EXPERIMENT 2

Vi EDITOR COMMANDS

AIM:

To study the various command so Operated in vi editor in UNIX.

DESCRIPTION:

The Vi editor is a visual editor used to create and edit text, files, documents and programs. It displays the content of files on the screen and allows a user to add, delete or change part of text. There are three mode available in the Vi editor, they are

1. Command mode

2. Input(or)insert mode.

StaringVi

Syntax:

```
$vi<filename>(or)
     $vi
```

This command would open a display screen with 25 lines and with tiltd (~) symbol at the start of each line. The first syntax would save the file in the filename mentioned and for then ext the file name must be mentioned at the end.

INSERTINGANDREPLACINGCOMMANDS:

To move editor from command node toed it mode, you have to press the<ESC>key. For inserting and replacing the following commands are used.

1. <ESC>a Command

This command issued to move the edit mode and start to append after the current character.

2. <ESC>A Command

This command is also used to append the file, but this command append at the end of current line.

3. <ESC>i Command

This command issued to insert the text before the current cursor position.

4. <ESC>I Command

This command issued to insert at the beginning of the current line.

5. <ESC>o Command

This command is insert a blank line below the current line &allow insertion of contents.

6. <ESC>O Command

This command issued to insert a blank line above &allow insertion of contents.

7. <ESC>r Command

This command is to replace the particular character with the given characters.

8. <ESC>R Command

This command issued to replace the particular text with a given text.

9. <ESC>s Command

This command replaces a single character with a group of character.

10. <ESC>S Command

This command issued to replace a current line with group of characters.

DELETING THE TEXT FROM Vi

1. <ESC>x Command

To delete a character to right of current cursor positions.

2. <ESC>X Command

To delete a character to left of current cursor positions.

3. <ESC>dw Command

This command is to delete a single word or number of word storing of current cursor position.

4. <ESC>db Command

This command is to delete a single word to the left of the current cursor position.

5. <ESC>dd Command

This command issued to delete the current line(or)a number of lines below the current line.

6. <ESC>d$ Command

This command isused to delete the text from current cursor position to last character of current line.

SAVING AND OUITING FROM Vi

1. <ESC>d$ Command
To save the given text present in the file.

2. <ESC>q! Command
To quit the given text without saving.

3. <ESC>wq Command
This command quits the vi editor after saving the text in the mentioned file.

4. <ESC>x Command
This command is same as‘wq’command it saves and quit.

5. <ESC>q Command
This command would quit the window but it would ask for again to save the file.

BASIC LINUX COMMANDS

1.man

Used to display the user manual of any command that we can run on the terminal. It provides a detailed view of the command which includes name, synopsis, description, options, exit status, return values, errors, files, versions, examples, authors and see also.
Example:
$ man printf

- **-a option**: This option helps us to display all the available intro manual pages in succession.

2. echo

echo command in linux is used to display line of text/string that are passed as an argument . This is a built in command that is mostly used in shell scripts and batch files to output status text to the screen or a file.
Syntax :
echo [option] [string]
Displaying a text/string **:**
Syntax :
echo [string]

3.ls

The ls command is used to list files or directories in Linux and other Unix-based operating systems.

- ls .. command to list the contents of the parent directory one level above.
- ls ~ command to list the contents in the users's home directory:
- ls -d */ command to list only directories:
- ls * command to list the contents of the directory with it's subdirectories:
- ls -s command (the **s** is lowercase) to list files or directories with their sizes:

4. cat

The *cat* command is the simplest way to view the contents of a file**.** It displays the contents of the file(s) specified on to the output terminal**.**

5. more

The ***more*** command displays the contents of the file one screen at a time for large files**.** If the contents of the file fit a single screen, the output will be the same as the *cat* command.

6. less

The ***less*** command is similar to the ***more*** command but provides extensive features**.** One important one is that it allows backward as well as forward movement in the file**,** even with pipes**.**

7. cd

This command changes your current directory location. By default, your Unix login session begins in your home directory.

- To switch to a subdirectory (of the current directory) named myfiles, enter:

```
cd myfiles
```

- To switch to a directory named /home/dvader/empire_docs, enter:

```
cd /home/dvader/empire_docs
```

8. mkdir

This command will make a new subdirectory.

- To create a subdirectory named my stuff in the current directory, enter:

```
mkdir mystuff
```

9. pwd

This command reports the current directory path. Enter the command by itself:

```
pwd
```

10. mv

This command will move a file. You can use mv not only to change the directory location of a file, but also to rename files.

- To rename a file named old name in the current directory to the new name new name, enter:

mv -i oldname newname

11. cp

This command copies a file, preserving the original and creating an identical copy. If you already have a file with the new name, cp will overwrite and destroy the duplicate. For this reason, it's safest to always add -i after the cp command, to force the system to ask for your approval before it destroys any files. The general syntax for cp is:

cp -i oldfile newfile

12. rm

This command will remove (destroy) a file. You should enter this command with the -i option, so that you'll be asked to confirm each file deletion. To remove a file named junk, enter:

rm -i jun

13. tar

tar command in Linux is one of the important command which provides archiving functionality in Linux. An Archive file is a file that is composed of one or more files along with metadata. Archive files are used to collect multiple data files together into a single file for easier portability and storage, or simply to compress files to use less storage space.

