INDEX

SL NO.		NAME OF EXPERIMENT	PAGE NO.	REMARK
I		BASIC COMMANDS IN LINUX		
	SHELL PROGRAMS			
1		FIND LARGEST AMONG THREE NUMBER		
2		CONCATENATE TWO FILES		
3		CONTENTS OF A FILE INTO UPPERCASE		
4		CALCULATE THE PERCENTAGE OF MARKS AND DISPLAY GRADE		
5		REVERSE THE CONTENTS OF A FILE		
6		CHECK WHETHER LEAP YEAR OR NOT		
7		SWAP TWO NUMBERS		
8		FIND SUM OF DIGITS AND REVERSE OF A NUMBER, CHECK WHETHER IT IS PALINDROME		
9		PRINT CALENDER		
10		MENU DRIVEN PROGRAM		
11		LIST OF FILES HAVING READ, WRITE AND EXECUTE PERMISSIONS		
12		PRIME NUMBER		
13		CHECK FILE OR DIRECTORY		
14		FIBONACCI SERIES		
15		REVERSE AND PALINDROME		
16		PRINT THE PATTERN 1 1 2 1 2 3		
17		MULTIPLICATION TABLE		
18		COUNT THE WORD, CHARACTER, WHITE SPACE AND SPECIAL CHARACTER		

BASIC COMMANDS IN LINUX

1. ls

List information about the FILEs (the current directory by default).

Syntax

```
ls [OPTION]... [FILE]...
```

Option Description

-l use a long listing format

Example:

```
$ ls -l
```

Output:

```
total 8
drwxrwxr-x 2 it it 80 Aug 20 12:13 linux
-rw-rw-r-- 1 it it 9 Aug 20 12:10
LinuxLabCycle.docx drwxrwxr-x 2 it it 40 Aug 20
12:02 networking drwxrwxr-x 2 it it 40 Aug 20
12:01 php
-rw-rw-r-- 1 it it 10 Aug 20 12:11 PhpLabCycle.docx
```

-r reverse order while sorting

Example::

```
$ ls -r
```

Output::

```
PhpLabCycle.docx php networking LinuxLabCycle.docx
linux
```

-R list subdirectories recursively

Example:

```
ls -R
```

Output:

```
./linux:
mod1.docx  mod2.pdf
./networking:
./php:
```

-s print the allocated size of each file, in blocks

Example:

```
ls -s
```

Output:

```
total 8
0 linux 4 LinuxLabCycle.docx 0 networking 0 php 4
PhpLabCycle.docx
```

-c sort by ctime, newest first

Example:

```
ls -c
```

Output:

```
linux PhpLabCycle.docx LinuxLabCycle.docx networking
php
```

-m fill width with a comma separated list of entries

Example:

```
ls -m
```

Output:

```
linux, LinuxLabCycle.docx, networking, php,
PhpLabCycle.docx
```

2.grep

grep, egrep, fgrep, rgrep - print lines matching a pattern

Syntax

```
grep [OPTIONS] PATTERN [FILE...]
```

Option Description

-v Print the version number of grep to the standard output stream.

Example:

```
$ grep -v a f1
```

Output:

```
A
N
j
u

d
E
E
p
u
```

```
A
C
H
U

A
n
i
L
```

-c Suppress normal Output; instead print a count of matching lines for each input file.

Example:

```
~$ grep -c A f1
```

Output:

```
3
```

-I Ignore case distinctions in both the PATTERN and the inputFiles.

Example:

```
~$ grep -i e f1
```

Outpt:

```
m
e
e
r
a

m
e
e
r
a

d
E
E
p
u
```

-o Print only the matched (non-empty) parts of a matching line.

Example:

```
~$ grep ra -o f1
```

```
O
u
t
p
u
```

```
t
:

r
a

r
a
```

-n Prefix each line of Output: with the 1-based line number within its input file.

Example:

```
~$ grep -n a f1
```

Output:

```
1:meera
2:Susan
3:meera
6:aCHU
```

-l Suppress normal Output; instead print the name of each input file from which Output:would normally have beenprinted.

Example:

```
~$ grep -l a f1 f2
```

Output:

```
f1
f2
```

-L Suppress normal output; instead print the name of each input file from which no output would normally have been printed.

Example:

```
~$ grep -L meera f1 f2
```

Output:

```
f2
```

-w Select only those lines containing matches that form whole words.

