

LAB PROGRAMS:

PROGRAM1: Shell script to find if the given year is leap or not

CODE:

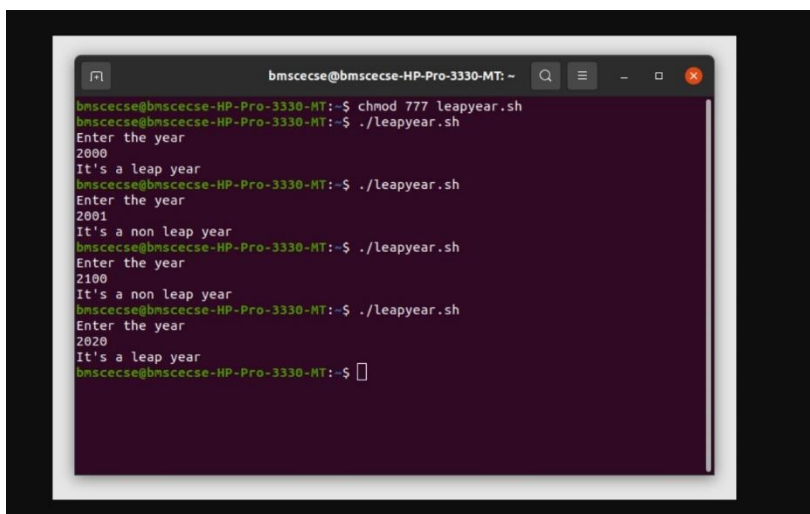
```
#!/bin/sh

echo "Enter the year "

read year

if [ $((year%400)) -eq 0 ]
then
    echo "It's a leap year"
elif [ $((year%4)) -eq 0 ]
then
    if [ $((year%100)) -eq 0 ]
    then
        echo "It's a non leap year"
    else
        echo "It's a leap year "
    fi
else
    echo "It's a non leap year "
fi
```

OUTPUT:

A screenshot of a terminal window with a dark purple background. The window title is 'bmscsecse@bmscsecse-HP-Pro-3330-MT: ~'. The terminal shows the following sequence of commands and outputs: 1. Command: 'chmod 777 leapyear.sh'. Output: 'bmscsecse@bmscsecse-HP-Pro-3330-MT:~\$ chmod 777 leapyear.sh'. 2. Command: './leapyear.sh'. Output: 'bmscsecse@bmscsecse-HP-Pro-3330-MT:~\$./leapyear.sh', 'Enter the year', '2000', 'It's a leap year'. 3. Command: './leapyear.sh'. Output: 'bmscsecse@bmscsecse-HP-Pro-3330-MT:~\$./leapyear.sh', 'Enter the year', '2001', 'It's a non leap year'. 4. Command: './leapyear.sh'. Output: 'bmscsecse@bmscsecse-HP-Pro-3330-MT:~\$./leapyear.sh', 'Enter the year', '2100', 'It's a non leap year'. 5. Command: './leapyear.sh'. Output: 'bmscsecse@bmscsecse-HP-Pro-3330-MT:~\$./leapyear.sh', 'Enter the year', '2020', 'It's a leap year'. The prompt 'bmscsecse@bmscsecse-HP-Pro-3330-MT:~\$' is shown at the end of the last line.

PROGRAM2: Shell script to find the area of a circle

CODE:

```
#!/bin/sh

echo "Enter the radius of the circle "

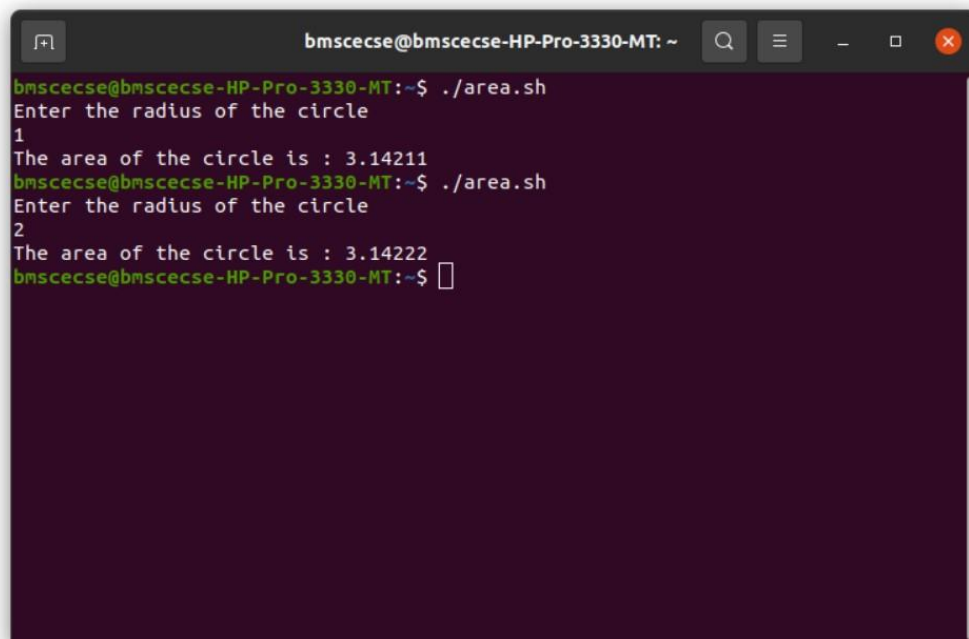
read r

pi=3.142

area=`echo $pi\$r\$r|bc`

echo "The area of the circle is : $area"
```

OUTPUT:

A terminal window with a dark purple background and white text. The window title is 'bmscecse@bmscecse-HP-Pro-3330-MT: ~'. The terminal shows the execution of a shell script named 'area.sh'. The prompt is 'bmscecse@bmscecse-HP-Pro-3330-MT:~\$./area.sh'. The script prompts 'Enter the radius of the circle' and receives input '1'. It then outputs 'The area of the circle is : 3.14211'. The prompt is 'bmscecse@bmscecse-HP-Pro-3330-MT:~\$./area.sh'. The script prompts 'Enter the radius of the circle' and receives input '2'. It then outputs 'The area of the circle is : 3.14222'. The prompt is 'bmscecse@bmscecse-HP-Pro-3330-MT:~\$' followed by a cursor.

```
bmscecse@bmscecse-HP-Pro-3330-MT:~$ ./area.sh
Enter the radius of the circle
1
The area of the circle is : 3.14211
bmscecse@bmscecse-HP-Pro-3330-MT:~$ ./area.sh
Enter the radius of the circle
2
The area of the circle is : 3.14222
bmscecse@bmscecse-HP-Pro-3330-MT:~$
```

PROGRAM3: Shell script to check whether the number is zero/ positive/ negative

CODE:

```
#!/bin/sh

echo "Enter the number "

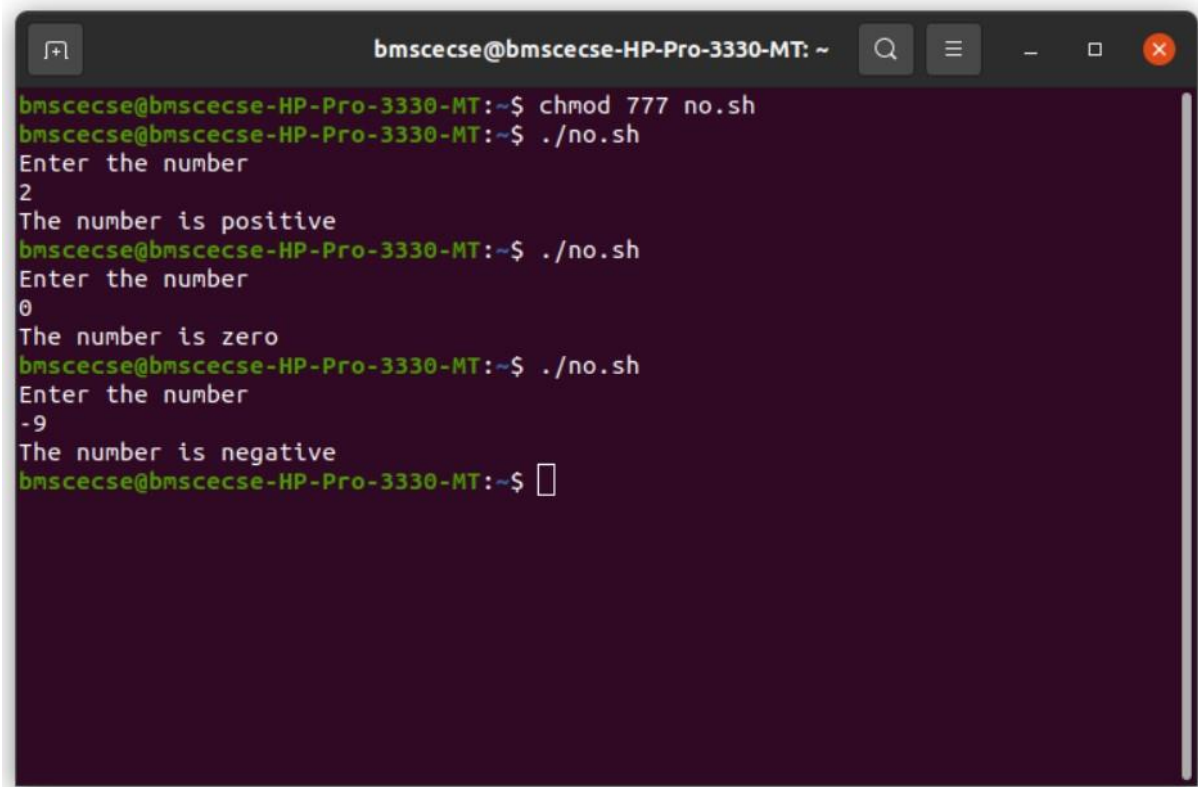
read num

if [ $num -eq 0 ]
then

    echo "The number is zero "
```

```
elif [ $num -lt 0 ]
then
    echo "The number is negative "
else
    echo "The number is positive"
fi
```

