you are currently in /usr/local/bin and you type cd ~, you will be returned to your home directory, which might be something like /home/your\_username**.**
* **date: -** It will show the current date.
* **cal: -** It will show the calendar.
* **pwd: -** Present working directory.
* **mkdir: -** Make directory.
* **touch: -** Make file any of the type. For Example if you want to create testing.txt then write touch testing.txt
* **rmdir:** - Remove directory.
* **rm: -** Remove file
* **du:** - The du command in Linux is a powerful utility used to estimate and display the disk space usage of files and directories. It stands for "disk usage".
* **echo: -** The echo command is used to display a line of text. You can redirect its output to a file using the > or >> operators.

**echo "This text will overwrite the file." > filename.txt**

* **cat: -** The cat command in Linux is a versatile and fundamental command-line utility used for handling files. Its name, "cat", is short for "concatenate", reflecting its original purpose of combining files.

**Useful options**

The cat command offers several options to customize its behavior:

* -n or --number: Numbers all output lines.
* -b or --number-nonblank: Numbers only non-empty output lines.
* -s or --squeeze-blank: Suppresses multiple adjacent blank lines, replacing them with a single blank line.
* -E or --show-ends: Displays a dollar sign ($) at the end of each line, indicating line endings.
* -T or --show-tabs: Displays tab characters as ^I.
* -A or --show-all: Combines the functionality of -v, -E, and -T.
* -v or --show-nonprinting: Displays non-printing characters (except tabs and newlines) using ^ and M- notation
* **head: -** The head command in Linux is a command-line utility used to display the first few lines of one or more text files. It's useful for quickly inspecting the contents of a large file or the output of a command.

head -n 5 filename.txt # Displays the first 5 lines

head -c 20 filename.txt # Displays the first 20 bytes

**Useful options**

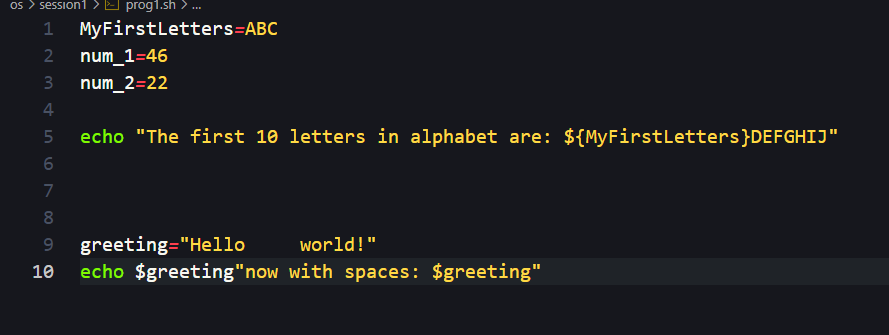
* -n <number>: Specifies the number of lines.
* -c <number>: Specifies the number of bytes.
* -q or --quiet: Suppresses the filename header for multiple files.
* -v or --verbose: Always displays the filename header.
* **tail: -** The tail command in Linux is a command-line utility used to display the last part of a file or files. By default, it shows the last 10 lines of the input. It's essentially the inverse of the head command, which shows the beginning lines of a file.

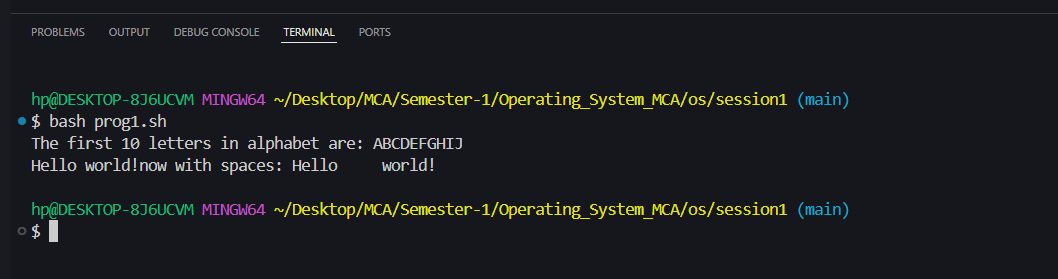
tail filename.txt

This command outputs the last 10 lines of filename.txt to your terminal.