Example:

```
~$ grep -w meera f1 f2
```

Output:

```
f1:meera
f1:meera
```

3.rm

Remove files or directories

Syntax

```
rm [OPTION]... FILE...
```

Option Description

-I prompt before every removal.

Example:

```
~$ rm -i f3
```

Output:

```
rm: remove regular file „f3"? y
```

-f ignore nonexistent files and arguments, never prompt.

Example:

```
~$ rm -f f5
```

-r remove directories and their contents recursively.

Example:

```
~$ rm -v -r flag
```

Output:

```
removed directory: „flag"
```

-v explain what is being done.

Example:

```
~$ rm -v f2
```

Output:

```
removed „f2"
```

4. wc

print newline, word, and byte counts for each file.

Syntax

```
wc [OPTION]... [FILE]...
```

Option Description

-c print the byte counts.

Example:

```
~$ wc -c f1
```

Output:

```
44 f1
```

-m print the character counts.

Example:

```
~$ wc -m f2
```

Output:

```
19 f2
```

-l print the newline counts.

Example:

```
~$ wc -l f2
```

Output:

```
7 f2
```

-L print the length of the longest line.

Example:

```
~$ wc -L f1
```

Output:

```
5 f1
```

-w print the word counts

Example:

```
~$ wc -w f1
```

Output:

```
8 f1
```

5.pwd

print name of current/working directory.

Syntax

```
pwd [OPTION]...
```

Example:

```
~$ pwd
```

Output:

```
/home/chinjunv
```

6. cat

Concatenate files and print on the standard output.

Syntax

```
cat [OPTION]... [FILE]...
```

Option Description

-A equivalent to -v

Example:

```
~$ cat -A f2
```

Outp

```
u
t
:

a
n
u
$
go
vi
nd
$
ra
dh
u$
ji
ji
$
lo
li
tt
a$
ap
pu
$
```

-b number nonempty output lines, overrides –n.

Example:

```
~$ cat -b f2
```

Output:

```
1    anu
```

```
      2     govind
      3     radhu
      4     jiji
      5     lolitta
```

-s suppress repeated empty output lines.

Example:

```
      ~$ cat -s f4
```

Output:

```
      devu
      sainu
      binu
```

-E display $ at end of each line.

Example:

```
      ~$ cat -E f1
```

Output:

```
      m
      e
      e
      r
      a
      $

      S
      u
      s
      a
      n
      $

      m
      e
      e
      r
      a
      $

      A
      N
      j
      u
      $

      d
      E
      E
      p
      u
      $

      a
```

```
C
H
U
$

A
C
H
U
$

A
n
i
L
$
```

-n number all output lines.

Example:

```
~$ cat -n f2
```

Output:

```
1    anu
2    govind
3    radhu
4    jiji
5    lolitta
```

7.date

print or set the system date and time.

Syntax

```
date [OPTION]... [+FORMAT]
```

Option Description

-d display time described by STRING, not 'now'.

Example:

```
~$ date --date='15 days ago'
```

Output:

```
Fri Sep 5 13:22:39 IST 2014
```

-r display the last modification time of FILE.

Example:

```
~$ date -r f3
```

Output:

```
Sat Sep 20 13:12:50 IST 2014
```

-u print or set Coordinated Universal Time.

Example:

```
~$ date –u
```

Output:

```
Sat Sep 20 07:54:11 UTC 2014
```

-R output date and time in RFC 2822 format.