Syntax:
tar [options] [archive-file] [file or directory to be archived]
Options:
-c : Creates Archive
-x : Extract the archive
-f : creates archive with given filename
-t : displays or lists files in archived file
-u : archives and adds to an existing archive file
-v : Displays Verbose Information
-A : Concatenates the archive files

14 . Cut

It can be used to cut parts of a line by **byte position, character and field**. Basically the cut command slices a line and extracts the text **Options and their Description with examples:**

15. Paste

It is used to join files horizontally (parallel merging) by outputting lines consisting of lines from each file specified, separated by **tab** as delimiter, to the standard output. When no file is specified, or put dash ("-") instead of file name, paste reads from standard input and gives output as it is until a interrupt command is given.

16. wc

wc stands for **word count**. As the name implies, it is mainly used for counting purpose.
Options:

- -l: This option prints the **number of lines** present in a file.
  With one file name
  $ wc -l state.txt
- **-w:** This option prints the **number of words** present in a file.

  With one file name
  $ wc -w state.txt
- **-c:** This option displays **count of bytes** present in a file.

  With one file name
  $ wc -c state.txt
- **-L:** The 'wc' command allow an argument **-L**, it can be used to print out the length of longest (number of characters) line in a file.

17. head

The head command, as the name implies, print the top N number of data of the given input. By default, it prints the first 10 lines of the specified files. If more than one file name is provided then data from each file is preceded by its file name.
Syntax:

head [OPTION]... [FILE]...

18. tail

The tail command, as the name implies, print the last N number of data of the given input. By default it prints the last 10 lines of the specified files. If more than one file name is provided then data from each file is precedes by its file name.

Syntax:

tail [OPTION]... [FILE]...

19. grep

The grep filter searches a file for a particular pattern of characters, and displays all lines that contain that pattern. The pattern that is searched in the file is referred to as the regular expression (grep stands for global search for regular expression and print out).

Syntax:

grep [options] pattern [files]

Options Description

-c : This prints only a count of the lines that match a pattern

-h : Display the matched lines, but do not display the filenames.

-i : Ignores, case for matching

-l : Displays list of a filenames only.

-n : Display the matched lines and their line numbers.

-v : This prints out all the lines that do not matches the pattern

-e exp : Specifies expression with this option. Can use multiple times.

-f file : Takes patterns from file, one per line.

20. expr

The **expr** command in Unix evaluates a given expression and displays its corresponding output. It is used for:

- Basic operations like addition, subtraction, multiplication, division, and modulus on integers.
- Evaluating regular expressions, string operations like substring, length of strings etc.

Syntax:

$expr expression

Option –version : It is used to show the version information.

Syntax:

$expr --version

21. chmod

chmod command is used to change the access mode of a file.

The name is an abbreviation of **change mode**.

Syntax :

chmod [reference][operator][mode] file..

The references are represented by one or more of the following letters:

Reference	Class	Description
u	owner	file's owner
g	group	users who are members of
o	others	file's owner nor members of the file's group
a	All	three of the above, same as ugo

There are three basic modes which correspond to the basic permissions:

r Permission to read the file.

w Permission to write (or delete) the file.

x Permission to execute the file, or, in

the case of a directory, search it.

22. chown

chown command is used to change the file Owner or group. Whenever you want to change ownership you can use chown command.

Syntax:
```
chown [OPTION]… [OWNER][:[GROUP]] FILE…
```
```
chown [OPTION]… –reference=RFILE FILE…
```

Example: To change owner of the file:
```
chown owner_name file_name
```

23. piping and redirection

pipe is a form of redirection (transfer of standard output to some other destination) that is used in Linux . Pipe is used to combine two or more commands, and in this, the output of one command acts as input to another command, and this command's output may act as input to the next command and so on

Syntax :

```
command_1 | command_2 | command_3 | .... | command_N
```

24. Useradd

useradd is a command in Linux that is used to add user accounts to your system.

Syntax:
```
useradd [options] name_of_the_user
```

25.usermod

usermod command or modify user is a command in Linux that is used to change the properties of a user in Linux through the command line.

26.Userdel

userdel command in Linux system is used to delete a user account and related files.

Syntax:
userdel [options] LOGIN

Options:

- **userdel -f:** This option forces the removal of the specified user account.
  - **userdel -r:** Whenever we are deleting a user using this option then the files in the user's home directory will be removed along with the home directory itself and the user's mail spool.

27. passwd

passwd command in Linux is used to change the user account passwords.

Syntax:
passwd [options] [username]

passwd options:

- **-d, –delete:** This option deletes the user password and makes the account password-less.
- **-e, –expire:** This option immediately expires the account password and forces the user to change password on their next login.

28.df

df command that displays the amount of disk space available on the file system containing each file name argument.

df Syntax :
df [OPTION]...[FILE]...

OPTION : to the options compatible with df command
FILE : specific filename in case you want to know
the disk space usage of a particular file system only.

29.top

top command is used to show the Linux processes. It provides a dynamic real-time view of the running system.
Examples
Display Specific User Process
top -u paras

30. ps

ps for viewing information related with the processes on a system which stands as abbreviation for **"Process Status".** ps command is used to list the currently running processes and their PIDs along with some other information depends on different options. It reads the process information from the virtual files in **/proc** file-system. /proc contains virtual files, this is the reason it's referred as a virtual file system.

Syntax –

ps [options]

31. scp

scp (secure copy) command in Linux system is used to copy file(s) between servers in a secure way.

