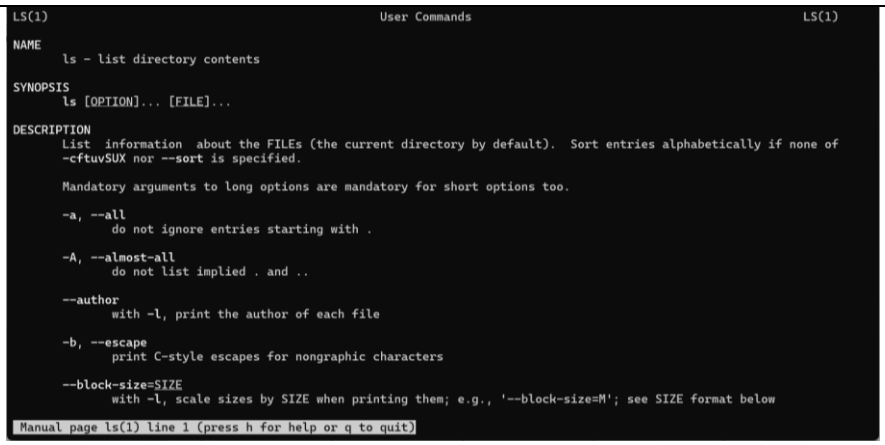
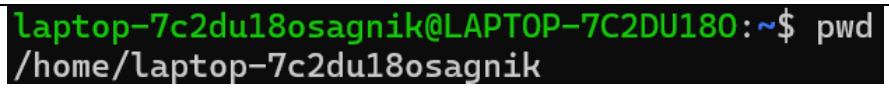
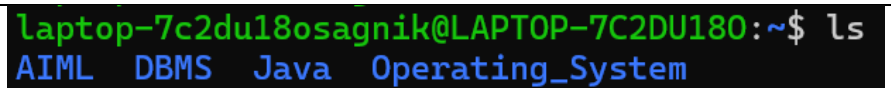
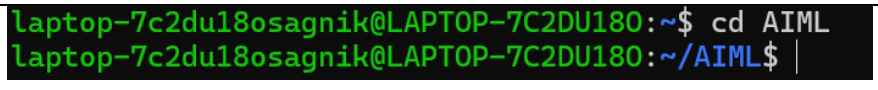
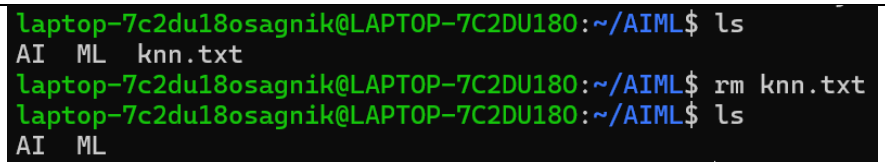
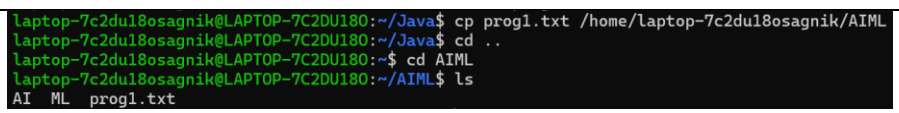
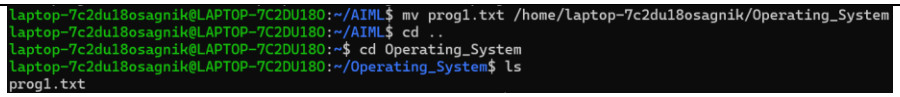
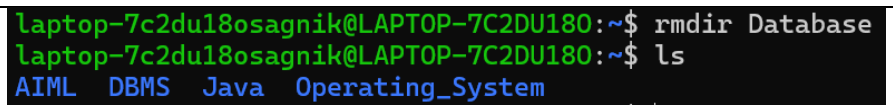
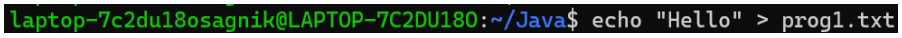
Assignment 1

0) For each command give a brief description of what it does and how it can be used.