Example:

```
~$ date -R
```

Output:

```
Sat, 20 Sep 2014 13:25:16 +0530
```

8. mv

move (rename) files

Syntax

```
mv [OPTION]... [-T] SOURCE DEST
```

Option Description

-l use a long listing format

Example:

```
~$ mv f newfile
```

-f do not prompt before overwriting

Example:

```
~$ mv –f newfile
```

-I prompt before overwrite

Example:

```
~$ mv –i f newfile
```

Output:

```
mv: overwrite „newfile"? y
```

-n do not overwrite an existing file

Example:

```
mv –n newfile file
```

-v explain what is being done

Example:

```
~$ mv -v  filelinux
```

Output:

```
„file" -> „linux"
```

9. cat

concatenate files and print on the standard output

Syntax

```
cat [OPTION]... [FILE]...
```

Example:

```
~$ cat linux
```

Outp
u
t
:

```
f
i
l
e
dir
ect
ory
bas
h
a
s
h

z
s
h
```

Option Description

-A show all

Example:

```
~$ cat -A linux
```

Outpu
t
:

```
f
i
l
e
$
dire
```

```
ctor
y$
bash
$
ash$
```

-b number nonempty output lines, overrides –n

Example:

```
~$ cat -b flower
```

Output:

```
1    rose

2    jasmine
3    lotus

4    lilly
```

-s suppress repeated empty output lines

Example:

```
~$ cat -b -s flower
```

Output:

```
1    rose

2    jasmine
3    lotus

4    lilly
```

-n number all output lines

Example:

```
~$ cat -n flower
```

Output:

```
1    rose
2
3
4
5    jasmine
6    lotus
7
8
9    lilly
10
```

-E display $ at end of each line

Example:

```
$ cat -E flower
```

Outpu
t
:

```
r
o
s
e
$
$
$
$
ja
sm
in
e$
lo
tu
s$
$
$
lilly$
$
```

10. head

output the first part of files

Syntax

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

Option Description

-c print the first K bytes of each file

Example:

```
~$ head -c 10 linux
```

Output:

```
File
```

-n print the first K lines instead of the first 10

Example:

```
~$ head -2 linux
```

Outp
u
t
:

```
f
i
l
e
directory
```

-q never print headers giving file names

Example:

```
~$ head -q linux
```

Outp

```
u
t
:

f
i
l
e
dir
ect
ory
bas
h
ash
```

11. tail

output the last part of files

Syntax

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

Option Description

-c output the last K bytes

Example:

```
~$ cat linux
```

Outp

```
u
t
:

f
i
l
e
dir
ect
ory
bas
```

```
h
ash
```

Example:

```
~$ tail -c 3 linux
```

Output:

```
sh
```

-f outputappended data as the file grows

Example:

```
~$ tail –f linux
```

Outp

```
u
t
:

f
i
l
e
dir
ect
ory
bas
h
ash
new data
```

-n output the last K lines

Example:

```
~$ tail –3 linux
```

Output:

```
dir
ect
ory
bas
h
ash
```

12. paste

merge lines of files

Syntax

```
paste [OPTION]... [FILE]...
```

Option Description

-d reuse characters from LIST instead of TABs

Example:

```
~$ paste -d linux
```

Outp
u
t
:
n
e
w

n
e
w

e
x
a
m

e
x
a
m

-s paste one file at a time instead of in parallel

Example:
```
~$ paste -s linux
```
Output:
```
file directory bash ash
```

13. file

determine file type

Syntax

```
file [-bchiklLNnprsvz0] [--apple] [--mime-encoding] [--
mime-type][-e testname] [-F separator] [-f namefile] [-m
magicfiles] file ...
```

Option Description

-b do not prepend filenames to output lines (brief mode)

Example:
```
~$ file -b flower
```
Output:
```
ASCII text
```

-c cause a checking printout of the parsed form of the magic file

Example:

```
~$ file -c flower
```

Output:

```
cont offset    type opcode    mask value     desc
```

-e exclude the test named in testname from the list of tests made to determine the file type

Example:

```
~$ file -e ascii flower
```

Output:

```
flower: data
```

-F Use the specified string as the separator between the filename and the file result returned. Defaults to ':'

Example:

```
~$ file -F @ flower
```

Output:

```
flower@ ASCII text
```

14. chmod

change file mode bits

Syntax

```
chmod [OPTION]...  MODE[,MODE]...  FILE...
chmod [OPTION]...  OCTAL-MODE FILE  ...
chmod [OPTION]...  --reference=RFILE FILE...
```