OUTPUT:



```
bmscecse@bmscecse-HP-Pro-3330-MT: ~  
bmscecse@bmscecse-HP-Pro-3330-MT:~$ chmod 777 no.sh  
bmscecse@bmscecse-HP-Pro-3330-MT:~$ ./no.sh  
Enter the number  
2  
The number is positive  
bmscecse@bmscecse-HP-Pro-3330-MT:~$ ./no.sh  
Enter the number  
0  
The number is zero  
bmscecse@bmscecse-HP-Pro-3330-MT:~$ ./no.sh  
Enter the number  
-9  
The number is negative  
bmscecse@bmscecse-HP-Pro-3330-MT:~$
```

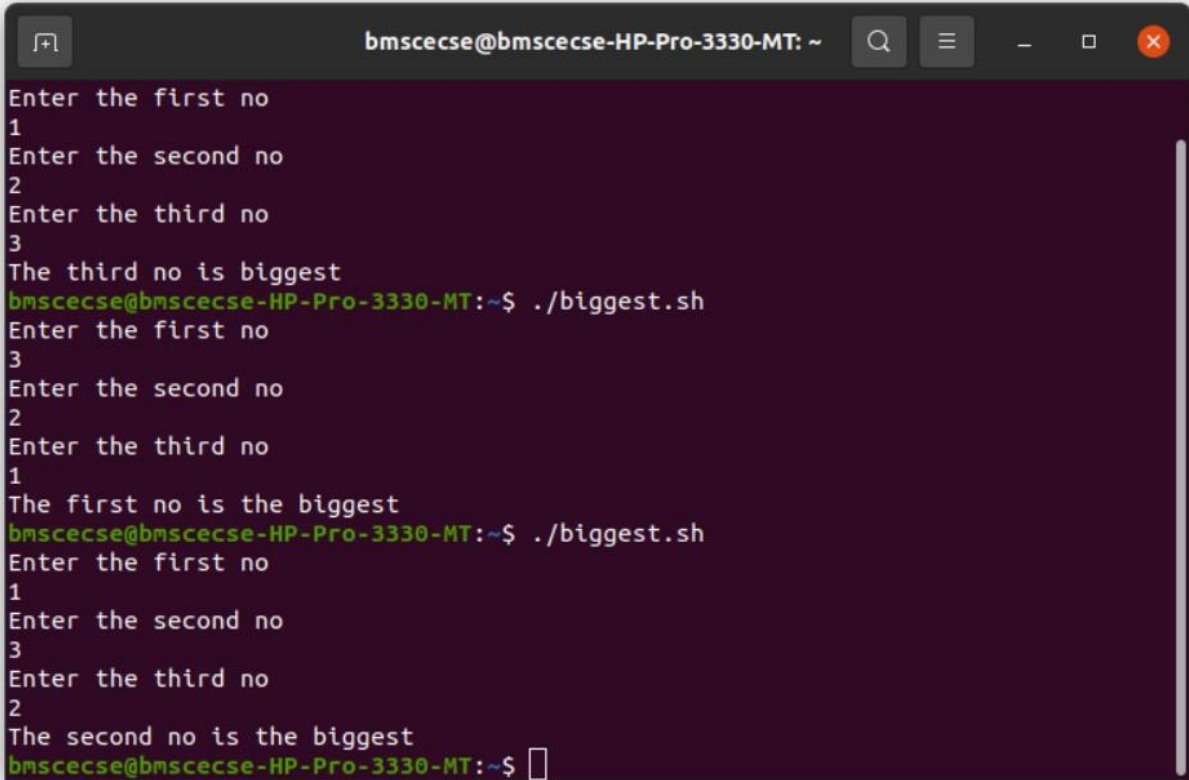
PROGRAM4: Shell script to find the biggest of three numbers

CODE:

```
#!/bin/sh  
  
echo "Enter the first no "  
read f  
  
echo "Enter the second no"  
read s  
  
echo "Enter the third no "  
read t
```

```
if [ $f -gt $s -a $f -gt $t ]
then
    echo "The first no is the biggest "
elif [ $s -gt $f -a $s -gt $t ]
then
    echo "The second no is the biggest "
else
    echo "The third no is biggest"
fi
```

OUTPUT:

A terminal window titled 'bmscecse@bmscecse-HP-Pro-3330-MT: ~' with standard window controls. The terminal shows the execution of a script named 'biggest.sh'. In the first run, inputs are 1, 2, and 3, resulting in the output 'The third no is biggest'. In the second run, inputs are 3, 2, and 1, resulting in 'The first no is the biggest'. In the third run, inputs are 1, 3, and 2, resulting in 'The second no is the biggest'. The prompt is '~\$' and the cursor is at the end of the last line.

```
bmscecse@bmscecse-HP-Pro-3330-MT: ~  
Enter the first no  
1  
Enter the second no  
2  
Enter the third no  
3  
The third no is biggest  
bmscecse@bmscecse-HP-Pro-3330-MT:~$ ./biggest.sh  
Enter the first no  
3  
Enter the second no  
2  
Enter the third no  
1  
The first no is the biggest  
bmscecse@bmscecse-HP-Pro-3330-MT:~$ ./biggest.sh  
Enter the first no  
1  
Enter the second no  
3  
Enter the third no  
2  
The second no is the biggest  
bmscecse@bmscecse-HP-Pro-3330-MT:~$
```

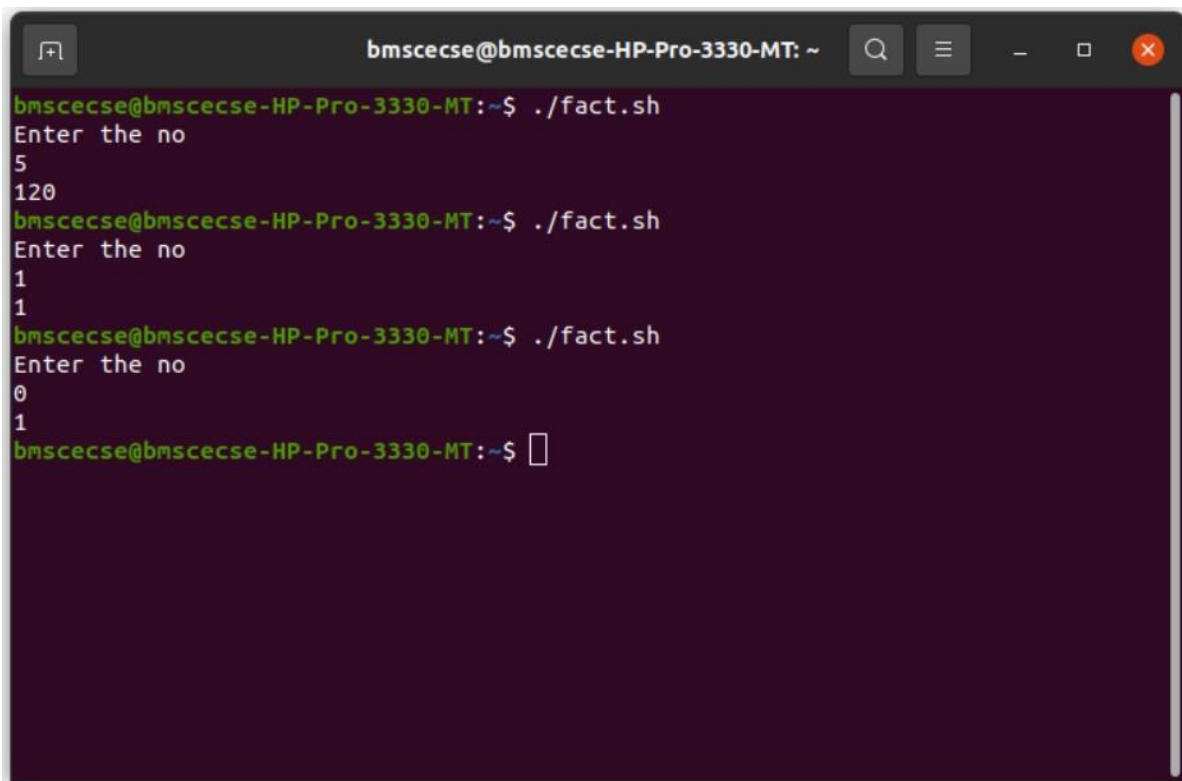
PROGRAM5: Shell script to find the factorial of a number

CODE:

```
#!/bin/bash  
  
echo "Enter the no "  
  
read no  
  
st=1
```

```
fact=1
for (( c=$st; c<=$no; c++))
do
    fact=`expr $c\*$fact|bc`
done
echo "factorial is "$fact
```

OUTPUT:

A terminal window titled 'bmscecse@bmscecse-HP-Pro-3330-MT: ~' with standard window controls. It shows three runs of a script './fact.sh'. The first run takes input '5' and outputs '120'. The second run takes input '1' and outputs '1'. The third run takes input '0' and outputs '1'.