Solution:

Com-mand	Description	Syntax	Sample Output
man	To display the manual page for a given command	`$ man [command]`	
who	Displays currently logged-in users of the system	`$ who`	
whoami	Displays the username of the currently logged-in user	`$ whoami`	
pwd	Prints the current working directory	`$ pwd`	
ls	Lists files and directories in the current directory	`$ ls`	
cd	Changes the current directory	`$ cd [directory name]`	
rm	Removes files	`$ rm [file name]`	
cp	Makes copies of files and directories	`$ cp [source] [destination]`	

mv	Moves files to another directory	`$ mv [source] [destination]`	
mkdir	Creates a new directory	`$ mkdir [directory]`	
rmdir	Removes an empty directory	`$ rmdir [directory]`	
echo	Displays a text or message on the screen	`$ echo [argument]`	
cat	Display file contents on terminal	`$ cat [file name]`	
wc	Counts words, lines, and characters in a file	`$ wc [file name]`	

1) **Provide a short write-up on the following:**
   - **History of Unix and Linux**
   - **Kernel of an Operating System**
   - **Multi-Tasking OS**
   - **Multi-User OS**

   Solution:

   - ***History of Unix and Linux:*** Unix, developed in the late 1960s and early 1970s at AT&T's Bell Labs, was created by Ken Thompson, Dennis Ritchie, and others. It introduced many concepts that became foundational to modern operating systems, such as the hierarchical file system and multi-tasking capabilities. Unix's design emphasized simplicity, portability, and the use of plain text for data storage. Linux, inspired by Unix, was created by Linus Torvalds in 1991. Unlike Unix, which was proprietary, Linux was developed as an open-source project. It quickly gained popularity due to its flexibility, robustness, and the collaborative nature of its development, leading to widespread adoption in various domains, from personal computers to enterprise servers and mobile devices.

   - ***Kernel of an Operating System:*** The kernel is the core component of an operating system, responsible for managing system resources and facilitating communication between hardware and software. It operates in a privileged mode, handling tasks such as memory management, process scheduling, input/output operations, and ensuring system security and stability. The kernel acts as an intermediary, ensuring that applications can run efficiently and safely without directly accessing hardware resources. There are different types of kernels, including monolithic, microkernel, and hybrid, each with its approach to structuring and managing the operating system's core functions.

   - ***Multi-Tasking OS:*** A multi-tasking operating system allows multiple processes or tasks to run concurrently on a single CPU or across multiple CPUs. It achieves this by allocating time slices to each task and switching between them rapidly, giving the illusion that they are running simultaneously. This enhances system

efficiency and responsiveness, enabling users to run multiple applications, such as a web browser, text editor, and media player, at the same time. Multi-tasking can be either preemptive, where the OS decides the allocation of CPU time, or cooperative, where tasks voluntarily yield control to each other.

- ***Multi-User OS:*** A multi-user operating system allows multiple users to access and utilize the computer resources simultaneously. Each user has a distinct profile and set of permissions, ensuring data security and privacy. Multi-user systems are commonly used in environments such as servers, mainframes, and networked computers, where resources need to be shared efficiently among many users. Features like user authentication, process isolation, and resource quotas are integral to multi-user operating systems, ensuring that users can operate independently without interfering with each other's activities.

2) List all the files and directories of '/bin' with detail information from your current directory.

Solution:

```
laptop-7c2du18osagnik@LAPTOP-7C2DU18O:~$ ls /bin
 NF                              id                              rsync
'['                              info                            rsync-ssl
 aa-enabled                      infobrowser                     rtstat
 aa-exec                         infocmp                         run-one
 aa-features-abi                 infotocap                       run-one-constantly
 add-apt-repository              install                         run-one-until-failure
 addpart                         install-info                    run-one-until-success
 addr2line                       instmodsh                       run-parts
 apport-bug                      ionice                          run-this-one
 apport-cli                      ip                              runcon
 apport-collect                  ipcmk                           rview
 apport-unpack                   ipcrm                           rvim
 apropos                         ipcs                            savelog
 apt                             iptables-xml                    scp
 apt-add-repository              ischroot                        screen
 apt-cache                       join                            screendump
 apt-cdrom                       journalctl                      script
 apt-config                      json_pp                         scriptlive
 apt-extracttemplates            kbd_mode                        scriptreplay
 apt-ftparchive                  kbdinfo                         sdiff
 apt-get                         kbxutil                         sed
 apt-key                         keep-one-running                select-editor
 apt-mark                        kernel-install                  sensible-browser
 apt-sortpkgs                    keyring                         sensible-editor
 ar                              kill                            sensible-pager
 arch                            killall                         seq
 as                              kmod                            setarch
 awk                             last                            setfont
 b2sum                           lastb                           setkeycodes
```

3) List all the files including hidden files in your parent directory.

Solution:

```
laptop-7c2du18osagnik@LAPTOP-7C2DU18O:~$ ls -la ..
total 12
drwxr-xr-x  3 root                  root                  4096 Jul 29 21:44 .
drwxr-xr-x 19 root                  root                  4096 Jul 30 19:24 ..
drwxr-x---  6 laptop-7c2du18osagnik laptop-7c2du18osagnik 4096 Jul 30 21:18 laptop-7c2du18osagnik
```

4) List only the directory files in your current directory.