Option Description

-v --verbose output a diagnostic for every file processed

Example:

```
chmod o-rf1
```

Output:

```
-rwxrw---- 1 lb233 lb233   10 Jan  8 16:10 f1
```

15.cp

copy files and directories

Syntax

```
cp [OPTION]... [-T] SOURCE DEST
cp [OPTION]... SOURCE... DIRECTORY
cp [OPTION]... -t DIRECTORY SOURCE...
```

Example:

```
cat>>file1
12
13

cat>
>fil
e2
flag
colo
r
```

Example:

```
cp file1
file2 cat
file2
```

Output:

```
12
13
```

16.tee

read from standard input and write to standard Output: and files

Syntax

```
tee [OPTION]... [FILE]...
```

Option Description

-a --append append to the given FILEs, do not overwrite

Example:

```
tee
-a
f1
app
le
ora
nge
```

Output:

```
1
2
3
4
5

a
p
p
l
e

o
r
a
```

```
n
g
e
```

17.tr

translate or delete characters

Syntax

```
tr [OPTION]... SET1 [SET2]
```

Example:

```
tr "[a-z]" "[A-Z]" <f2
```

Output:

```
A
B
C
D
E
```

18.echo

display a line of text

Syntax

```
echo [SHORT-OPTION]... [STRING]...

echo LONG-OPTION
```

Example:

```
echo sree narayana gurukulam
```

Output:

```
sree narayana gurukulam
```

19.sort

sort lines of text files

Syntax

```
sort [OPTION]... [FILE]...
```

Option Description

-n --numeric -sort compare according to string numerical value

Example:

```
cat>>f5
```

Output:

```
d
e
b
i
a
n

s
u
s
e

f
e
d
o
r
a
ma
nd
ri
va
re
dh
at
```

Example:

```
sort -n f5
```

Output:

```
d
e
b
i
a
n

f
e
d
o
r
a
ma
nd
ri
va
re
dh
at
su
se
```

20. find

find - search for files in a directory hierarchy

Syntax

```
find [-H] [-L] [-P] [-D debugopts] [-Olevel]
[path...] [expression]
```

Option Description

-d A synonym for -depth, for compatibility

Example:

```
find /home/mca/India -type d
```

Output:

```
/home/mca/India
/home/mca/India/bharat
```

Example:

```
find /home/mca -name f?
```

Output:

```
/home/mca/India/f1
/home/mca/India/f2
/home/mca/f4
```

21. pipe

the standard output of one command can be send on standard input to another command

Example:

cat f2 | tee mca

Output:

```
1. Unix
2. Redhat
3. Fedora
4. Open SUSE
```

Example:

```
cat f2 | head -3 | tail -1
```

Output:

```
3. Fedora
```

22.chown

chown - change file owner and group

Syntax

```
chown [OPTION]... [OWNER][:[GROUP]] FILE...
chown [OPTION]... --reference=RFILE FILE...
```

Option Description

-v	--verbose	output a diagnostic for every file processed
-R	--recursive	operate on files and directories recursively

Example:

```
sudo chown
mca f2 ls -l
```

Output:

```
-rw-rw-r-- 1 mca mca   41 Jan  8 20:39 f2
```


CYCLE-1

FIND LARGEST AMONG THREE NUMBERS

Aim: Write a shell program to find largest among three numbers

Program:

```
#!/bin/bash
echo "enter three numbers"
read num1
read num2
read num3
if [ $num1 -gt $num2 -a $num1 -gt $num3 ]
then
echo "$num1 is greater"
elif [ $num2 -gt $num3 ]
then
echo "$num2 is greater"
else
echo "$num3 is greater"
fi
```

output

```
school@school-presario-CQ57-Notebook-PC:~/Almin$ bash larger.sh
enter three numbers
12
100
13
100 is greater
```

CONCATENATE TWO FILES

Aim:Write a shell script to concatenate 2 files

Program:

```
#!/bin/bash
echo "enter file name1"
read f1
```

```
echo "enter filename2"
read f2
if test -e $f1 -a -e $f2
then
cat $f1 $f2 >f3
echo “File is:”
cat f3
else
echo "file not found"
fi
```

output

```
school@school-presario-CQ57-Notebook-PC:~/Almin$ bash concat.sh
enter file name1
f1
enter filename2
f2
File is:
a
b
c
d
1
2
3
4
```

CONTENTS OF A FILE INTO UPPERCASE

Aim:Write a shell script to convert the contents of a file into uppercase

Program:

```
#!/bin/bash
echo "enter the file name"
read f1
if test -e $f1
then
echo "contents of file"
cat $f1
echo "contents changed to uppercase"
tr '[a-z]' '[A-Z]' <$f1
```

```
else
echo "file not found"
fi
```

Output

```
school@school-presario-CQ57-Notebook-PC:~/Almin $ bash convert.sh
enter the file name
f1
contents of file
a
b
c
d
contents changed to uppercase
A
B
C
D
```

CALCULATE THE PERCENTAGE OF MARKS AND DISPLAY GRADE

Aim: Write a shell program to read marks scored in 3 subjects (out of 100). Calculate the percentage of marks & display the grade of n students based on the following conditions.