Options:

- **scp –P port:** Specifies the port to connect on the remote host.
- **scp –p:** Preserves modification times, access times, and modes from the original file.
- **scp –q:** Disables the progress meter.
- **scp –r:** Recursively copy entire directories.

32. ssh

ssh stands for **“Secure Shell”**. It is a protocol used to securely connect to a remote server/system. ssh is secure in the sense that it transfers the data in encrypted form between the host and the client.

Syntax:

ssh user_name@host(IP/Domain_name)

33. ssh-copy-id

The ssh-copy-id command is a simple tool that allows you to install an SSH key on a remote server’s authorized keys. The ssh-copy-id tool, part of the OpenSSH package, is available in all major Linux distribution repositories, and you can use your package manager to install this command.

34. ssh-keygen

Use the ssh-keygen command to generate a public/private authentication key pair. Authentication keys allow a user to connect to a remote system without supplying a password. Keys must be generated for each user separately. If you generate key pairs as the root user, only the root can use the keys.

RESULT

EXPERIMENT3:
STUDY OF FILE HIERARCHY IN COMMON LINUX DISTRIBUTION

Sudo apt install tree

Once installed,stay in your terminal window and run *tree* like this:

$/tree

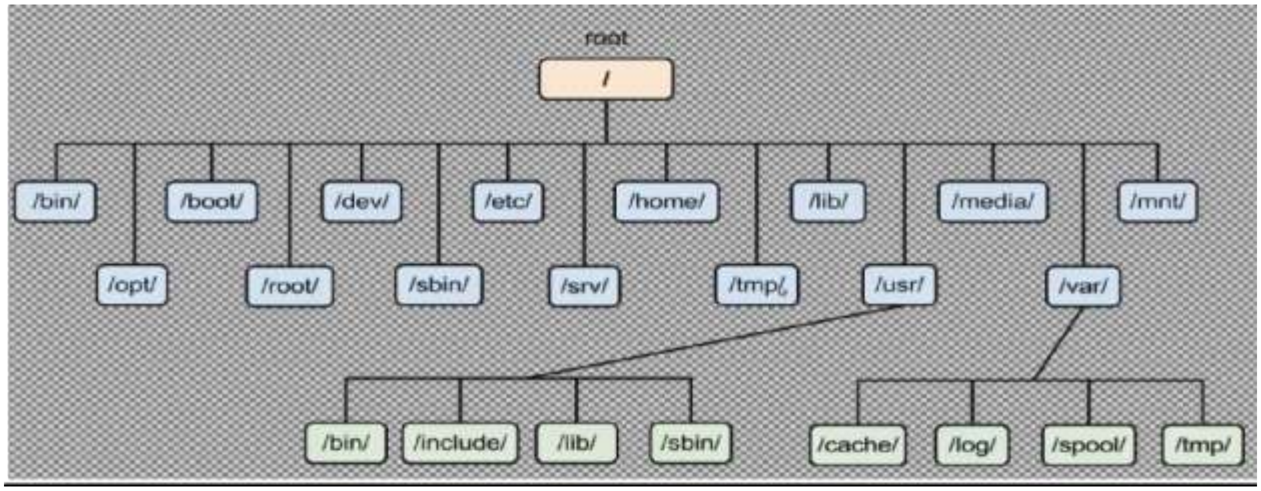
The/ in the in instruction above refers to the *root* directory. The root directory is the one from which all other directories branch off from. When you run thread tell it to start with/, you will See the whole directory tree, all directories and all the subdirectories in the whole system, with all their files, fly by.

/tree -L1

The instruction above can be translated as"*showmeonly the1st Level of the directory tree starting at / (root)*".The-*L* option tells *tree* how many levels down you want to see.

Now, let's look at what each directory issued for. While we go through each, you can peek at the contents using *ls*.

Directories



From top to bottom, the directories you are seeing areas follows.

- ***/bin***

/bin is the directory that contains *bin*aries, that is, some of the applications and programs you can run. You will find the *ls* program mentioned above in this directory, as well as other basic tools for making and removing files and directories, moving them around, and so on.

- ***/boot***

The */boot* directory contains files required for starting your system.If you mess up one of the files in here, you may not be able to run your Linux and it is a paint or repair.

- ***/dev***

/dev contains *dev*ice files. Many of these are generated at boot time or even on the fly. For example, if you plug in a new webcam or a USB pendrive into your machine, a new device entry will automatically pop up here.

- ***/etc***
/etc is the directory where names start to get confusing. */etc* gets its name from the earliest Unixes and it was literally "et cetera" because it was the dumping ground for system files administrators were not sure where else to put.
- ***/home***
/home is where you will find your users 'personal directories. In mycase, under*/home* there are two directories:*/home/paul*,which contains all my stuff;and*/home/guest*,in case any body needs to borrow my computer.

- ***/lib***
/lib is where *lib*raries live. Libraries are files containing code that your applications can use. They contain snippets of code that applications use to draw windows on your desktop, control peripherals ,or send files to your hard disk.
- ***/media***
The */media* directory is where external storage will be automatically mounted when you plug it in and try to access it.
- ***/mnt***
The */mnt* directory, however,is a bit of remnant from days gone by. This is where you would manually mount storage devices or partitions. It is not used very often nowadays.

- ***/opt***
The */opt* directory is often where software you compile(that is,you build yourself from source code and do not install from your distribution repositories) sometimes lands. Applications will end up in the*/opt/bin* directory and libraries in the*/opt/lib* directory.