Solution:

```
laptop-7c2du18osagnik@LAPTOP-7C2DU18O:~$ ls -d */
AIML/  DBMS/  Java/  Operating_System/
```

5) Create a file ‘text1’ by taking input from the keyboard.

Solution:

```
laptop-7c2du18osagnik@LAPTOP-7C2DU18O:~/Operating_System$ touch text1.txt
laptop-7c2du18osagnik@LAPTOP-7C2DU18O:~/Operating_System$ echo "Welcome to OS Lab" > text1.txt
laptop-7c2du18osagnik@LAPTOP-7C2DU18O:~/Operating_System$ cat text1.txt
Welcome to OS Lab
```

6) Copy the contents of file’ text1’ to another file ‘text2’.

Solution:

```
laptop-7c2du18osagnik@LAPTOP-7C2DU18O:~/Operating_System$ cp text1.txt text2.txt
laptop-7c2du18osagnik@LAPTOP-7C2DU18O:~/Operating_System$ cat text2.txt
Welcome to OS Lab
```

7) Append the contents of file ‘text2 ‘to file ‘text1’.

Solution:

```
laptop-7c2du18osagnik@LAPTOP-7C2DU18O:~/Operating_System$ cat text2.txt >> text1.txt
laptop-7c2du18osagnik@LAPTOP-7C2DU18O:~/Operating_System$ cat text1.txt
Welcome to OS Lab
Welcome to OS Lab
```

8) Count the number of lines in the file ‘text1’.

Solution:

```
laptop-7c2du18osagnik@LAPTOP-7C2DU18O:~/Operating_System$ wc -l text1.txt
2 text1.txt
```

Assignment 2

0) For each command give a brief description of what it does and how it can be used.

Solution:

Com-mand	Description	Syntax	Sample Output
cal	Displays a calendar.	**$ cal**	laptop-7c2du18osagnik@LAPTOP-7C2DU18O:~$ cal August 2024 Su Mo Tu We Th Fr Sa 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
date	Displays or sets the system date and time.	**$ date**	laptop-7c2du18osagnik@LAPTOP-7C2DU18O:~$ date Tue Aug 6 18:42:21 IST 2024
cmp	Compares two files byte by byte.	**$ cmp [file1] [file2]**	laptop-7c2du18osagnik@LAPTOP-7C2DU18O:~$ echo "Welcome to OS Lab" > text1.txt laptop-7c2du18osagnik@LAPTOP-7C2DU18O:~$ echo "Welcome to AIML Lab" > text2.txt laptop-7c2du18osagnik@LAPTOP-7C2DU18O:~$ cmp text1.txt text2.txt text1.txt text2.txt differ: byte 12, line 1
comm	Compares two sorted files line by line.	**$ comm [file1] [file2]**	laptop-7c2du18osagnik@LAPTOP-7C2DU18O:~$ comm text1.txt text2.txt Welcome to AIML Lab Welcome to OS Lab
diff	Shows the differences between files.	**$ diff [file1] [file2]**	laptop-7c2du18osagnik@LAPTOP-7C2DU18O:~$ diff text1.txt text2.txt 1c1 < Welcome to OS Lab --- > Welcome to AIML Lab
head	Outputs the first part of files as mentioned.	**$ head -n [number] [file]**	laptop-7c2du18osagnik@LAPTOP-7C2DU18O:~$ cat <<EOP > nfile.txt > This is a sample file > It contains several lines > Some lines have the word hello > Others do not have > EOP laptop-7c2du18osagnik@LAPTOP-7C2DU18O:~$ head -n 2 nfile.txt This is a sample file It contains several lines
tail	Outputs the last part of files.	**$ tail -n [number] [file]**	laptop-7c2du18osagnik@LAPTOP-7C2DU18O:~$ tail -n 3 nfile.txt It contains several lines Some lines have the word hello Others do not have
sort	Sorts lines of text files or numerically.	**$ sort [file]** **or** **$ sort -n [file]**	laptop-7c2du18osagnik@LAPTOP-7C2DU18O:~$ cat <<EOP > num.txt > 5 > 2 > 1 > 8 > 6 > EOP laptop-7c2du18osagnik@LAPTOP-7C2DU18O:~$ sort -n num.txt 1 2 5 6 8

bc	A command-line calculator.	`$ echo "[expression]" | bc`	`laptop-7c2du18osagnik@LAPTOP-7C2DU18O:~$ echo "7 + 12" | bc` `19`
expr	Evaluates expressions.	`$ expr [expression]`	`laptop-7c2du18osagnik@LAPTOP-7C2DU18O:~$ expr 4 + 16 / 2` `12`

grep	Searches for patterns in files.	`$ grep [pattern] [file]`	`laptop-7c2du18osagnik@LAPTOP-7C2DU18O:~$ grep have nfile.txt`
			`Some lines have the word hello`
			`Others do not have`

1) **a. Display the current time in 12-hour format.**

Solution:

```
laptop-7c2du18osagnik@LAPTOP-7C2DU18O:~$ date +"%r"
07:20:34 PM
```

b. With a user-specified date, display only the day of the week.