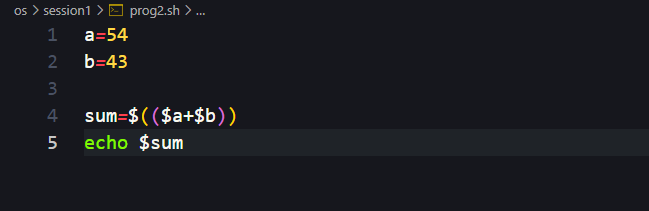
* tail -n 4 filename.txt: - This will show bottom 4 line of content.

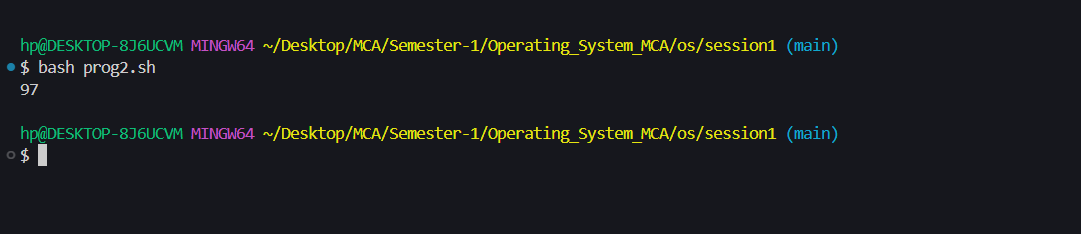
# Q1. Write sell script for creating a variable.



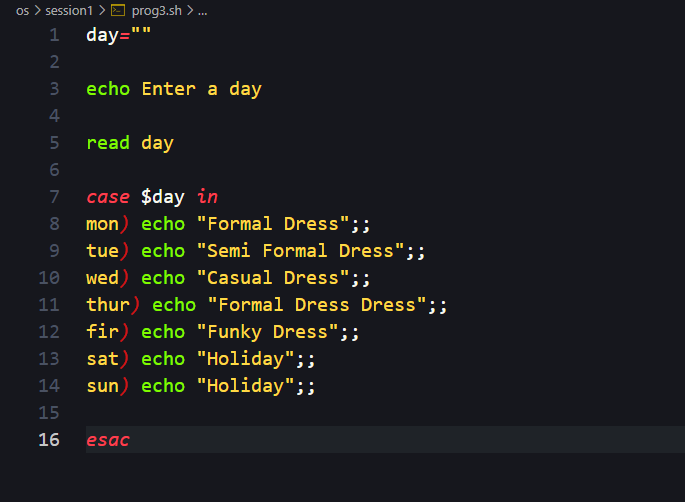


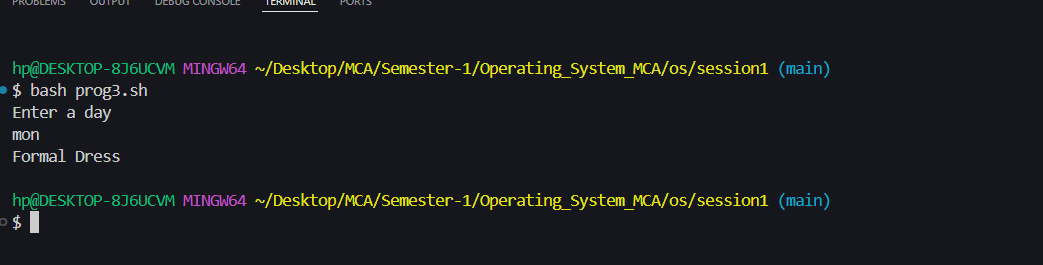
# Q2. Write shell script to add two number.



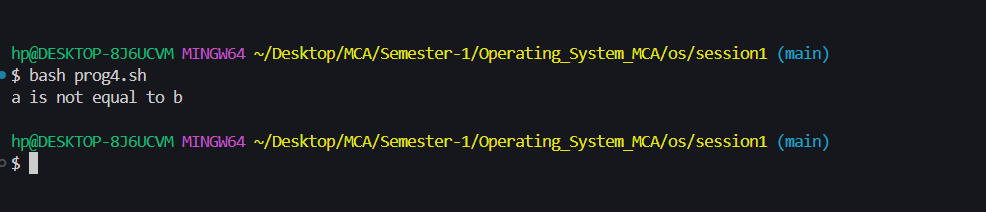
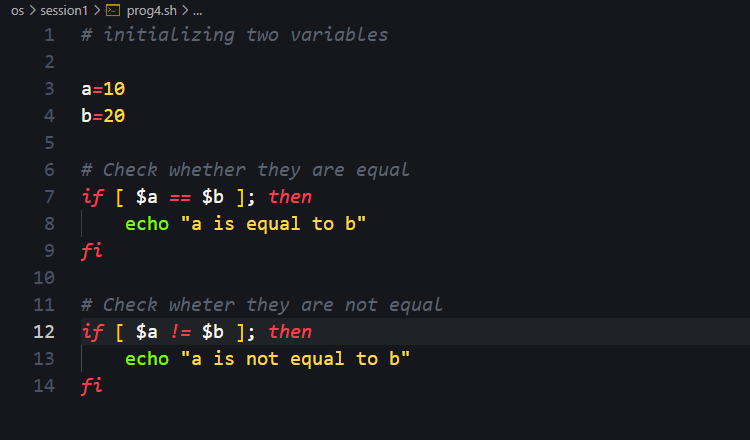