- ***/proc***
/proc, like */dev* is a virtual directory. It contains information about your computer, such as information about your CPU and the kernel your Linux system is running.

- ***/root***
/root is the home directory of the super user (also known as the "Administrator") of the system.

- ***/run***
/run is another new directory. System processes use it to store temporary data for their own various reasons.

- ***/sbin***
/sb in is similar to */bin*,but it contains applications that only the superuser(hencetheinitial*s*)will need. You can use these applications with the sudo command that temporarily concedes you superuser power so many distributions.

- ***/usr***
The */usr* directory was where users' home directories were originally kept back in the early daysof UNIX. However, now */home* is where users kept their stuff as we saw above.
Directories that point to */usr/bin*.

- ***/srv***

  The */srv* directory contains data for servers. If you are running a web server from your Linux box,your HTML files for your sites would go into */srv/http* (or */srv/www*). If you were running an FTPserver,your files wouldgo into*/srv/ftp*.

- ***/sys***

  /sys is another virtual directory like*/proc* and */dev* and also contains information from devices connected to your computer. In some cases you can also manipulate those devices.

- ***/tmp***

  /tmp contains temporary files, usually placed there by applications that you are running. The files and directories often (not always) contain data that an application doesn't need right now, but may need later on.
- ***/var***

  /var was originally given its name because its contents was deemed *variable,* in that it changed frequently.

To explore the file system your self ,use the cd command:

- ***cd***

  will take you to the directory of your choice (*cd* stands for *change directory*.
  If you get confused,
- ***pwd***

  will always tell you where you (*pwd* stands for *print working directory*).Also,
- ***cd***

  with no options or parameters, will take you back to your own home directory, where things are safe and cosy.
- ***cd..***

  will take you upon elevel,getting you one level closer to the/root directory.If you are in */usr/share/wallpapers* and runcd ..,you will move upto */usr/share*.

EXPERIMENT4
SHELL SCRIPTING PROGRAMS

ARITHMETIC OPERATION

AIM:

Write a shell script program to perform arithmetic operation

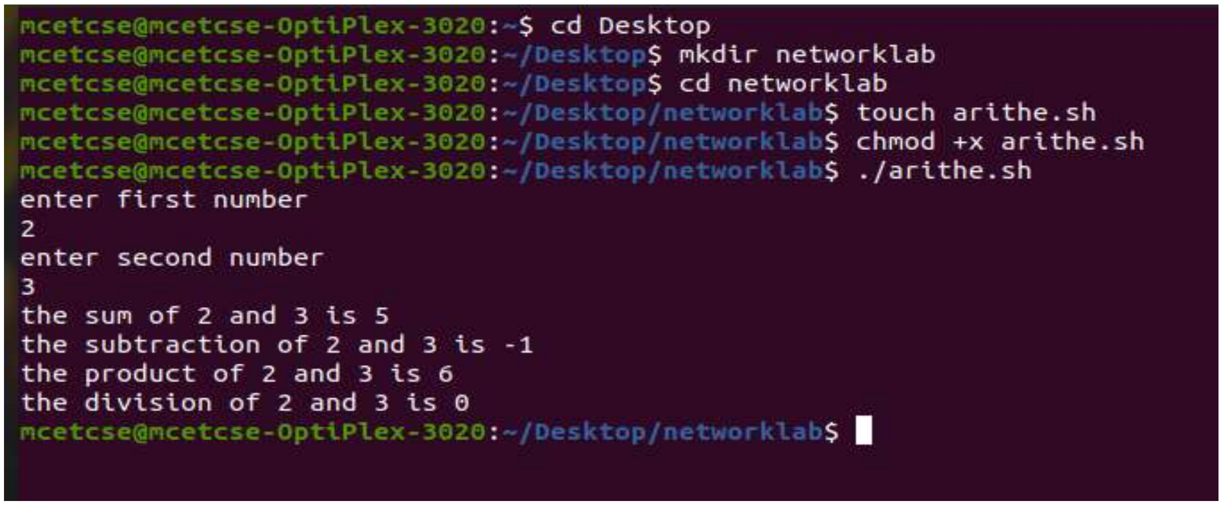
ALGORITHM

1. Start
2. Read the first number
3. Read second number
4. Find the sum
5. Find the difference
6. Find multiplication
7. Find the division
8. Display the result of sum
9. Display the result of difference
10. Display the result of multiplication
11. Display the result of division
12. Stop

PROGRAM:

```
echo"enter first number"
read a
echo"enter second number"
read b
add=`expr$a + $b`
sub=`expr$a - $b`
mul=`expr$a \* $b`
div=`expr$a / $b`
echo"the sum of $aand$bis$add"
echo"the subtraction of $aand$bis$sub"
echo"the product of $aand$bis$mul"
echo"the division of $aand$bis$div"
```

OUTPUT:



```
mcetcse@mcetcse-OptiPlex-3020:~$ cd Desktop
mcetcse@mcetcse-OptiPlex-3020:~/Desktop$ mkdir networklab
mcetcse@mcetcse-OptiPlex-3020:~/Desktop$ cd networklab
mcetcse@mcetcse-OptiPlex-3020:~/Desktop/networklab$ touch arithe.sh
mcetcse@mcetcse-OptiPlex-3020:~/Desktop/networklab$ chmod +x arithe.sh
mcetcse@mcetcse-OptiPlex-3020:~/Desktop/networklab$ ./arithe.sh
enter first number
2
enter second number
3
the sum of 2 and 3 is 5
the subtraction of 2 and 3 is -1
the product of 2 and 3 is 6
the division of 2 and 3 is 0
mcetcse@mcetcse-OptiPlex-3020:~/Desktop/networklab$
```

RESULT:

The shell Script program is executed successfully and the output is verified

STUDENT DETAILS

AIM:

Shell script program to print Student details.