Solution:

```
laptop-7c2du18osagnik@LAPTOP-7C2DU18O:~$ date -d "2020-07-15" +"%A"
Wednesday
```

2) **Write the command to find the square root of 4.**

Solution:

```
laptop-7c2du18osagnik@LAPTOP-7C2DU18O:~$ echo "sqrt(4)" | bc
2
```

3) **Show how we can calculate the following expression in the terminal of UNIX A=5, b=6, z=15**

Total = (A*b) + (z/A)

Display the Total.

Solution:

```
laptop-7c2du18osagnik@LAPTOP-7C2DU18O:~$ A=5; b=6; z=15;
laptop-7c2du18osagnik@LAPTOP-7C2DU18O:~$ Total=$(echo "($A * $b) + ($z / $A)" | bc)
laptop-7c2du18osagnik@LAPTOP-7C2DU18O:~$ echo "Total=$Total"
Total=33
```

4) **How can we sort a list of numbers in a file (both ascending and descending order)?**

Solution:

```
laptop-7c2du18osagnik@LAPTOP-7C2DU18O:~$ cat <<EOP > numbers.txt
> 7
> 15
> 2
> 55
> 12
> 45
> EOP
```

```
laptop-7c2du18osagnik@LAPTOP-7C2DU18O:~$ sort -n numbers.txt
2
7
12
15
45
55
laptop-7c2du18osagnik@LAPTOP-7C2DU18O:~$ sort -nr numbers.txt
55
45
15
12
7
2
```

5) Create the file *student.dat* as follows:
Roll | Name | Dept | Year
105 | Anik | CSE | 1st
101 | Debesh | CSE | 2nd
108 | Aniket | IT | 1st
200 | Mainak | ECE | 2nd
105 | Anik | CSE | 1st

a. Sort the data according to Roll.

Solution:

```
laptop-7c2du18osagnik@LAPTOP-7C2DU18O:~$ cat <<EOL > student.dat
> Roll | Name | Dept | Year
> 105 | Anik | CSE | 1st
> 101 | Debesh | CSE | 2nd
> 108 | Aniket | IT | 1st
> 200 | Mainak | ECE | 2nd
> 105 | Anik | CSE | 1st
> EOL
laptop-7c2du18osagnik@LAPTOP-7C2DU18O:~$ sort -t '|' -k1,1n student.dat
Roll | Name | Dept | Year
101 | Debesh | CSE | 2nd
105 | Anik | CSE | 1st
105 | Anik | CSE | 1st
108 | Aniket | IT | 1st
200 | Mainak | ECE | 2nd
```

b. Sort the data according to Dept.

Solution:

```
laptop-7c2du18osagnik@LAPTOP-7C2DU18O:~$ sort -t '|' -k3,3 student.dat
101 | Debesh | CSE | 2nd
105 | Anik | CSE | 1st
105 | Anik | CSE | 1st
Roll | Name | Dept | Year
200 | Mainak | ECE | 2nd
108 | Aniket | IT | 1st
```

c. Show only the records of students from the CSE Dept.

Solution:

```
laptop-7c2du18osagnik@LAPTOP-7C2DU18O:~$ grep "CSE" student.dat
105 | Anik | CSE | 1st
101 | Debesh | CSE | 2nd
105 | Anik | CSE | 1st
```

6) Show the last 2 lines of the file *animals.txt*.

Solution:

```
laptop-7c2du18osagnik@LAPTOP-7C2DU18O:~$ cat <<EOL > animals.txt
> Dog is a domestic animal
> Dog hates cat
> Cat drinks milk
> Dog is bigger than Cat
> Cat is also a domestic animal
> EOL
```

```
laptop-7c2du18osagnik@LAPTOP-7C2DU18O:~$ tail -n 2 animals.txt
Dog is bigger than Cat
Cat is also a domestic animal
```

7) Show the first 3 lines of the file *animals.txt*.

Solution:

```
laptop-7c2du18osagnik@LAPTOP-7C2DU18O:~$ head -n 3 animals.txt
Dog is a domestic animal
Dog hates cat
Cat drinks milk
```

8) List only the directory files in your current directory.