# Q3. Write shell script to demonstrate the case structure.

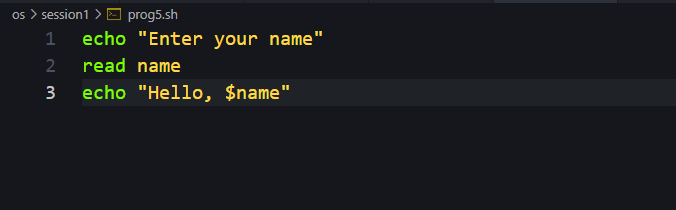


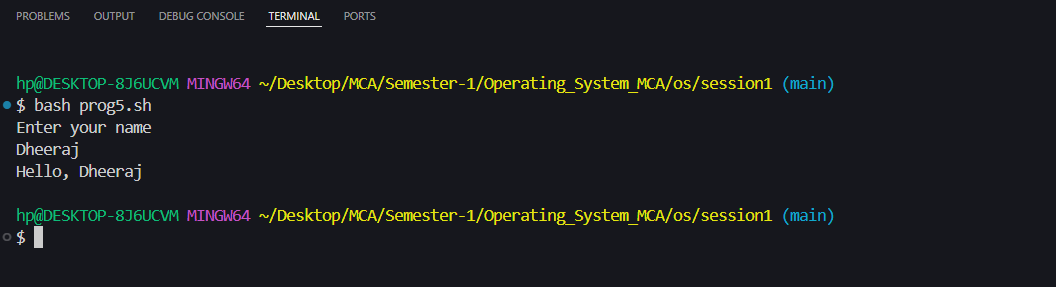


# Q4. Write shell script to check two number are equal or not.

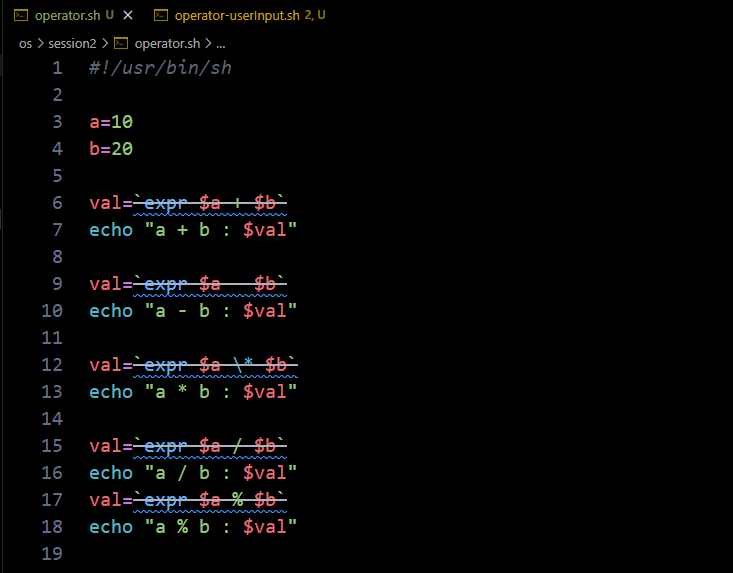


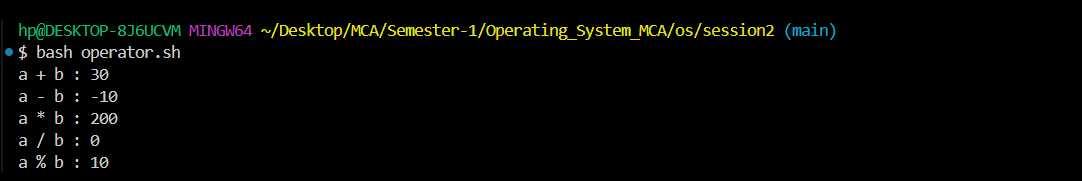
# Q5. Write shell script to take user input and print.