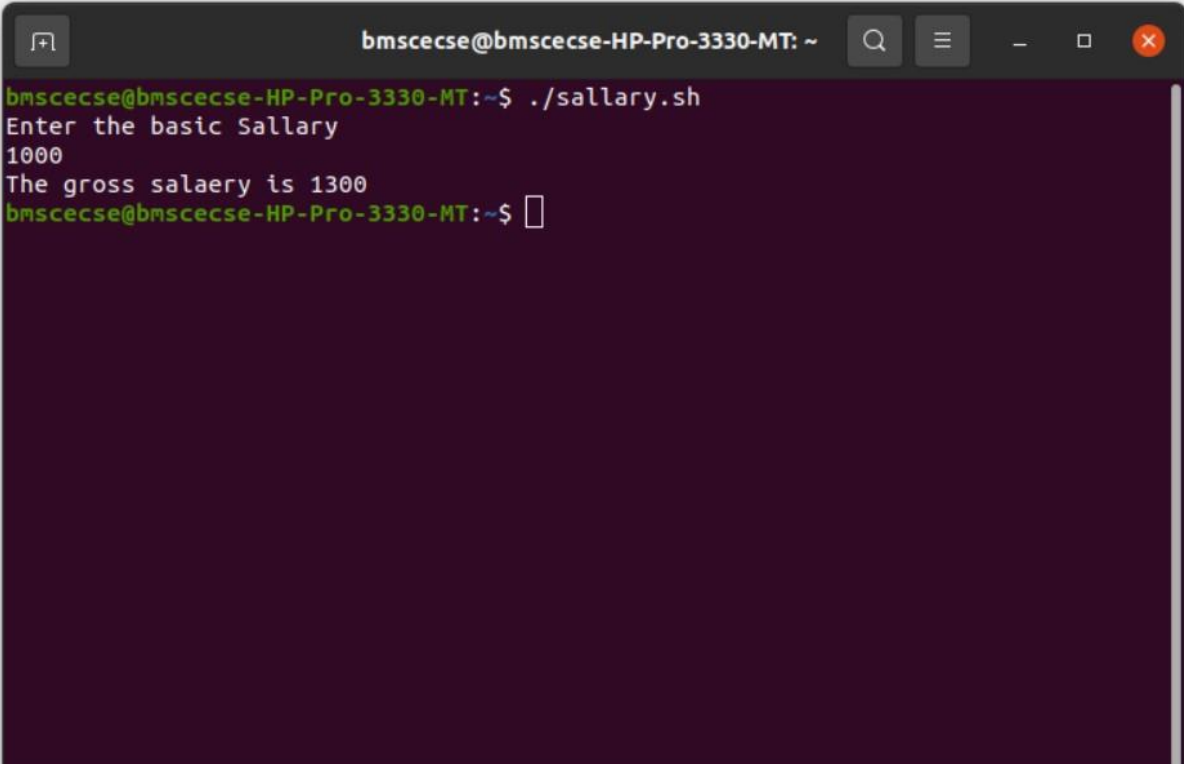
```
bmscecse@bmscecse-HP-Pro-3330-MT:~$ ./fact.sh
Enter the no
5
120
bmscecse@bmscecse-HP-Pro-3330-MT:~$ ./fact.sh
Enter the no
1
1
bmscecse@bmscecse-HP-Pro-3330-MT:~$ ./fact.sh
Enter the no
0
1
bmscecse@bmscecse-HP-Pro-3330-MT:~$
```

PROGRAM6: Shell script to compute the gross salary of an employee

CODE:

```
#!/bin/sh
echo "Enter the basic Sallary"
read basic
da=`expr $basic\*10/100|bc`
hra=`expr $basic\*20/100|bc`
gross_sal=`expr $basic+$da+$hra|bc`
echo "The gross salaery is "$gross_sal
```

OUTPUT:



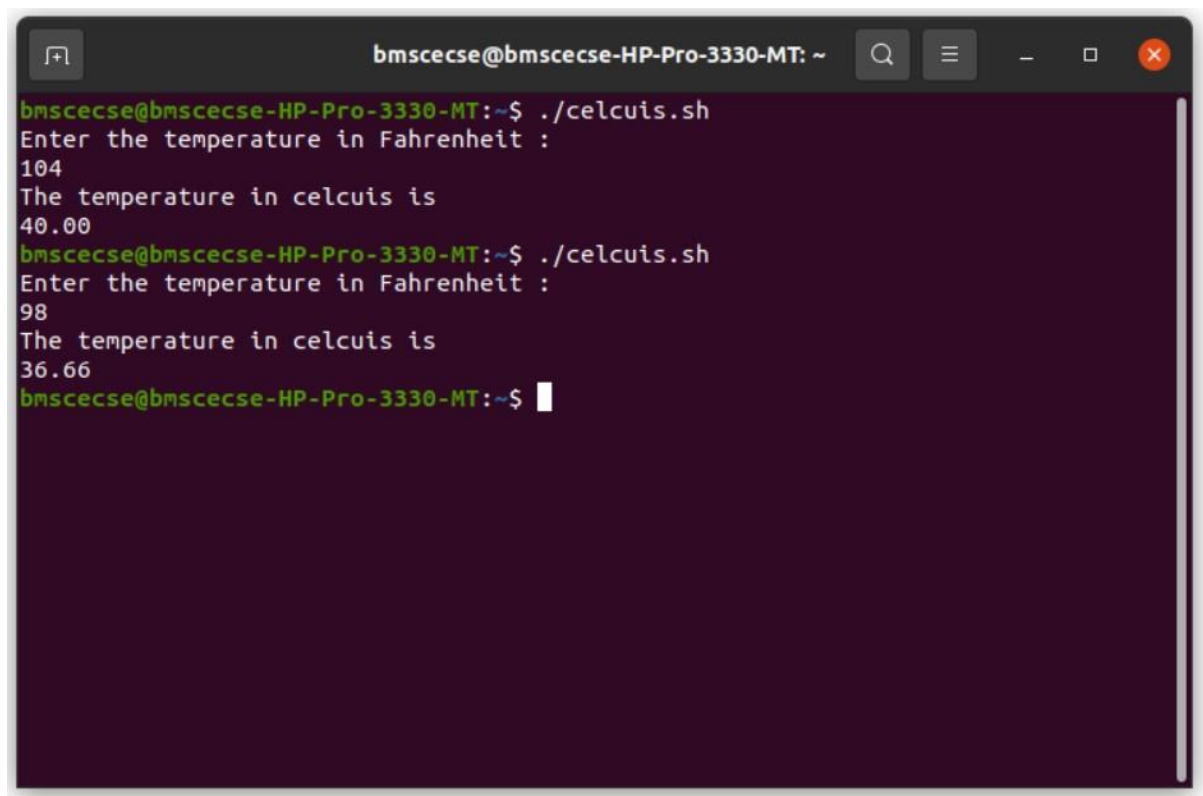
```
bmscecse@bmscecse-HP-Pro-3330-MT: ~$ ./salary.sh
Enter the basic Sallary
1000
The gross salaery is 1300
bmscecse@bmscecse-HP-Pro-3330-MT: ~$
```

PROGRAM7: Shell script to convert the temperature Fahrenheit to Celsius

CODE:

```
#!/bin/sh
echo "Enter the temperature in Fahrenheit : "
read temp
var=32
f=`expr $temp-$var|bc`
s=`expr $f*5|bc`
echo "The temperature in celcuis is "
echo "scale=2; $s/9"|bc
```

OUTPUT:



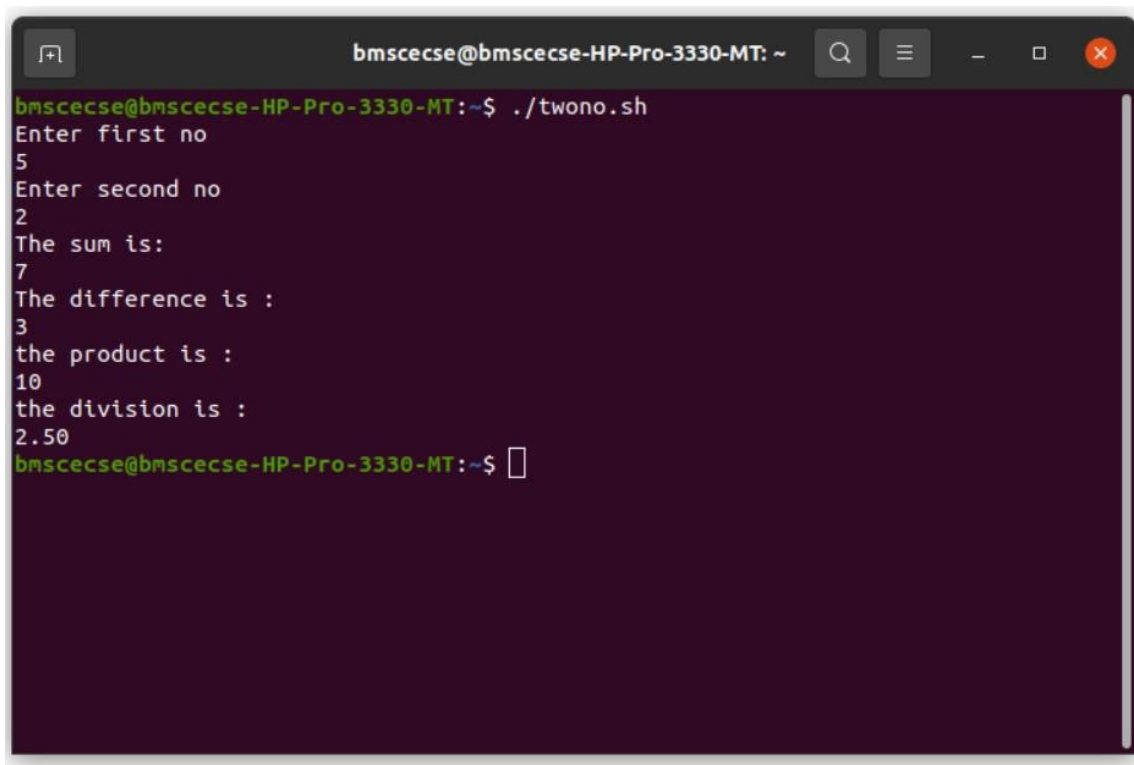
```
bmscecse@bmscecse-HP-Pro-3330-MT: ~  
bmscecse@bmscecse-HP-Pro-3330-MT:~$ ./celcuis.sh  
Enter the temperature in Fahrenheit :  
104  
The temperature in celcuis is  
40.00  
bmscecse@bmscecse-HP-Pro-3330-MT:~$ ./celcuis.sh  
Enter the temperature in Fahrenheit :  
98  
The temperature in celcuis is  
36.66  
bmscecse@bmscecse-HP-Pro-3330-MT:~$
```

PROGRAM8: Shell script to perform arithmetic operations on given two numbers

CODE:

```
#!/bin/sh  
echo "Enter first no"  
read f  
echo "Enter second no"  
read s  
echo "The sum is:"  
echo "$f+$s"|bc  
echo "The difference is :"  
echo "$f-$s"|bc  
echo "the product is :"  
echo "$f*$s"|bc  
echo "the division is :"  
echo "scale=2; $f/$s"|bc
```

OUTPUT:



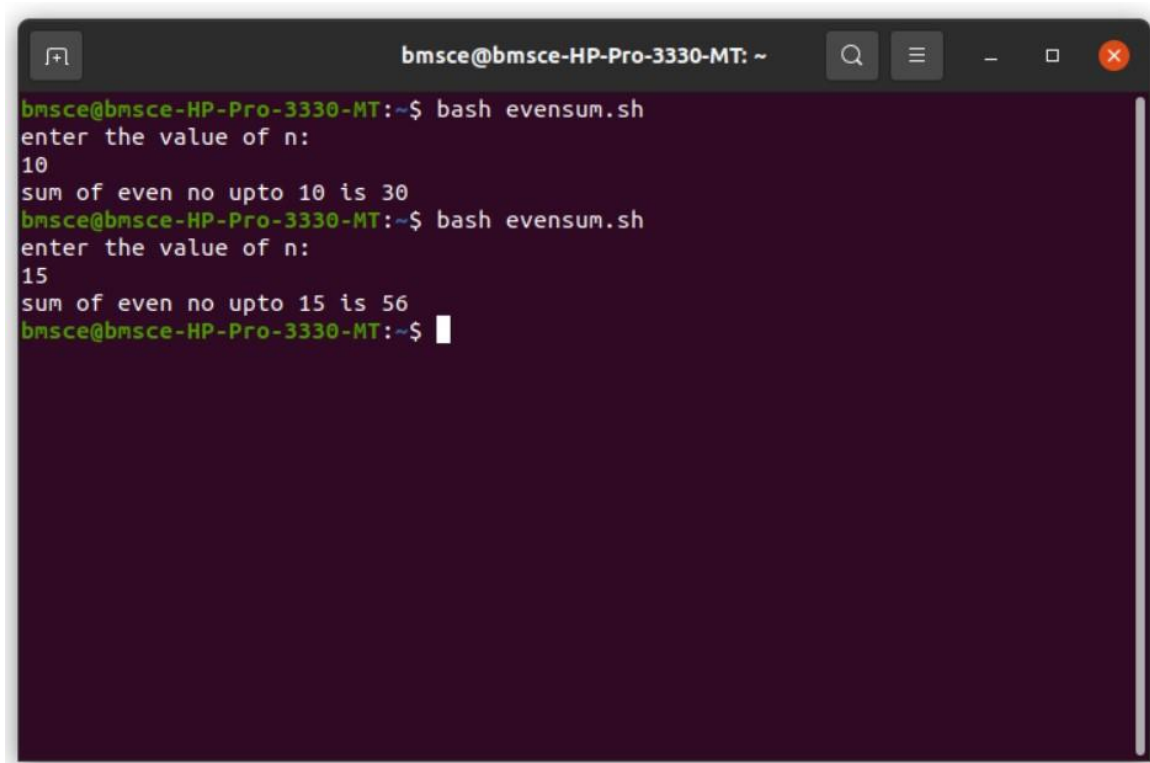
```
bmscecse@bmscecse-HP-Pro-3330-MT: ~  
bmscecse@bmscecse-HP-Pro-3330-MT:~$ ./twono.sh  
Enter first no  
5  
Enter second no  
2  
The sum is:  
7  
The difference is :  
3  
the product is :  
10  
the division is :  
2.50  
bmscecse@bmscecse-HP-Pro-3330-MT:~$
```

PROGRAM9: Shell script to find the sum of even numbers upto n

CODE:

```
#!/bin/bash  
echo "enter the value of n:"  
read n  
sum=0  
for ((c=0; c<=$n; c=c+2))  
do  
    sum=`expr $sum+$c | bc`  
done  
echo "sum of even no upto $n is $sum"
```


OUTPUT:



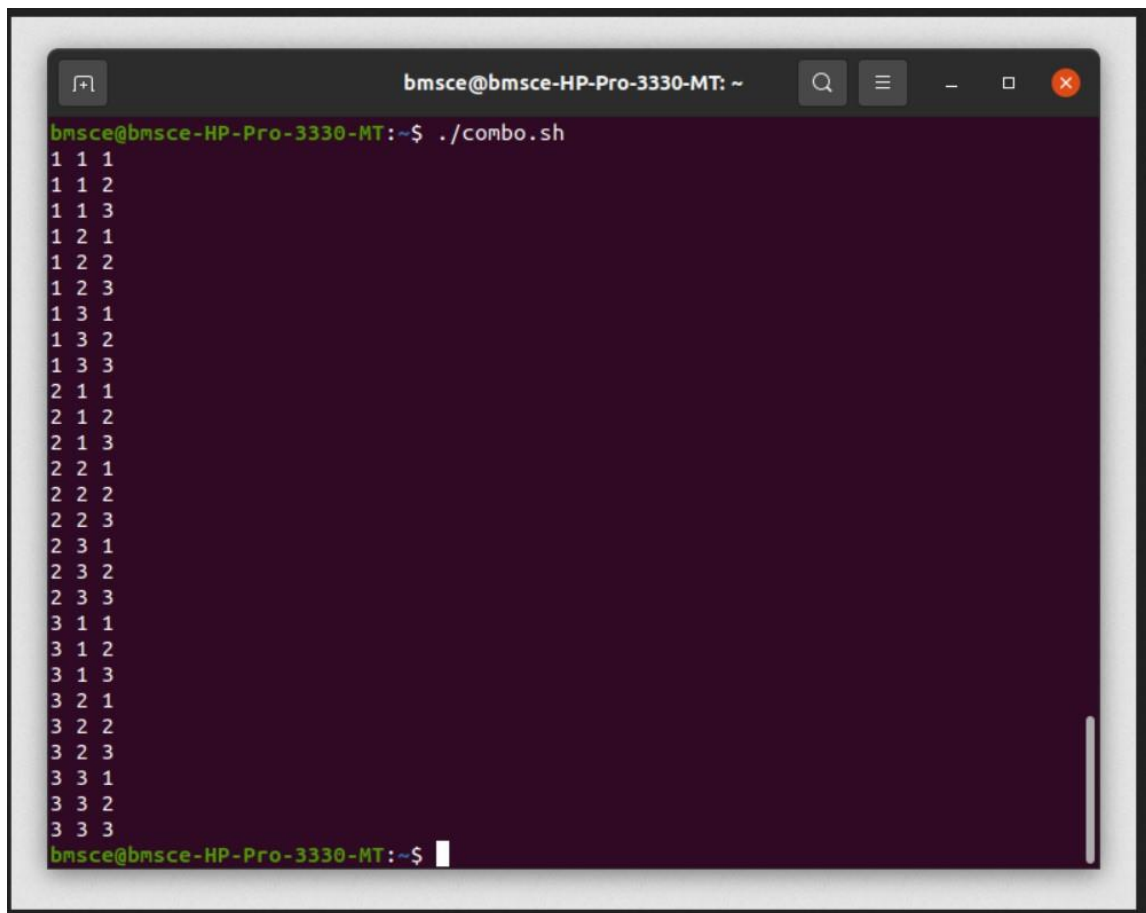
```
bmsce@bmsce-HP-Pro-3330-MT: ~  
bmsce@bmsce-HP-Pro-3330-MT:~$ bash evensum.sh  
enter the value of n:  
10  
sum of even no upto 10 is 30  
bmsce@bmsce-HP-Pro-3330-MT:~$ bash evensum.sh  
enter the value of n:  
15  
sum of even no upto 15 is 56  
bmsce@bmsce-HP-Pro-3330-MT:~$
```