Solution:

```
laptop-7c2du18osagnik@LAPTOP-7C2DU18O:~$ ls -d */
AIML/  DBMS/  Java/  Operating_System/
```

9) Count the number of directories in your current directory.

Solution:

```
laptop-7c2du18osagnik@LAPTOP-7C2DU18O:~$ ls -l | grep ^d | wc -l
4
```

10) Dog is a domestic animal
Dog hates cat
Cat drinks milk
Dog is bigger than Cat
Cat is also a domestic animal

a. Find the total number of lines contains the word 'Dog' in *animals.txt*.

Solution:

```
laptop-7c2du18osagnik@LAPTOP-7C2DU18O:~$ cat <<EOL > animals.txt
> Dog is a domestic animal
> Dog hates cat
> Cat drinks milk
> Dog is bigger than Cat
> Cat is also a domestic animal
> EOL
laptop-7c2du18osagnik@LAPTOP-7C2DU18O:~$ grep -c "Dog" animals.txt
3
```

b. Also find the total number of lines does not contain the word 'Dog' in *animals.txt*.

Solution:

```
laptop-7c2du18osagnik@LAPTOP-7C2DU18O:~$ grep -vc "Dog" animals.txt
2
```

c. Display the lines in *animals.txt* that end with the word 'cat'.

Solution:

```
laptop-7c2du18osagnik@LAPTOP-7C2DU18O:~$ grep "cat$" animals.txt
Dog hates cat
```

Assignment 3

Metacharacters And Its Applications

Character	Usage	Syntax	Sample Output

*	Matches any and all characters in a filename or path.	`*`	`laptop-7c2du18osagnik@LAPTOP-7C2DU18O:~$ ls *`
			`animals.txt file2.txt file4.txt file6.txt num.txt sorted_by_dept.dat t1.txt text2.txt text4.tx`
			`file1.txt file3.txt file5.txt nfile.txt numbers.txt student.dat text1.txt text3.txt text5.tx`
			
			`AIML:`
			`AI ML`
			
			`DBMS:`
			
			`Java:`
			`prog1.txt`
			
			`Operating_System:`
			`prog1.txt text1.txt text2.txt`

?	Matches a single character in a filename or path.	`?`	`laptop-7c2du18osagnik@LAPTOP-7C2DU18O:~$ ls file?.txt` `file1.txt file2.txt file3.txt file4.txt file5.txt file6.txt`
[ ]	Matches any one of the specified set of characters within it.	`[a,b,c]`	`laptop-7c2du18osagnik@LAPTOP-7C2DU18O:~$ ls [a,s]*` `animals.txt sorted_by_dept.dat student.dat`
-	Specifies a range of characters within square brackets.	`[a-z]`	`laptop-7c2du18osagnik@LAPTOP-7C2DU18O:~$ ls text[1-3].txt` `text1.txt text2.txt text3.txt`

>	Redirects the output of a command to a file, overwriting the file if it exists.	`command > output.txt`	`laptop-7c2du18osagnik@LAPTOP-7C2DU18O:~$ cat > file1.txt`
			`1`
			`5`
			`2`
			`6`
			`3`

<	Redirects the input to a command from a file.	`command < input.txt`	`laptop-7c2du18osagnik@LAPTOP-7C2DU18O:~$ sort < file1.txt`
			`1`
			`2`
			`3`
			`5`
			`6`

|	Connect the output of one command to the input of another command.	`command | command`	`laptop-7c2du18osagnik@LAPTOP-7C2DU18O:~$ cat file1.txt | wc -l` `5`

$	Refers to a shell variable.	`$variable_name`	`laptop-7c2du18osagnik@LAPTOP-7C2DU18O:~$ A=65`
			`laptop-7c2du18osagnik@LAPTOP-7C2DU18O:~$ echo "Value of A=$A"`
			`Value of A=65`

Assignment 4

1. **Practice chmod command and all its parameters.**

Solution:

The *chmod* command is basically used to change the file permissions. The general syntax is:

```
chmod [options] mode file
```

- **mode**: Defines the permissions to be set.
- **file**: The file whose permissions are to be changed.

Different Modes:

- **u**: User who owns the file.
- **g**: Group that owns the file.
- **o**: Others (everyone else).
- **a**: All (u, g, and o).