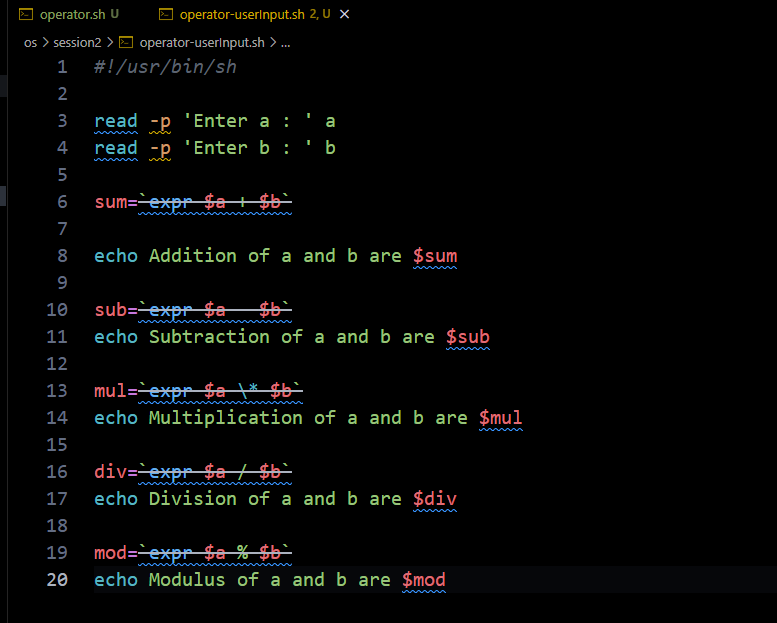


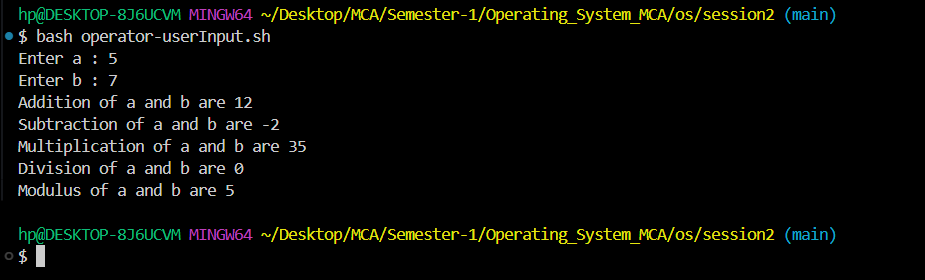
## Q. Write a bash script to perform Arithmetic operation.