PROGRAM10: Shell script to print the combinations of numbers 123

CODE:

```
#!/bin/sh  
for i in 1 2 3  
do  
    for j in 1 2 3  
    do  
        for k in 1 2 3  
        do  
            echo $i $j $k  
        done  
    done  
done
```

OUTPUT:

A terminal window titled 'bmsce@bmsce-HP-Pro-3330-MT: ~' with search, menu, and window control icons. The prompt is 'bmsce@bmsce-HP-Pro-3330-MT:~\$./combo.sh'. The output is a list of 27 combinations of three numbers (1, 2, or 3) in a 3x3 grid format. The combinations are: (1,1,1), (1,1,2), (1,1,3), (1,2,1), (1,2,2), (1,2,3), (1,3,1), (1,3,2), (1,3,3), (2,1,1), (2,1,2), (2,1,3), (2,2,1), (2,2,2), (2,2,3), (2,3,1), (2,3,2), (2,3,3), (3,1,1), (3,1,2), (3,1,3), (3,2,1), (3,2,2), (3,2,3), (3,3,1), (3,3,2), (3,3,3). The prompt 'bmsce@bmsce-HP-Pro-3330-MT:~\$' is visible at the bottom with a cursor.

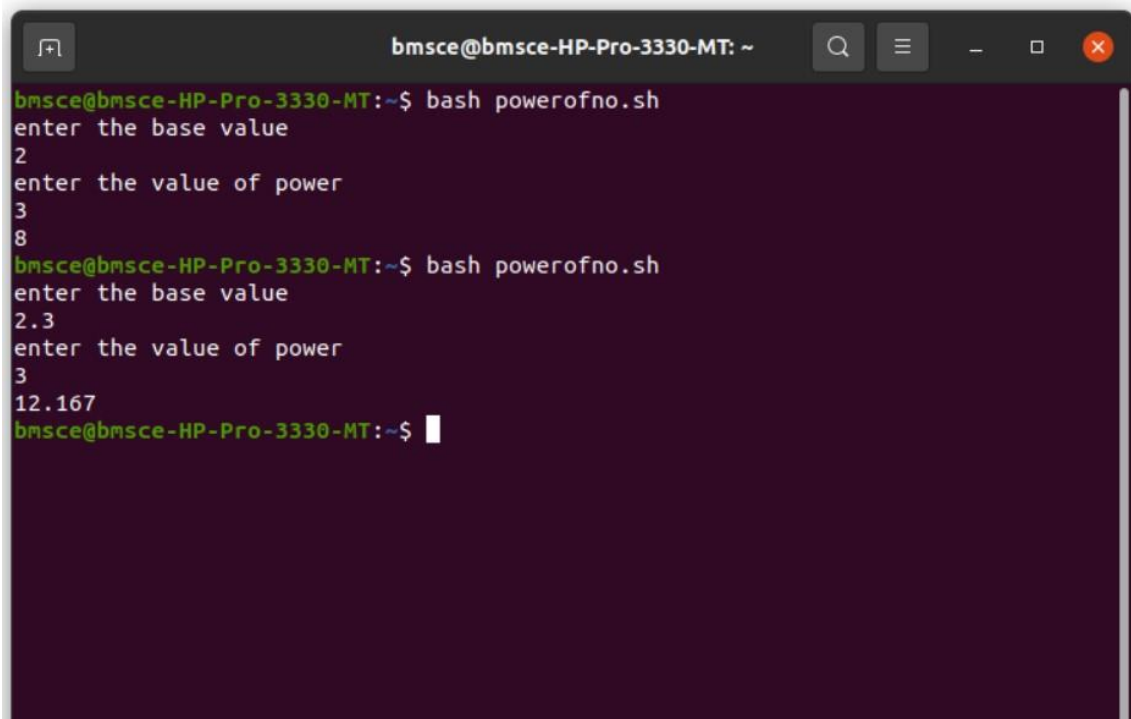
```
bmsce@bmsce-HP-Pro-3330-MT:~$ ./combo.sh
1 1 1
1 1 2
1 1 3
1 2 1
1 2 2
1 2 3
1 3 1
1 3 2
1 3 3
2 1 1
2 1 2
2 1 3
2 2 1
2 2 2
2 2 3
2 3 1
2 3 2
2 3 3
3 1 1
3 1 2
3 1 3
3 2 1
3 2 2
3 2 3
3 3 1
3 3 2
3 3 3
bmsce@bmsce-HP-Pro-3330-MT:~$
```

PROGRAM11: Shell script to find the power of a number

CODE:

```
#!/bin/bash
echo "enter the base value"
read b
echo "enter the value of power"
read p
res=1
for ((c=1; c<=$p; c++))
do
    res=`echo "scale=3; $b*$res"|bc`
done
echo $res
```

OUTPUT:



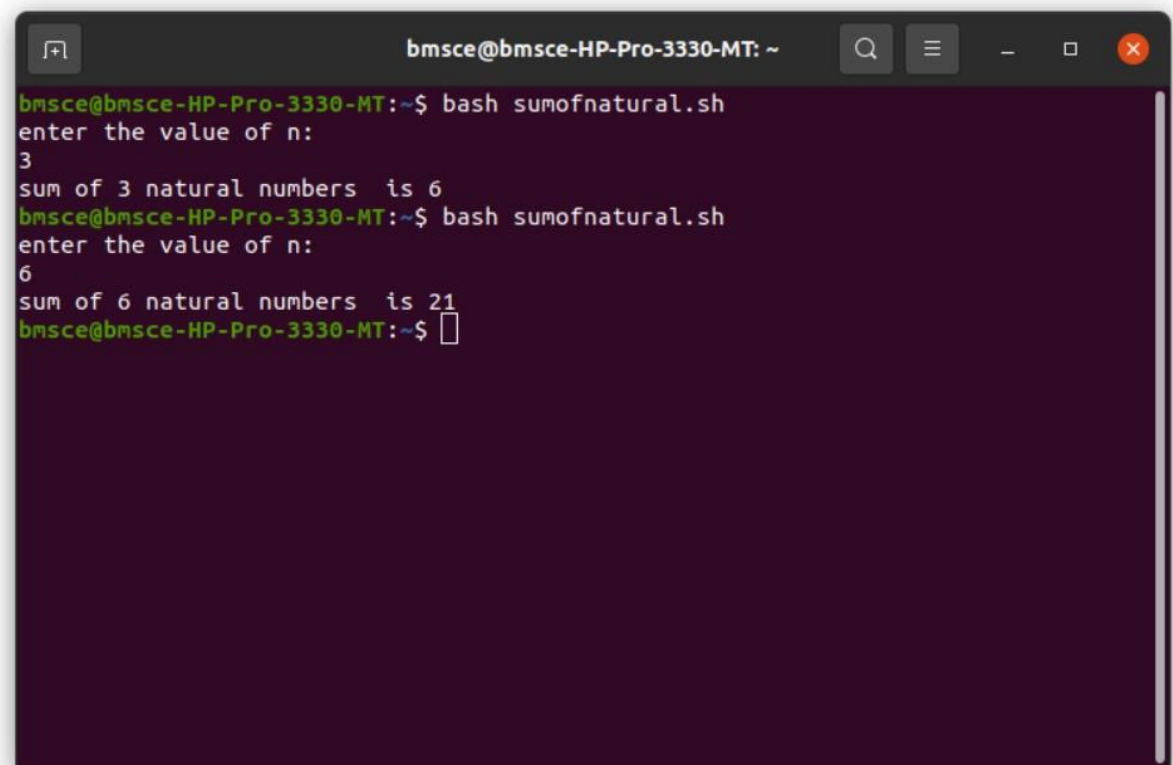
```
bmsce@bmsce-HP-Pro-3330-MT: ~  
bmsce@bmsce-HP-Pro-3330-MT:~$ bash powerofno.sh  
enter the base value  
2  
enter the value of power  
3  
8  
bmsce@bmsce-HP-Pro-3330-MT:~$ bash powerofno.sh  
enter the base value  
2.3  
enter the value of power  
3  
12.167  
bmsce@bmsce-HP-Pro-3330-MT:~$
```