Permissions:

- **r**: Read.
- **w**: Write.
- **x**: Execute.

```
laptop-7c2du18osagnik@LAPTOP-7C2DU18O:~$ ls -l
total 56
drwxr-xr-x 2 laptop-7c2du18osagnik laptop-7c2du18osagnik 4096 Jul 30 20:01 AIML
drwxr-xr-x 2 laptop-7c2du18osagnik laptop-7c2du18osagnik 4096 Jul 30 19:39 DBMS
drwxr-xr-x 2 laptop-7c2du18osagnik laptop-7c2du18osagnik 4096 Jul 30 20:17 Java
drwxr-xr-x 2 laptop-7c2du18osagnik laptop-7c2du18osagnik 4096 Jul 30 20:38 Operating_System
-rw-r--r-- 1 laptop-7c2du18osagnik laptop-7c2du18osagnik  108 Aug  6 19:53 animals.txt
-rw-r--r-- 1 laptop-7c2du18osagnik laptop-7c2du18osagnik   10 Aug 13 19:33 file1.txt
-rw-r--r-- 1 laptop-7c2du18osagnik laptop-7c2du18osagnik    0 Aug 13 18:46 file2.txt
-rw-r--r-- 1 laptop-7c2du18osagnik laptop-7c2du18osagnik    0 Aug 13 18:46 file3.txt
-rw-r--r-- 1 laptop-7c2du18osagnik laptop-7c2du18osagnik    0 Aug 13 18:46 file4.txt
-rw-r--r-- 1 laptop-7c2du18osagnik laptop-7c2du18osagnik    0 Aug 13 18:46 file5.txt
-rw-r--r-- 1 laptop-7c2du18osagnik laptop-7c2du18osagnik    0 Aug 13 18:46 file6.txt
-rw-r--r-- 1 laptop-7c2du18osagnik laptop-7c2du18osagnik   98 Aug  6 19:07 nfile.txt
-rw-r--r-- 1 laptop-7c2du18osagnik laptop-7c2du18osagnik   36 Aug 13 19:31 num.txt
-rw-r--r-- 1 laptop-7c2du18osagnik laptop-7c2du18osagnik   16 Aug  6 19:34 numbers.txt
-rw-r--r-- 1 laptop-7c2du18osagnik laptop-7c2du18osagnik  146 Aug  6 19:46 sorted_by_dept.dat
-rw-r--r-- 1 laptop-7c2du18osagnik laptop-7c2du18osagnik  146 Aug  6 19:38 student.dat
-rw-r--r-- 1 laptop-7c2du18osagnik laptop-7c2du18osagnik   50 Jul 30 21:19 t1.txt
-rw-r--r-- 1 laptop-7c2du18osagnik laptop-7c2du18osagnik   18 Aug 13 18:45 text1.txt
-rw-r--r-- 1 laptop-7c2du18osagnik laptop-7c2du18osagnik   20 Aug 13 18:45 text2.txt
-rw-r--r-- 1 laptop-7c2du18osagnik laptop-7c2du18osagnik    0 Aug 13 18:45 text3.txt
-rw-r--r-- 1 laptop-7c2du18osagnik laptop-7c2du18osagnik    0 Aug 13 18:46 text4.txt
-rw-r--r-- 1 laptop-7c2du18osagnik laptop-7c2du18osagnik    0 Aug 13 18:46 text5.txt
```

i. Adding execute permission for the user:

```
chmod u+x file1.txt
```

```
-rwxr--r-- 1 laptop-7c2du18osagnik laptop-7c2du18osagnik   10 Aug 13 19:33 file1.txt
```

ii. Removing read permission for the group:

```
chmod g-r numbers.txt
```

```
-rw----r-- 1 laptop-7c2du18osagnik laptop-7c2du18osagnik   16 Aug  6 19:34 numbers.txt
```

iii. Setting execute-only permissions for everyone:

```
chmod a=x text2.txt
```

```
---x--x--x 1 laptop-7c2du18osagnik laptop-7c2du18osagnik   20 Aug 13 18:45 text2.txt
```

iv. Setting full permissions (read, write, execute) for the user, and read-only for group and others:

```
chmod 744 nfile.txt
```

```
-rwxr--r-- 1 laptop-7c2du18osagnik laptop-7c2du18osagnik   98 Aug  6 19:07 nfile.txt
```

v. Setting no permissions at all (completely restrict access):

```
chmod 000 resul.txt
```

```
---------- 1 laptop-7c2du18osagnik laptop-7c2du18osagnik    0 Aug 13 18:46 file3.txt
```

2. Determine if a file exists or not using test command.

Solution:

The *test* command is used to evaluate expressions. To check if a file exists, we can use:

```
test -e file
```

This returns a status code:

- *0* if the file exists.
- *1* if the file does not exist.

```
laptop-7c2du18osagnik@LAPTOP-7C2DU18O:~$ test -e text1.txt; echo $?
0
laptop-7c2du18osagnik@LAPTOP-7C2DU18O:~$ test -e result.txt; echo $?
1
```

3. Compare two strings using test command.

Solution:

```
laptop-7c2du18osagnik@LAPTOP-7C2DU18O:~$ if test "hello" = "world"; then
    echo "The strings are equal."
else
    echo "The strings are not equal."
fi
The strings are not equal.
```

```
laptop-7c2du18osagnik@LAPTOP-7C2DU18O:~$ if test "OS" = "OS"; then
    echo "The strings are equal."
else
    echo "The strings are not equal."
fi
The strings are equal.
```

Assignment 5

1) Write a shell program to add two numbers given by user.