%	Grade
80-100	A
70-79	B
60-69	C
<=59	D(Failed)

Program:

```
#!/bin/bash
echo -n "enter the limit:"
read n
for((i=0;i<n;i++))
do
echo -n "enter the rollno:"
read rn
```

```
echo -n "enter the name:"
read name
echo -n "enter the marks:"
read m1 m2 m3
total=`expr $m1 + $m2 + $m3`
per=`expr "$total*100/300"|bc`
echo "MARK LIST"
echo "--------------"
echo "RollNo:$rn"
echo "Name:$name"
echo "Total:$total"
if [ $per -le 100 -a $per -ge 80 ]
then
echo "A grade"
elif [ $per -le 79 -a $per -ge 70 ]
then
echo "B grade"
elif [ $per -le 69 -a $per -ge 60 ]
then
echo "C grade"
else
echo "FAILED"
fi
done
```

Output

```
school@school-presario-CQ57-Notebook-PC:~/Almin $ bash mark.sh
enter the limit:2
enter the rollno:1
enter the name:Almin
enter the marks:
45 65 85
MARK LIST
--------------
```

RollNo:1

Name:Almin

Total:195

C grade

enter the rollno:2

enter the name:Aleena

enter the marks:

100 100 100

MARK LIST

RollNo:2

Name:Aleena

Total:300

A grade

REVERSE THE CONTENTS OF A FILE

Aim:Write a shell program to reverse the contents of a file.

Program:

```
#!/bin/bash
echo -n "Read a file:"
read file
if [ -e $file ]
then
        echo "Content of $file"
        cat $file
        echo "Reversed content"
        tac $file
else
```

```
        echo "File not existing"
fi
```

output

```
school@school-presario-CQ57-Notebook-PC:~/Almin $ bash reverse.sh
Read a file:f1
Content of f1
ab
cd
Reversed content
cd
ab
```

CHECK WHETHER LEAP YEAR OR NOT

Aim:Write a script that receives year as argument and check whether it is leap year. If no argument is given assume the current year.

Program:

```
#!bin/bash
yr=$1
if [ $# -eq 0 ]
then
        yr=`date +"%Y"`
fi
a=$(($yr % 4))
b=$(($yr%100))
c=$(($yr%400))
if [ $a -eq 0 -a $b -ne 0 ]
```

```
then
        echo "$yr is leap year"
elif [ $c -eq 0 ]
then
        echo "$yr is leap year "
else
        echo "$yr is not a leap year"
fi
```

Output

```
school@school-presario-CQ57-Notebook-PC:~/Almin $ bash leapyear 2016
2016 is leap year
school@school-presario-CQ57-Notebook-PC:~/Almin $ bash leapyear
2017 is not a leap year
```

CYCLE-2

SWAP TWO NUMBERS

Aim:Write a shell program to swap two numbers with and without a temporary variable.

Program:

```
#!/bin/bash

echo "Enter the two numbers"

read x y

a=$x

b=$y

echo "   Using Temporary Variable"

echo "Before swap"

echo "x=$x"

echo "y=$y"

t=$y

y=$x

x=$t

echo "After swap"

echo "x=$x"

echo "y=$y"

echo "   Without Using Temporary Variable"

echo "Before swap"

echo "x=$a"

echo "y=$b"

b=$((a+b))

a=$((b-a))

b=$((b-a))
```

```
echo "After swap"
echo "x=$a"
echo "y=$b"
```

output

```
school@school-presario-CQ57-Notebook-PC:~/Almin$ bash swap.sh
Enter the two numbers
100 200
 Using Temporary Variable
 Before swap
 x=100
 y=200
 After swap
 x=200
 y=100
 Without Using Temporary Variable
 Befoe swap
 x=100
 y=200
 After swap
 x=200
 y=100
```

FIND SUM OF DIGITS AND REVERSE OF A NUMBER,CHECK WHETHER IT IS PALINDROME

Aim: Write a shell program to find

a) Sum of digits of a number
b) Reverse of the number
c) Determine whether the given number is a palindrome