PROGRAM12: Shell script to find the sum of n natural numbers

CODE:

```
#!/bin/bash  
echo "enter the value of n:"  
read n  
sum=0  
for ((c=0; c<=$n; c++))  
do  
    sum=`expr $sum+$c|bc`  
done  
echo "sum of $n natural numbers is $sum"
```

OUTPUT:



```
bmsce@bmsce-HP-Pro-3330-MT: ~  
bmsce@bmsce-HP-Pro-3330-MT:~$ bash sumofnatural.sh  
enter the value of n:  
3  
sum of 3 natural numbers is 6  
bmsce@bmsce-HP-Pro-3330-MT:~$ bash sumofnatural.sh  
enter the value of n:  
6  
sum of 6 natural numbers is 21  
bmsce@bmsce-HP-Pro-3330-MT:~$
```

PROGRAM13: Shell script to display the pass class of a student

CODE:

```
#!/bin/sh  
pass=0  
fail=0  
i=1  
while [ $i -le 6 ]  
do  
    echo "Enter the cie and see marks(out of 50 for see) of the sub$i "  
    read cie see  
    total=`expr $cie+$see|bc`  
    case $total in  
        100) echo "S grade "  
            pass=$((pass+1)) ;;  
        9[0-9]) echo "S grade "  
            pass=$((pass+1)) ;;
```

```

8[0-9]) echo "A grade "
        pass=$((pass+1)) ;;
7[0-9]) echo "B grade "
        pass=$((pass+1)) ;;
6[0-9]) echo "C grade "
        pass=$((pass+1)) ;;
5[0-9]) echo "D grade "
        pass=$((pass+1)) ;;
4[0-9]) echo "E grade "
        pass=$((pass+1)) ;;
[0123][0-9]) echo "F grade "
        fail=$((fail+1)) ;;
*)echo "error in input"

esac

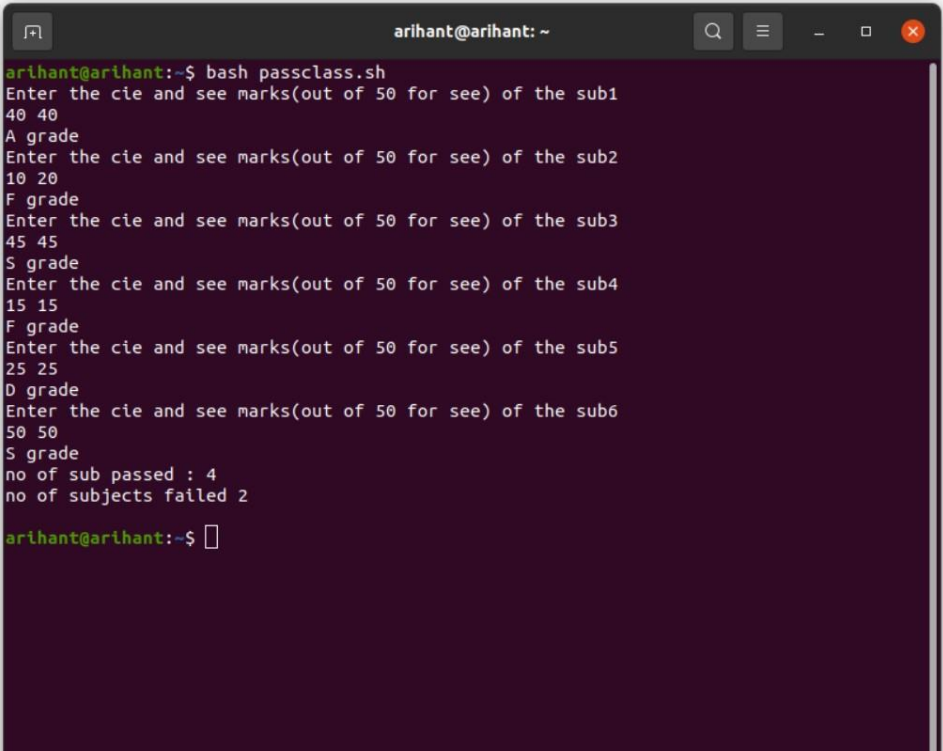
i=$((i+1))

done

echo -e "no of sub passed : $pass\nno of subjects failed $fail\n"

```

OUTPUT:



```

arihant@arihant:~$ bash passclass.sh
Enter the cie and see marks(out of 50 for see) of the sub1
40 40
A grade
Enter the cie and see marks(out of 50 for see) of the sub2
10 20
F grade
Enter the cie and see marks(out of 50 for see) of the sub3
45 45
S grade
Enter the cie and see marks(out of 50 for see) of the sub4
15 15
F grade
Enter the cie and see marks(out of 50 for see) of the sub5
25 25
D grade
Enter the cie and see marks(out of 50 for see) of the sub6
50 50
S grade
no of sub passed : 4
no of subjects failed 2
arihant@arihant:~$

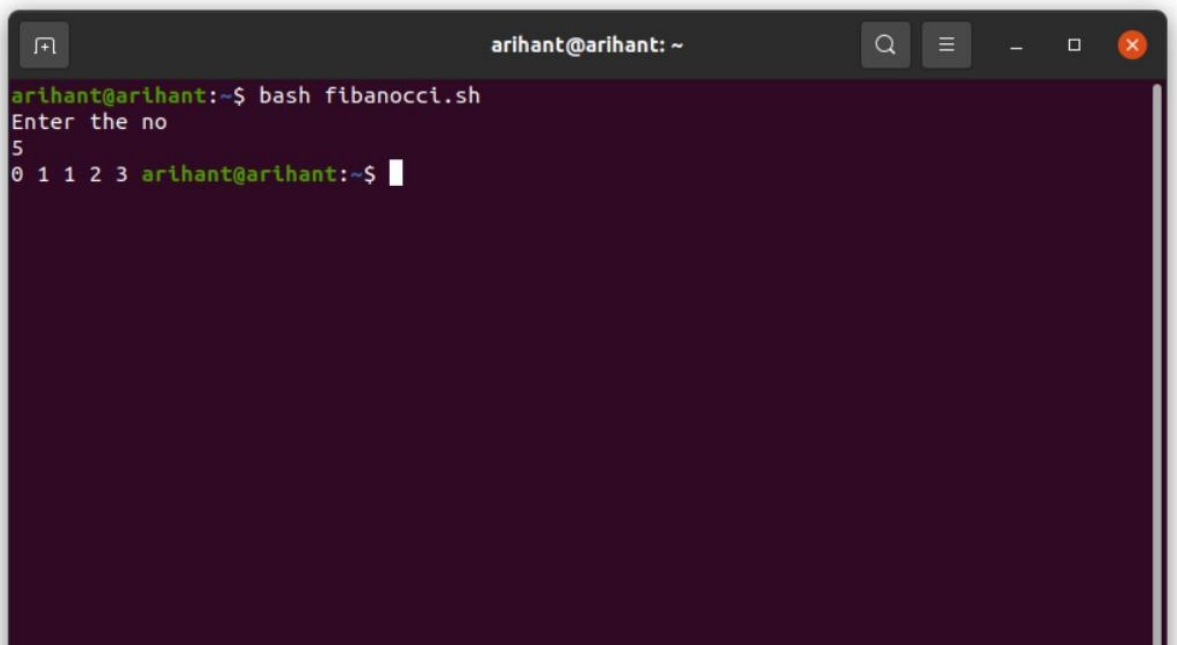
```

PROGRAM14: Shell script to find the Fibonacci series up to n

CODE:

```
#!/bin/sh
echo "Enter the no"
read no
m=0
n=1
while [ $no -gt 0 ]
do
    echo -e "$m \c"
    temp=$m
    m=$((m+$n))
    n=$temp
    no=$((no-1))
done
```

OUTPUT:

A terminal window with a dark background and light green text. The window title is 'arihant@arihant: ~'. The prompt is 'arihant@arihant:~\$'. The user has entered 'bash fibanocci.sh'. The script outputs 'Enter the no' and the user has entered '5'. The script then outputs the Fibonacci sequence '0 1 1 2 3' followed by a space and the prompt 'arihant@arihant:~\$'.

```
arihant@arihant:~$ bash fibanocci.sh
Enter the no
5
0 1 1 2 3 arihant@arihant:~$
```