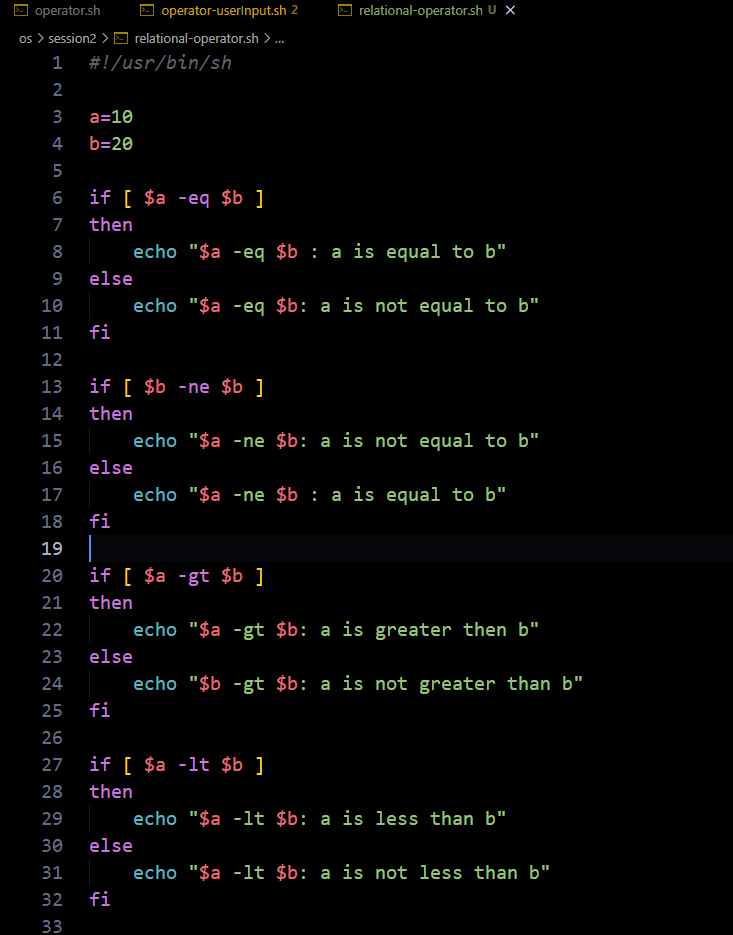


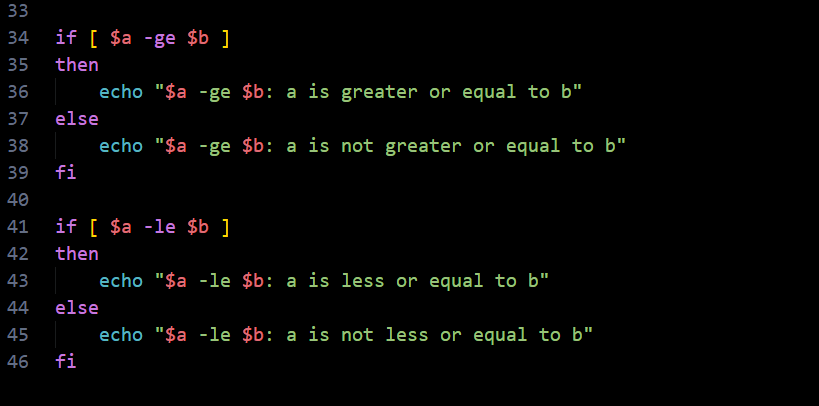
## Q. Write a bash script program to perform Arithmetic operation with user input.

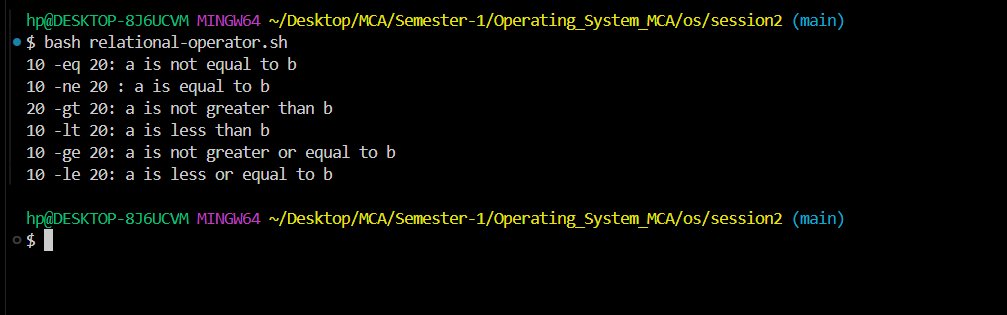




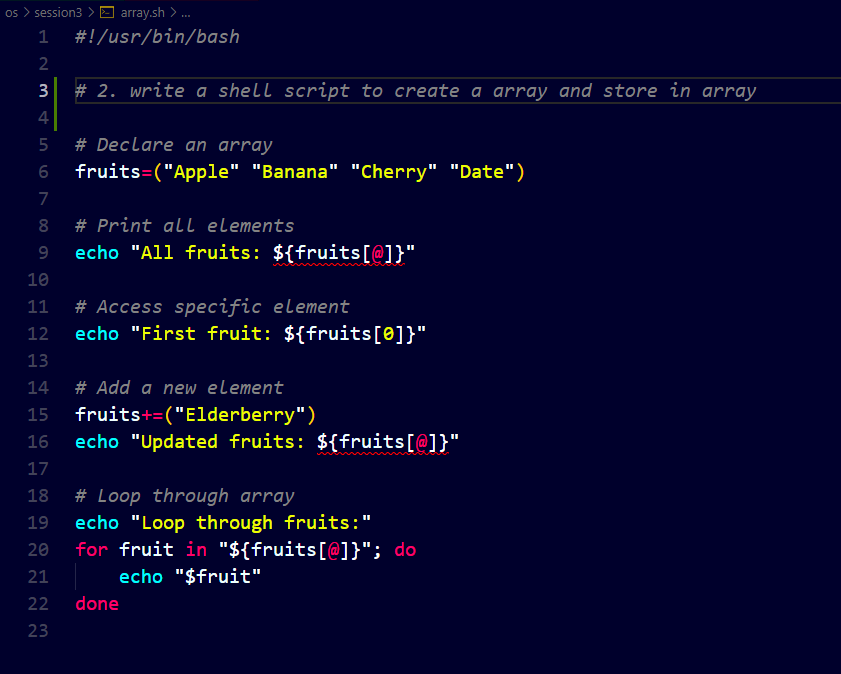
## Q. Write bash script program to perform relational operator