Solution:

prog1.sh

```
echo "Enter first number:"
read num1
echo "Enter second number:"
read num2
sum=$((num1 + num2))
echo "The sum of $num1 and $num2 is: $sum"
```

```
laptop-7c2du18osagnik@LAPTOP-7C2DU18O:~$ bash prog1.sh
Enter first number:
12
Enter second number:
34
The sum of 12 and 34 is: 46
```

2) Write a shell program to perform the swapping between two numbers taken from user during run time.

Solution:

prog2.sh

```
echo "Enter first number:"
read num1
echo "Enter second number:"
read num2
echo "Before swapping: num1 = $num1, num2 = $num2"
temp=$num1
num1=$num2
num2=$temp
echo "After swapping: num1 = $num1, num2 = $num2"
```

```
laptop-7c2du18osagnik@LAPTOP-7C2DU18O:~$ bash prog2.sh
Enter first number:
5
Enter second number:
15
Before swapping: num1 = 5, num2 = 15
After swapping: num1 = 15, num2 = 5
```

3) Write a shell program to perform the multiply two numbers given as command line arguments.

Solution:

prog3.sh

```
if [ $# -ne 2 ]; then
  echo "Please provide exactly two numbers as arguments."
  exit 1
fi
product=$(( $1 * $2 ))
echo "The product of $1 and $2 is: $product"
```

```
laptop-7c2du18osagnik@LAPTOP-7C2DU18O:~$ bash prog3.sh 10 3
The product of 10 and 3 is: 30
```

4) Write a shell program to print the largest among three numbers by passing the numbers through command line arguments.

Solution:

prog4.sh

```
if [ $# -ne 3 ]; then
  echo "Please provide exactly three numbers as arguments."
  exit 1
fi
if [ $1 -ge $2 ] && [ $1 -ge $3 ]; then
  largest=$1
elif [ $2 -ge $1 ] && [ $2 -ge $3 ]; then
  largest=$2
else
  largest=$3
fi
echo "The largest number among $1, $2, and $3 is: $largest"
```

```
laptop-7c2du18osagnik@LAPTOP-7C2DU18O:~$ bash prog4.sh 70 45 58
The largest number among 70, 45, and 58 is: 70
```

5) Write a shell program to display the following mark sheets of students by taking the input marks of student through the terminal

Marks range	Grade
90>=M<=100	A
70>=M<=89	B
40>=M<=69	C
M<40	F

Solution:

prog5.sh

```
echo "Enter the marks of the student:"
read marks
if [ $marks -ge 90 ] && [ $marks -le 100 ]; then
  grade="A"
elif [ $marks -ge 70 ] && [ $marks -le 89 ]; then
  grade="B"
elif [ $marks -ge 40 ] && [ $marks -le 69 ]; then
  grade="C"
elif [ $marks -lt 40 ]; then
  grade="F"
else
  echo "Invalid marks entered."
  exit 1
fi
echo "The grade for marks $marks is: $grade"
```

```
laptop-7c2du18osagnik@LAPTOP-7C2DU18O:~$ bash prog5.sh
Enter the marks of the student:
92
The grade for marks 92 is: A
```

Assignment 6

1) Write a shell program to calculate the factorial of a number.

Solution:

factorial.sh

```
echo -n "Enter a number: "
read num

fact=1
for i in $(seq 1 $num)
do
  fact=$((fact * i))
done
echo "Factorial of $num is $fact"
```

```
laptop-7c2du18osagnik@LAPTOP-7C2DU18O:~$ bash factorial.sh
Enter a number: 6
Factorial of 6 is 720
```

2) Write a shell menu driven program to do the following:

a. **Display the current working directory.**
b. **Check whether an input number is even or odd.**
c. **Display the number of counts of all the files in the directory.**
d. **Print the long listing of all the files.**