PROGRAM15: Shell script to count the number of vowels of a string

CODE:

```
#!/bin/sh

echo "Enter the string "

read str

count=0

len=`expr length $str`

while [ $len -gt 0 ]

do

    ch=`expr $str | cut -c $len`

    case $ch in

        [aeiouAEIOU]) count=$((count+1)) ;;

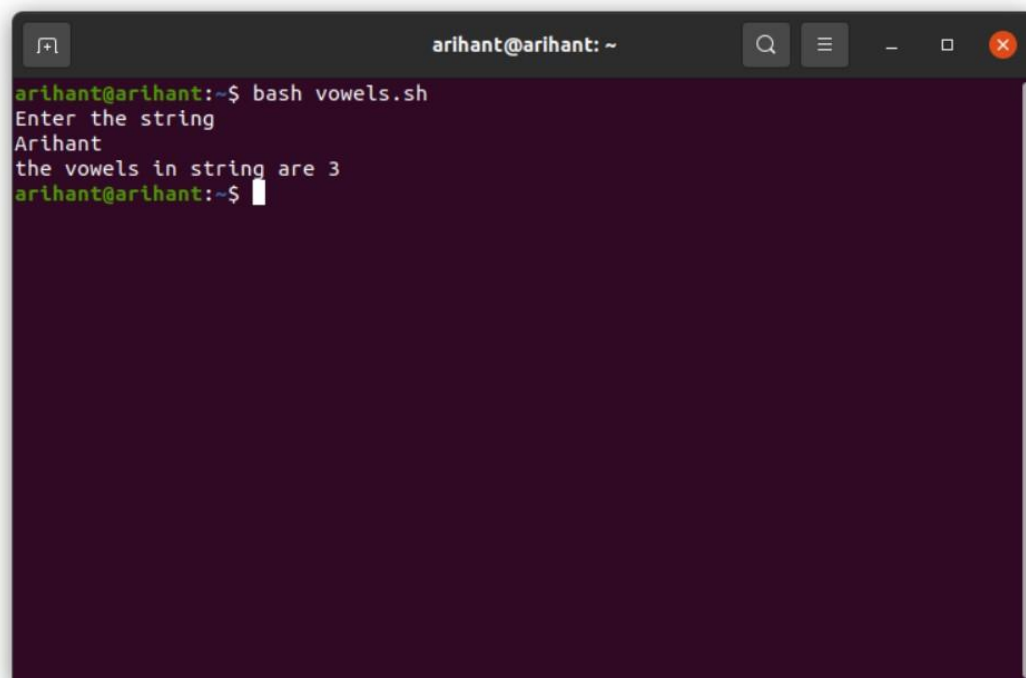
    esac

    len=$((len-1))

done

echo "the vowels in string are $count "
```

OUTPUT:

A terminal window with a dark purple background and a title bar that reads 'arihant@arihant: ~'. The terminal shows the following sequence of commands and output: the user runs 'bash vowels.sh', the script prompts 'Enter the string', the user enters 'Arihant', the script outputs 'the vowels in string are 3', and the prompt returns to 'arihant@arihant:~\$'.

```
arihant@arihant:~$ bash vowels.sh
Enter the string
Arihant
the vowels in string are 3
arihant@arihant:~$
```

PROGRAM16: Shell script to check number of lines, words, characters in a file

CODE:

```
#!/bin/sh

echo "Enter the filename "

read fname

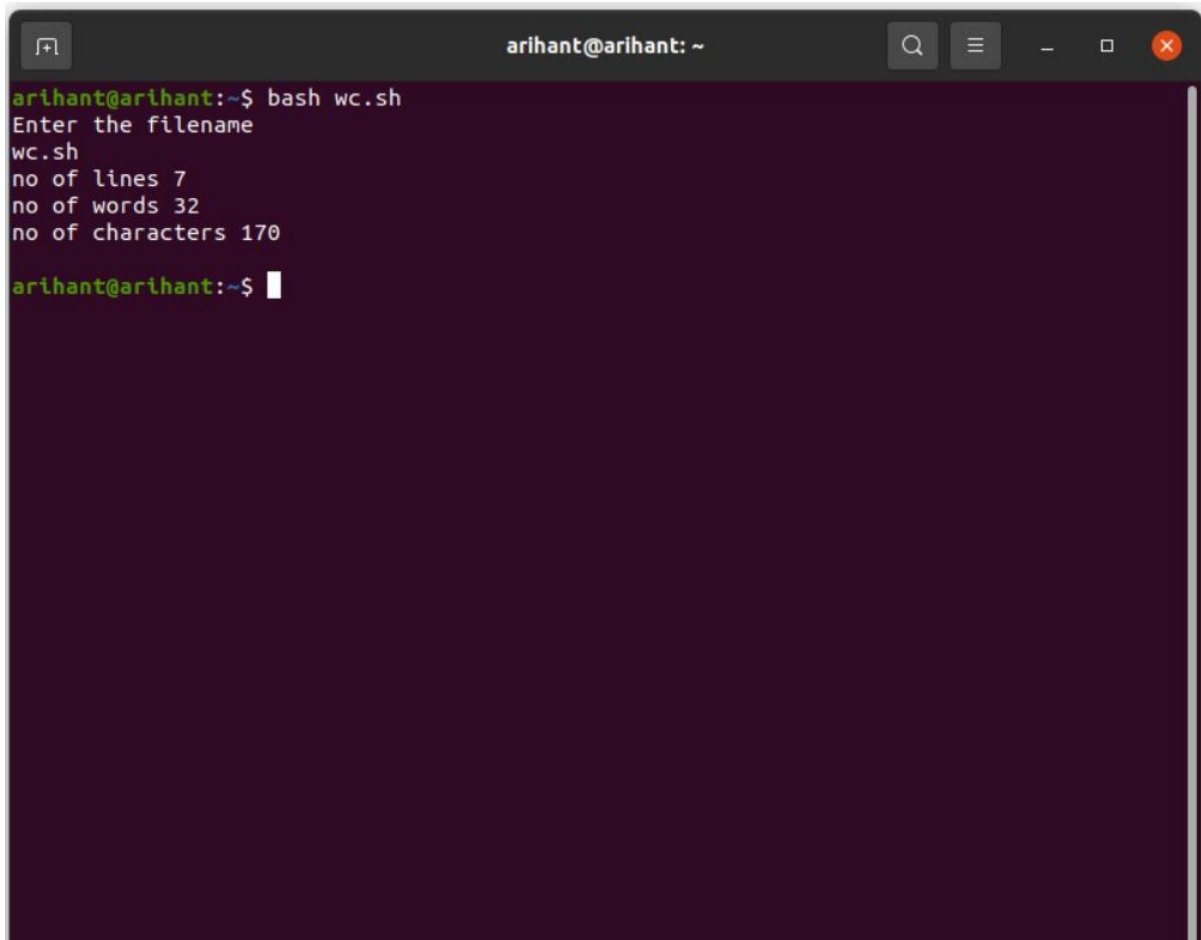
l=`wc -l < $fname`

w=`wc -w < $fname`

c=`wc -m < $fname`

echo -e "no of lines $l\nno of words $w\nno of characters $c\n"
```

OUTPUT:

A terminal window with a dark purple background and light green text. The window title is 'arihant@arihant: ~'. The terminal shows the following sequence of commands and output:

```
arihant@arihant:~$ bash wc.sh
Enter the filename
wc.sh
no of lines 7
no of words 32
no of characters 170
arihant@arihant:~$
```


PROGRAM17: Write a C/C++ program to that outputs the contents of its Environment list

CODE:

```
#include <stdio.h>

int main(int argc, char* argv[ ])
{
    int i;

    char **ptr;

    extern char **environ;

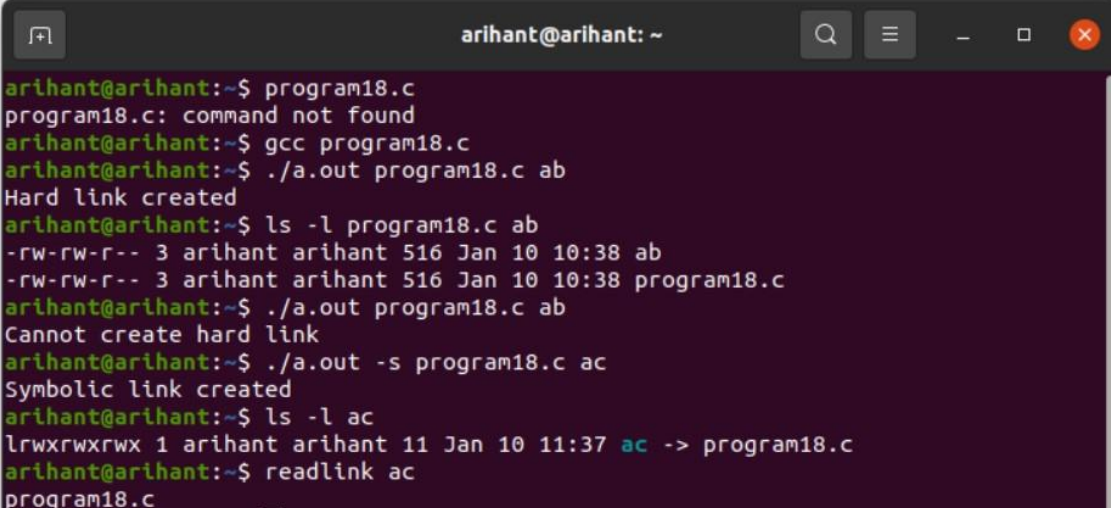
    for( ptr = environ; *ptr != 0; ptr++ ) /*echo all env strings*/
        printf("%s\n", *ptr);

    return 0;
}
```