Program:

```
#!/bin/bash
i=1
while test $i -gt 0
do
        echo -e " 1.SUM OF DIGITS\n2.REVERSE OF A NUMBER\n
        3.PALINDROME\nEnter your choice"
        read ch
        case $ch in
        1) echo "Enter the Number"
        read a
        sum=0
        while test $a -gt 0
        do
                b=` expr $a % 10`
                sum=` expr $sum + $b`
                a=` expr $a / 10`
        done
        echo "Sum of Digits=$sum";;
        2) echo "Enter a Number"
        read c
        r=0
        while test $c -gt 0
        do
                d=` expr $c % 10`
                r=$(((r*10)+d))
                c=` expr $c / 10`
        done
        echo "Reverse=$r";;
        3) echo "Enter a Number"
        read n1
        re=0
         n2=$n1
        while test $n1 -gt 0
        do
                m=` expr $n1 % 10`
                re=$(((res*10)+m))
                n1=` expr $n1 / 10`
        done
        if test $n2 -eq $re
        then
                echo "Number is Palindrome"
```

```
        else
                echo "Number is not Palindrome"
        fi;;
        *) echo "Wrong Choice";;
        esac
        echo "Do you want to continue(y/n)"
        read ans
        if test $ans = "y"
        then
                i=` expr $i + 1`
        else
         i=0
        fi
done
```

Output

```
1.SUM OF DIGITS
2.REVERSE OF A NUMBER
3.PALINDROME
Enter your choice
1
Enter the Number
451
Sum of Digits=10
Do you want to continue(y/n)
y
1.SUM OF DIGITS
2.REVERSE OF A NUMBER
3.PALINDROME
Enter your choice
2
Enter a Number
345
Reverse=543
Do you want to continue(y/n)
y
1.SUM OF DIGITS
2.REVERSE OF A NUMBER
3.PALINDROME
Enter your choice
3
Enter a Number
121
Number is Palindrome
Do you want to continue(y/n)
n
```

PRINT CALENDER

Aim:Write a shell script which will print the calendar, accepting the month name and year as command line argument.

Program:

```
#!/bin/bash
m=$1
y=$2
cal $m $y
```

Output

```
school@school-presario-CQ57-Notebook-PC:~/Almin$bash cal.sh 3 1995
     March 1995
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
```

MENU DRIVEN PROGRAM

Aim: Write a menu driven program to display the following options.

- **Contents of /etc/password**
- **List of output of 'who'**
- **Present working directory**
- **Exit**

Program:

```
#!/bin/bash
i=1
while test $i -gt 0
do
        echo -e"1.Contents of /etc/password\n2.List of output of 'who'\n3.Present
        working directory\n4.Exit\nEnter your choice"
        read ch
        case $ch in
        1)cat /etc/passwd;;
        2)who -a;;
        3)pwd;;
        4)exit;;
        *)echo "Wrong choice";;
        esac
done
```

Output

```
school@school-presario-CQ57-Notebook-PC:~/Almin$bash menu.sh
1.Contents of /etc/password
2.List of output of 'who'
3.Present working directory
4.Exit
Enter your choice
3
/home/Almin
1.Contents of /etc/password
2.List of output of 'who'
3.Present working directory
4.Exit
Enter your choice
4
```

LIST OF FILES HAVING READ, WRITE AND EXECUTE PERMISSIONS

Aim:Write a shell script that displays a list of all the files in the current directory to which the user has read, write and execute permissions.

Program:

```
#!/bin/bash

ls -l | grep "^-rwx" | tr -s " " | cut -d " " -f 9
```

Output

```
school@school-presario-CQ57-Notebook-PC:~/Almin$bash per.sh

f1
```

PRIME NUMBER

Aim:Write a shell script to find out whether the given number is prime number or not.

Program:

```
#!/bin/bash
echo "Enter the number"
read n
i=2
s=0
while [ $i –le $((n/2)) ]
do
        if [ $((n%i)) -eq 0 ]
        then
                s=1
        fi
        i=$((i+1))
done
if [ $n –eq 1 ]
then
        echo "Not prime"
elseif [ $s –eq 0 ]
then
        echo "Not prime"
else
        echo “Prime Number”
fi
```

Output

```
school@school-presario-CQ57-Notebook-PC:~/Almin$bash prime.sh
Enter the number
2
Prime Number
```

CHECK FILE OR DIRECTORY

Aim:Write a shell script that receives any number of file names as arguments checks if every argument supplied is a file or a directory and reports accordingly. Whenever the argument is a file, the number of lines on it is also reported.