Solution:

menu.sh

```
echo "Menu:"
echo "1. Display the current working directory"
echo "2. Check whether an input number is even or odd"
echo "3. Display the number of counts of all the files in the directory"
echo "4. Print the long listing of all the files"
echo -n "Enter your choice [1-4]: "
read choice

case $choice in
    1)
        echo "Current working directory: $(pwd)"
        ;;
    2)
        echo -n "Enter a number: "
        read num
        if (( num % 2 == 0 )); then
            echo "$num is even"
        else
            echo "$num is odd"
        fi
        ;;
    3)
        echo "Number of files in the directory: $(ls -1 | wc -l)"
        ;;
    4)
        echo "Long listing of all files:"
        ls -l
```

```
        ;;
    *)
        echo "Invalid choice, please try again."
        ;;
esac
```

```
laptop-7c2du18osagnik@LAPTOP-7C2DU18O:~$ bash menu.sh
Menu:
1. Display the current working directory
2. Check whether an input number is even or odd
3. Display the number of counts of all the files in the directory
4. Print the long listing of all the files
Enter your choice [1-4]: 2
Enter a number: 54
54 is even
laptop-7c2du18osagnik@LAPTOP-7C2DU18O:~$ bash menu.sh
Menu:
1. Display the current working directory
2. Check whether an input number is even or odd
3. Display the number of counts of all the files in the directory
4. Print the long listing of all the files
Enter your choice [1-4]: 3
Number of files in the directory: 34
```

3) Write a shell program to display all the prime numbers between 1 to 100 using while loop.

Solution:

prime.sh

```
num=2

echo "Prime numbers between 1 to 100 are:"

while [ $num -le 100 ]
do
    is_prime=1
    i=2

    while [ $i -le $(($num / 2)) ]
    do
        if [ $(($num % $i)) -eq 0 ]
        then
            is_prime=0
            break
        fi

        i=$((i + 1))
    done

    if [ $is_prime -eq 1 ]
    then
        echo $num
    fi

    num=$((num + 1))
done
```

```
laptop-7c2du18osagnik@LAPTOP-7C2DU18O:~$ bash prime.sh
Prime numbers between 1 to 100 are:
2
3
5
7
11
13
17
19
23
29
31
37
41
43
47
53
59
61
67
71
73
79
83
89
97
```

4) Write a shell menu program to find out whether a given letter is vowel or not.

Solution:

vowel.sh

```
while true
do
    echo "Menu:"
    echo "1. Check if a letter is a vowel"
    echo "2. Exit"
    echo -n "Enter your choice [1-2]: "
    read choice

    case $choice in
        1)
            echo -n "Enter a letter: "
            read letter
            let=$(echo $letter | tr '[:upper:]' '[:lower:]')

            if [[ $let == "a" || $let == "e" || $let == "i" || $let == "o" || $let == "u"
]]
            then
                echo "$letter is a vowel."
            else
                echo "$letter is not a vowel."
            fi
            ;;
        2)
            echo "Exiting..."
            exit 0
            ;;
```

```
        *)
            echo "Invalid choice, please try again."
            ;;
    esac

    echo
done
```

```
laptop-7c2du18osagnik@LAPTOP-7C2DU18O:~$ bash vowel.sh
Menu:
1. Check if a letter is a vowel
2. Exit
Enter your choice [1-2]: 1
Enter a letter: U
U is a vowel.

Menu:
1. Check if a letter is a vowel
2. Exit
Enter your choice [1-2]: 1
Enter a letter: T
T is not a vowel.

Menu:
1. Check if a letter is a vowel
2. Exit
Enter your choice [1-2]: 2
Exiting...
```

5) Write a shell script which will generate the output as follows:

```
*
* *
* * *
* * * *
```

Solution:

pattern.sh

```
rows=4

for ((i=1; i<=rows; i++))
do
    for ((j=1; j<=i; j++))
    do
        echo -n "* "
    done
    echo
done
```

```
laptop-7c2du18osagnik@LAPTOP-7C2DU18O:~$ bash pattern.sh
*
* *
* * *
* * * *
```

6) Write a shell script that computes the gross salary of a employee according to the following rules:

i. **If basic salary is < 1500 then HRA =10% of the basic and DA =90% of the basic.**

ii. **If basic salary is >=1500 then HRA =Rs500 and DA=98% of the basic.**

The basic salary is entered interactively through the key board.

Solution:

gsalary.sh

```
echo -n "Enter the basic salary: "
read basic

if [ $basic -lt 1500 ]
then
    hra=$(echo "scale=2; $basic * 0.10" | bc)
    da=$(echo "scale=2; $basic * 0.90" | bc)
else
    hra=500
    da=$(echo "scale=2; $basic * 0.98" | bc)
fi

gross_salary=$(echo "scale=2; $basic + $hra + $da" | bc)
echo "Gross Salary is: $gross_salary"
```

```
laptop-7c2du18osagnik@LAPTOP-7C2DU18O:~$ bash gsalary.sh
Enter the basic salary: 1750
Gross Salary is: 3965.00
```