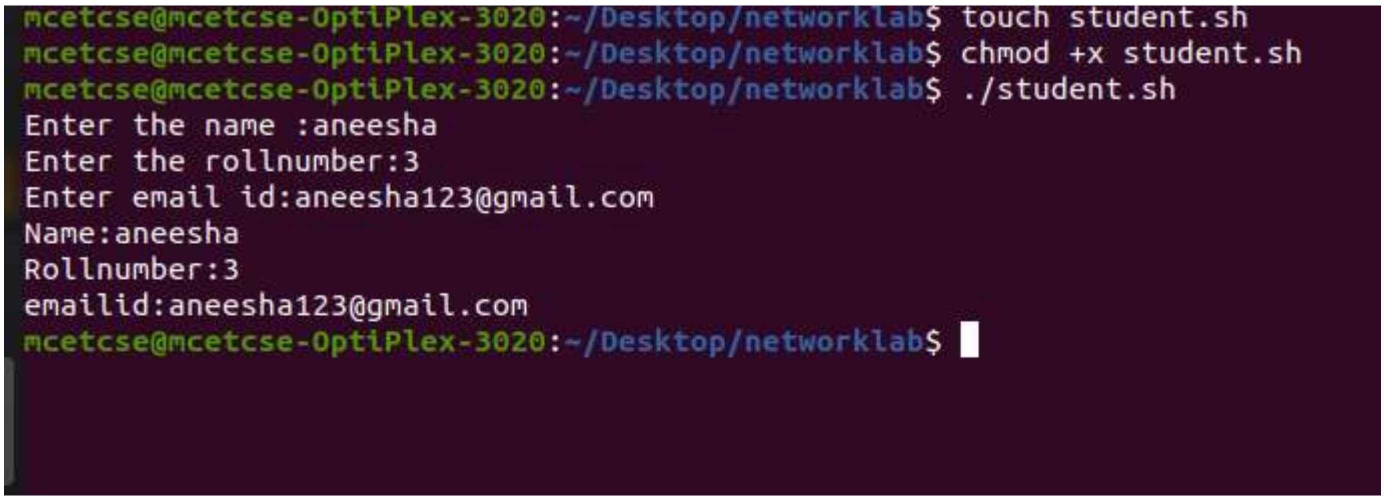
ALGORITHM

1. Start
2. Read the name of student
3. Read the roll number of student
4. Read the email id
5. Display the name of student
6. Display the name of roll number
7. Display the name of email id
8. Stop

SHELLSCRIPT:

```
echo-n Enter the name:
readp
echo-nEnter the roll number:
read q
echo-nEnter emailid:
readr
echo"Name:$p"
echo"Rollnumber:$q"
echo"emailid:$r"
```

OUTPUT:



```
mcetcse@mcetcse-OptiPlex-3020:~/Desktop/networklab$ touch student.sh
mcetcse@mcetcse-OptiPlex-3020:~/Desktop/networklab$ chmod +x student.sh
mcetcse@mcetcse-OptiPlex-3020:~/Desktop/networklab$ ./student.sh
Enter the name :aneesha
Enter the rollnumber:3
Enter email id:aneesha123@gmail.com
Name:aneesha
Rollnumber:3
emailid:aneesha123@gmail.com
mcetcse@mcetcse-OptiPlex-3020:~/Desktop/networklab$ 
```

RESULT:

Program successfully executed

LARGESTOFTWONUMBERS

AIM:

Shell script program to find the larger of two numbers

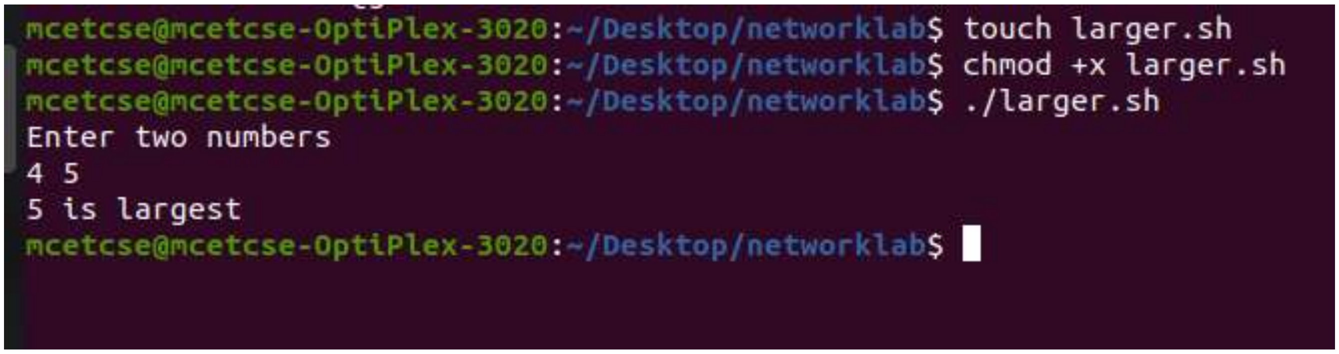
ALGORITHM

```
1.Start
2.Read the two numbers a and b
3. Check the a is greater than  b
4.Display a is greater
5. Otherwise Display b is greater
6.Stop
```