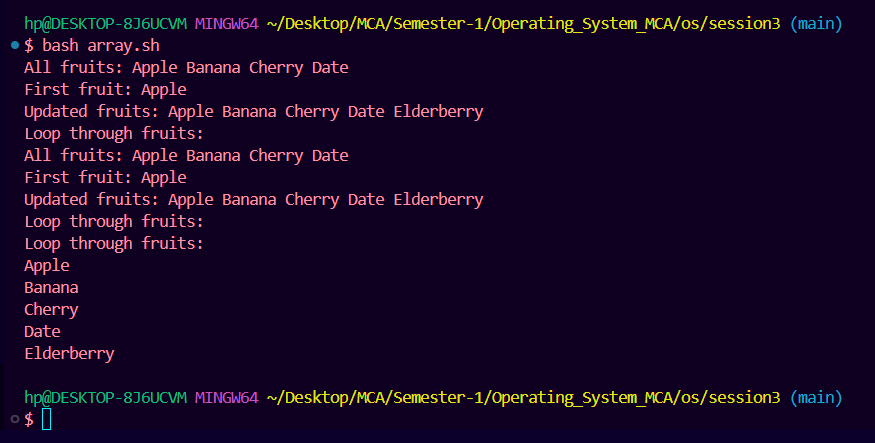




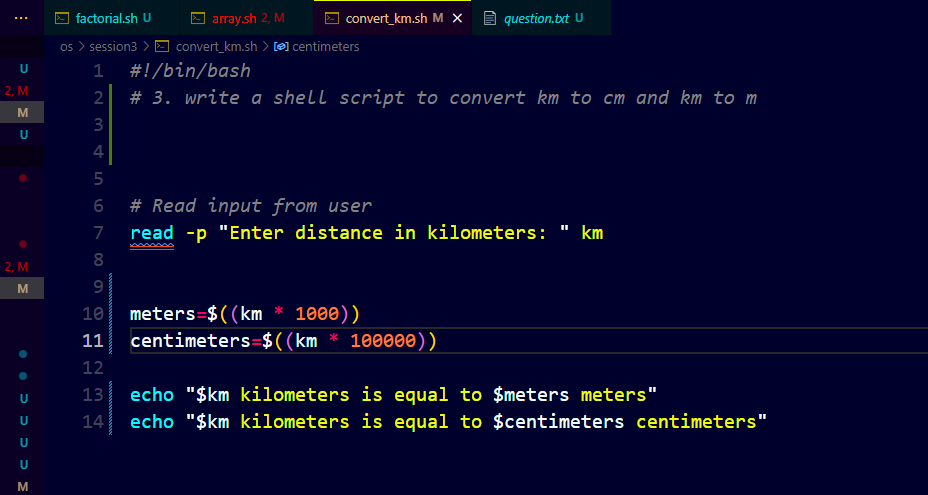


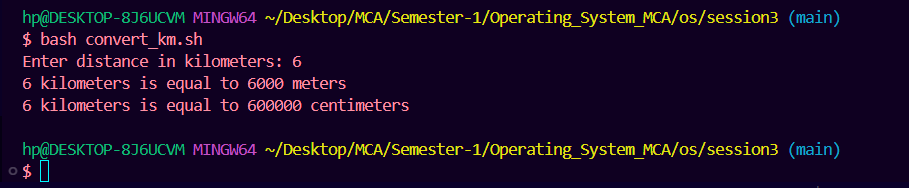
## Q2. write a shell script to create a array and store in array.





## Q3. write a shell script to convert km to cm and km to m





Assignment Question

Module 2

1. What are the premitive and non-premitive processes.

2. Explain with the help of example how sortest job first algorithm is better then first come first server algorithm.