Program:

```
#!/bin/bash
while test $# -gt 0
do
        x=$1
        if test -f $x
        then
                echo "$x is a file"
                w=$(wc -l $x)
                echo "no :of lines=$w"
        elif test -d $x
        then
                echo "$x is a directory"
        else
        echo "$x:cannot find such file or directory"
        fi
        shift
done
```

Output

```
school@school-presario-CQ57-Notebook-PC:~/Almin$bash check.sh new f2 f3
new is a directory
f2:cannot find such file or directory
f3 is a file
no :of lines=8
```

FIBONACCI SERIES

Aim:Write a shell program to display Fibonacci series using any looping construct.

Program:

```
#!bin/bash
echo "Enter the limit"
read n
a=0
b=1
if [ $n -eq 0 ]
then
        echo "plz enter limit greater than zero"
elif [ $n -eq 1 ]
then
        echo $a
else
        echo “Fibonacci…”
        echo –n $a
        echo –n $b
        for((i=2;i<$n;i++))
        do
                c=`expr $a + $b`
                echo -n $c
                a=$b
                b=$c
        done
fi
```

Output

```
school@school-presario-CQ57-Notebook-PC:~/Almin$bash fibonacci.sh
Enter the limit
5
Fibonacci…
0 1 1 2 3
```

REVERSE AND PALINDROME

Aim:Write a shell program to find out reverse string of the given string and check the given string is palindrome or not.

Program:

```
#!bin/bash
echo "Enter the string"
read s
echo "Reverse of the string is"
str=`expr $s | rev`
echo $str
if [ $s = $str ]
then
echo "String is palindrome"
else
echo "String is not palindrome"
fi
```

Output

```
school@school-presario-CQ57-Notebook-PC:~/Almin$bash pal.sh
Enter the string
malayalam
Reverse of the string is
malayalam
String is palindrome
```

PRINT THE PATTERN

Aim:Write a shell script to print the following patterns

1

1 2

1 2 3

1 2 3 4

Program:

```
#!bin/bash
echo "Enter number of raws"
read n
echo “Pattern is…”
for((i=1;$i<=$n;i++))
do
        for((j=I;$j<=$i;j++))
        do
                echo –ne “$j\t”
        done
        echo
done
```

Output

```
school@school-presario-CQ57-Notebook-PC:~/Almin$bash pattern.sh
Enter number of rows
5
Pattern is…
1
1       2
1       2       3
1       2       3       4
1       2       3       4       5
```

CYCLE-3

MULTIPLICATION TABLE

Aim: Write a shell program that takes a number as command line argument and prints its table in below format:

2 * 1 = 2

2 * 2 = 4

…

2 * 10 = 20

Program:

```
#!bin/bash
If [ $# -eq 0 ]
then
        Echo "Please enter argument"
else
        while [ $# -ne 0 ]
        do
                echo "Multiplication table of $1"
                i=1
                a=$1
                while [ $i –ne 11 ]
                do
                        s=$((a+i))
                        echo "$1 * $i = $s"
                        i=$((i+1))
                done
                shift
        done
fi
```

Output

```
school@school-presario-CQ57-Notebook-PC:~/Almin$ bash table 2
Multiplication table of 2
2 * 1 = 2
2 * 2 = 4
2 * 3 = 6
2 * 4 = 8
2 * 5 = 10
2 * 6 = 12
2 * 7 = 14
2 * 8 = 16
2 * 9 = 18
2 * 10 = 20
```

COUNT THE WORD, CHARACTER, WHITE SPACE & SPECIAL CHARACTER

Aim: Write a shell program to count number of word, character, white space & special symbol in a given text.

Program:

```
#!/bin/bash
echo "Enter the line"
read s
ws=0
sp=0
w=0
c=0
w=` echo $s|wc –w `
n=` echo $s|wc –c `
for((i=1;i<=n;i++))
do
        ch=` echo $s|cut –c $i `
        case $ch in
                " ")ws=$((ws+1));;
                [^Aa0-Zz9])sp=$((sp+1));;
        esac
done
echo "Number of words : $w"
echo "Number of white space: $ws"
echo "Number of special character: $sp"
echo "Number of character: $((n-ws-sp))"
```

Output

```
school@school-presario-CQ57-Notebook-PC:~/Almin $ bash count.sh
```

Aa Bb 12 $%

Number of words: 4

Number of whitespace: 3

Number of special character: 2

Number of character: 7