SHELLSCRIPT:

```
echo"Enter two numbers"
read a b
if [ $a  -gt $b]
then
echo"$a is largest"
else
echo"$b is largest"
fi
```

OUTPUT*:*



```
mcetcse@mcetcse-OptiPlex-3020:~/Desktop/networklab$ touch larger.sh
mcetcse@mcetcse-OptiPlex-3020:~/Desktop/networklab$ chmod +x larger.sh
mcetcse@mcetcse-OptiPlex-3020:~/Desktop/networklab$ ./larger.sh
Enter two numbers
4 5
5 is largest
mcetcse@mcetcse-OptiPlex-3020:~/Desktop/networklab$
```

RESULT:

The program is executed successfully.

GRADEOFASTUDENT

AIM:

Write a shell script program to find the grade of a student.

ALGORITHM

```
1.  Start
2.  Read the name of student
3.  Read the roll no of student
4.  Read the percentage of student
5.  Check the percentage is greater than 90 and less than 100,then display  Grade is A.
6.  Otherwise  Check the percentage is greater than 80 and less than 89,
    then display  Grade is B.
7.  Again Check the percentage is greater than 70 and less than 79,
    then display  Grade is C.
8.  Again Check the percentage is greater than 60 and less than 69,
    then display  Grade is D.
9.  Stop
```

PROGRAM:

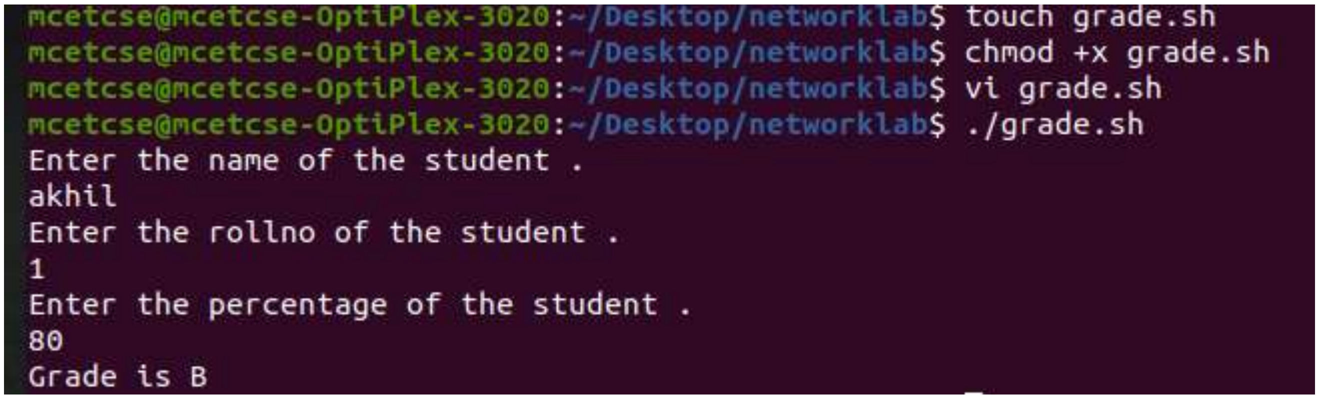
```
echo Enter the name of the student
read name
echo Enter the rollno of the student
read rollno
echo Enter the percentage of the student
read x
if [ $x –ge 90 –a $x –lt 100 ]
then
echo Grade is A
elif [ $x –ge 80 –a $x –lt 89 ]

then
echo Grade is B
elif [$x –ge 70 –a $x –lt 79]

then
echo Grade is C
elif [$x –ge 60 –a $x –lt 69]
```

```
then
echo Grade is D
else
echo Failed
fi
```

OUTPUT:



```
mcetcse@mcetcse-OptiPlex-3020:~/Desktop/networklab$ touch grade.sh
mcetcse@mcetcse-OptiPlex-3020:~/Desktop/networklab$ chmod +x grade.sh
mcetcse@mcetcse-OptiPlex-3020:~/Desktop/networklab$ vi grade.sh
mcetcse@mcetcse-OptiPlex-3020:~/Desktop/networklab$ ./grade.sh
Enter the name of the student .
akhil
Enter the rollno of the student .
1
Enter the percentage of the student .
80
Grade is B
```

RESULT:

The shell script was successfully executed and output is verified.

MULTIPLICATIONTABLE

AIM:

To read a number and print the multiplication table

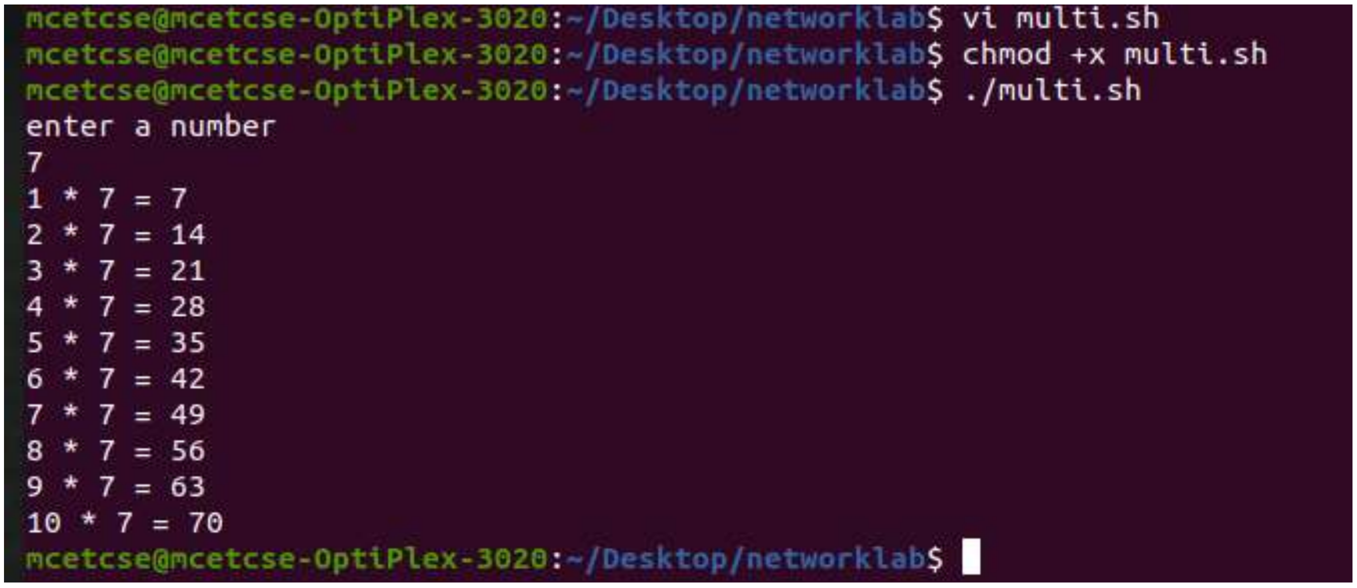
ALGORITHM

```
1.Start
2.Read the number
3.Initialize i =1
4.Check i is less than 10
5.Find the multiplication table
6.Display the result
7.Stop
```

PROGRAM

```
echo enter a number
read a
i=1
wile [ $ i -le 10]
do
res=$( expr$i \* $a)
echo $i"*"$a"="$res
i=$(expr $i + 1)
done
```

OUTPUT



```
mcetcse@mcetcse-OptiPlex-3020:~/Desktop/networklab$ vi multi.sh
mcetcse@mcetcse-OptiPlex-3020:~/Desktop/networklab$ chmod +x multi.sh
mcetcse@mcetcse-OptiPlex-3020:~/Desktop/networklab$ ./multi.sh
enter a number
7
1 * 7 = 7
2 * 7 = 14
3 * 7 = 21
4 * 7 = 28
5 * 7 = 35
6 * 7 = 42
7 * 7 = 49
8 * 7 = 56
9 * 7 = 63
10 * 7 = 70
mcetcse@mcetcse-OptiPlex-3020:~/Desktop/networklab$ 
```

RESULT

The shell script program is executed successfully and output is verified.

PRIMEORNOT

AIM:

Shell script program to find given number is prime or not

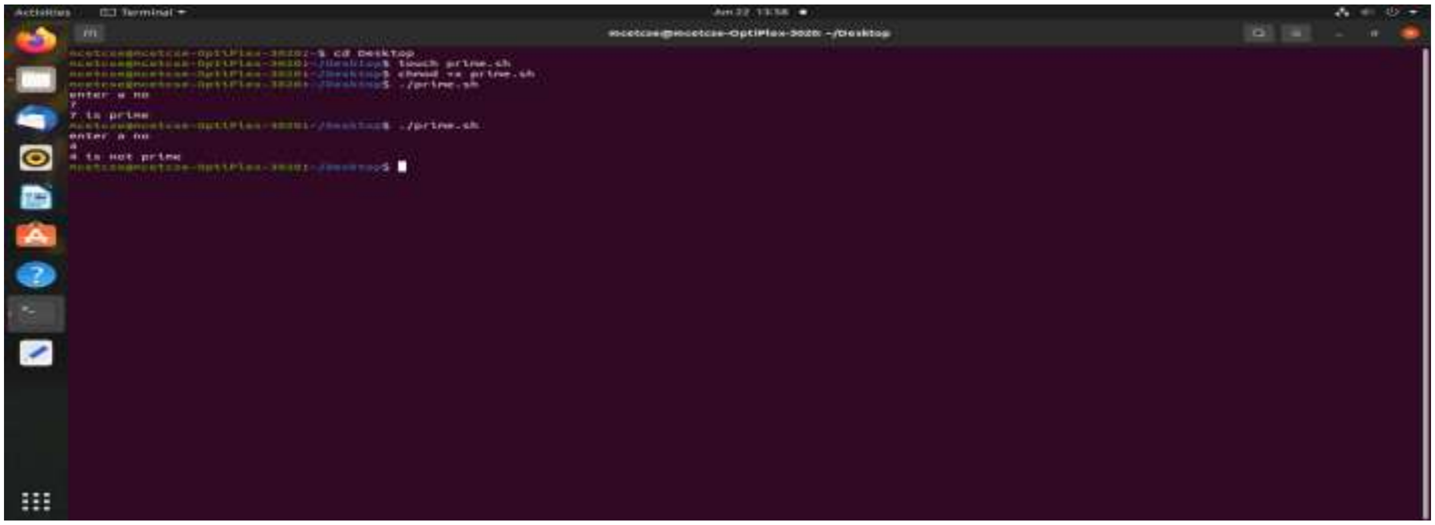
ALGORITHM

1. Start
2. Read the number num
3. Initialize the p=0
4. Initialize the iterator i =2
5. Iterate a while with condition ,i <= num/2
6. Check num is divisible by loop iterator ,then increment p
7. If the p is equal to 0
   Dispaly number is prime
8. Otherwise display number is not prime
9. Stop

PROGRAM:

```
echo enter a number
read n
i=2 p=0
a=`expr$n / 2`
while  [ $i –le $a ]
do
b=`expr$n % $i`
if[ $b eq 0 ]
then
p=1
break
fi
i=`expr$i + 1`
done
if [ $p  -eq 0]
then
echo $n is prime
else
echo $n is not prime
fi
```

<u>OUPUT</u>



```
mcetcse@mcetcse-OptiPlex-3020:~$ cd Desktop
mcetcse@mcetcse-OptiPlex-3020:~/Desktop$ touch prime.sh
mcetcse@mcetcse-OptiPlex-3020:~/Desktop$ chmod +x prime.sh
mcetcse@mcetcse-OptiPlex-3020:~/Desktop$ ./prime.sh
enter a no
7
7 is prime
mcetcse@mcetcse-OptiPlex-3020:~/Desktop$ ./prime.sh
enter a no
4
4 is not prime
mcetcse@mcetcse-OptiPlex-3020:~/Desktop$
```

<u>RESULT</u>

The Shell script program executed successfully and output is verified.

ARMSTRONG OR NOT

AIM

Shell script program to check whether the number is Armstrong or not

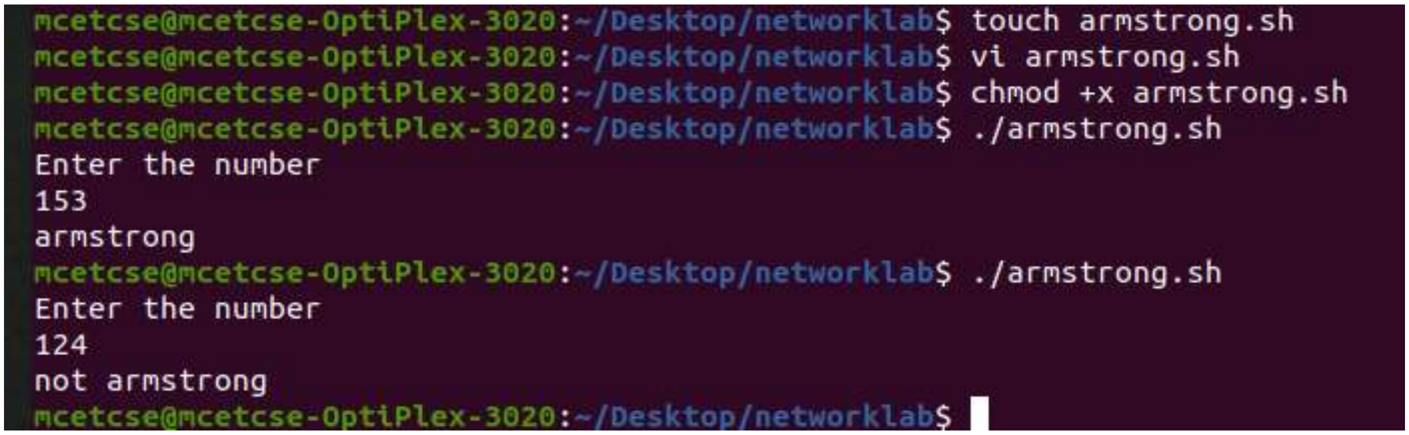
ALGORITHM

```
1. Start
2. Read the number
3.Initialize sum=0 and b=a.
4.Find the total number of digits in the number.
5.Repeat until (a != 0)
    5.1 Find the remainder d=  a% 10
    5.2 Find the result = sum+ d*d*d
    5.3Find the division a= a/10
9.if (result == number)
10.DisplayArmstrong
11 .Otherwise display Not Armstrong
12.Stop
```

SHELLSCRIPT:

```
echo Enter the number
read a
b=$a
s=0
while[ $a –gt 0]
do
d=`expr$a % 10`
s=`expr$s + $d \* $d \* $d`
a=`expr$a / 10`
done
if [ $s –eq $b ]
then
echo armstrong
else
echo not armstrong
fi
```

OUTPUT



```
mcetcse@mcetcse-OptiPlex-3020:~/Desktop/networklab$ touch armstrong.sh
mcetcse@mcetcse-OptiPlex-3020:~/Desktop/networklab$ vi armstrong.sh
mcetcse@mcetcse-OptiPlex-3020:~/Desktop/networklab$ chmod +x armstrong.sh
mcetcse@mcetcse-OptiPlex-3020:~/Desktop/networklab$ ./armstrong.sh
Enter the number
153
armstrong
mcetcse@mcetcse-OptiPlex-3020:~/Desktop/networklab$ ./armstrong.sh
Enter the number
124
not armstrong
mcetcse@mcetcse-OptiPlex-3020:~/Desktop/networklab$
```

RESULT

The Shell script program executed successfully and output isverified.