OUTPUT:

```
int main(int argc, char * argv[])
{
if(argc < 3 || argc > 4 || (argc == 4 && strcmp(argv[1], "-s")))
{
printf("Usage: ./a.out [-s] <org_file> <new_link>\n");
return 1;
}
if(argc == 4)
{
if((symlink(argv[2], argv[3])) == -1)
printf("Cannot create symbolic link\n");
else
printf("Symbolic link created\n");
}
else
{
if((link(argv[1], argv[2])) == -1)
printf("Cannot create hard link\n");
else
printf("Hard link created\n");
}
return 0;
}
```

OUTPUT:



```
arihant@arihant:~$ program18.c
program18.c: command not found
arihant@arihant:~$ gcc program18.c
arihant@arihant:~$ ./a.out program18.c ab
Hard link created
arihant@arihant:~$ ls -l program18.c ab
-rw-rw-r-- 3 arihant arihant 516 Jan 10 10:38 ab
-rw-rw-r-- 3 arihant arihant 516 Jan 10 10:38 program18.c
arihant@arihant:~$ ./a.out program18.c ab
Cannot create hard link
arihant@arihant:~$ ./a.out -s program18.c ac
Symbolic link created
arihant@arihant:~$ ls -l ac
lrwxrwxrwx 1 arihant arihant 11 Jan 10 11:37 ac -> program18.c
arihant@arihant:~$ readlink ac
program18.c
```

PROGRAM19: Write a C/C++ POSIX compliant program that prints the POSIX defined configuration options supported on any given system using feature test macros.

CODE:

```
#define _POSIX_SOURCE

#define _POSIX_C_SOURCE 199309L

#include<stdio.h>

#include<unistd.h>

int main()
{
#ifdef _POSIX_JOB_CONTROL

printf("System supports job control\n");

#else

printf("System does not support job control \n");

#endif

#ifdef _POSIX_SAVED_IDS

printf("System supports saved set-UID and saved set-GID\n");

#else

printf("System does not support saved set-UID and saved set-GID \n");

#endif

#ifdef _POSIX_CHOWN_RESTRICTED
```

```

printf("chown_restricted option is %d\n", _POSIX_CHOWN_RESTRICTED);
#else
printf("System does not support chown_restricted option \n");
#endif

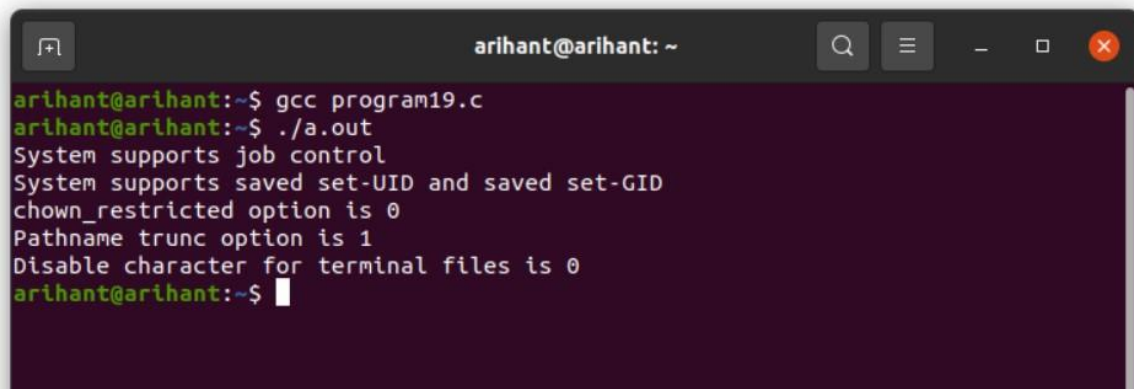
#ifdef _POSIX_NO_TRUNC
printf("Pathname trunc option is %d\n", _POSIX_NO_TRUNC);
#else
printf("System does not support system-wide pathname trunc option \n");
#endif

#ifdef _POSIX_VDISABLE
printf("Disable character for terminal files is %d\n", _POSIX_VDISABLE);
#else
printf("System does not support _POSIX_VDISABLE \n");
#endif

return 0;
}

```

OUTPUT:



```

arihant@arihant:~$ gcc program19.c
arihant@arihant:~$ ./a.out
System supports job control
System supports saved set-UID and saved set-GID
chown_restricted option is 0
Pathname trunc option is 1
Disable character for terminal files is 0
arihant@arihant:~$

```

PROGRAM20: Write a C/C++ program which demonstrates interprocess communication between a reader process and a writer process. Use mkfifo, open, read, write and close APIs in your program.

CODE:

```

#include<sys/types.h>

#include<unistd.h>

```

```

#include<fcntl.h>

#include<sys/stat.h>

#include<string.h>

#include<errno.h>

#include<stdio.h>

int main(int argc, char* argv[])

{

int fd;

char buf[256];

if(argc != 2 && argc != 3)

{

printf("USAGE %s <file> [<arg>]\n",argv[0]);

return 0;

}

mkfifo(argv[1],S_IFIFO | S_IRWXU | S_IRWXG | S_IRWXO );

if(argc == 2) //reader process

{

fd = open(argv[1], O_RDONLY|O_NONBLOCK);

while(read(fd, buf, sizeof(buf)) > 0)

printf("%s",buf);

}

else

{

fd = open(argv[1], O_WRONLY);

write(fd,argv[2],strlen(argv[2]));

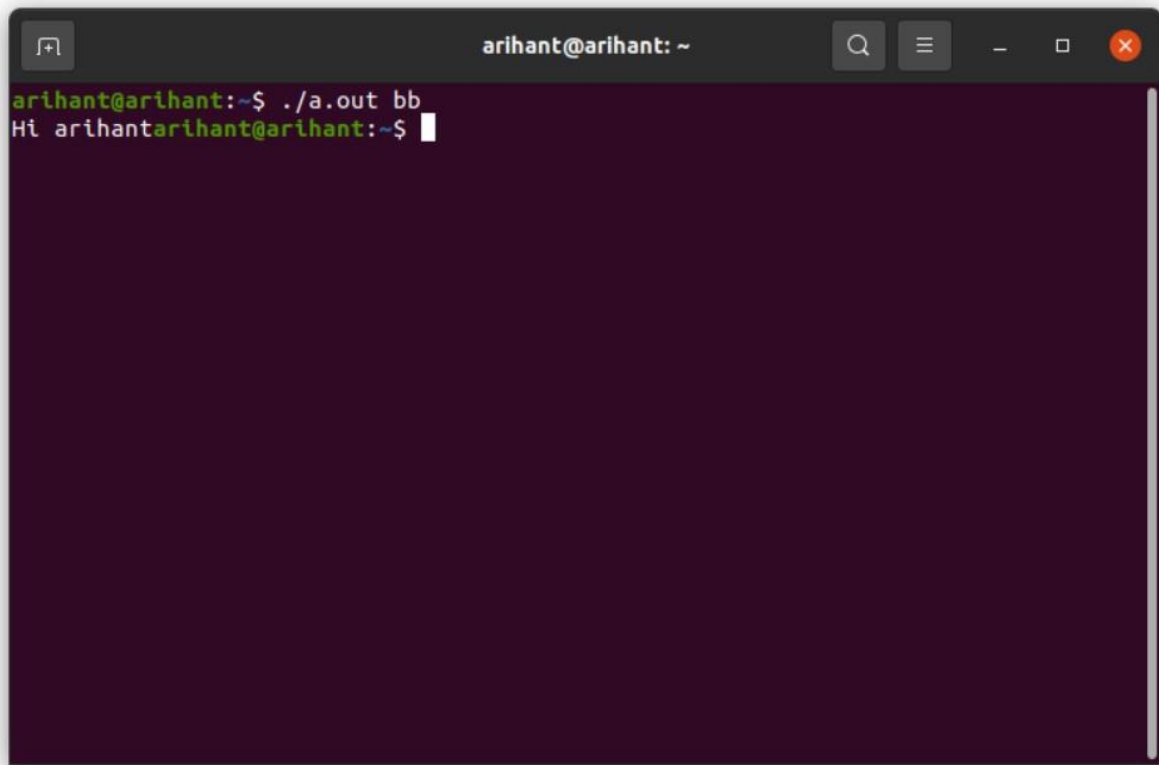
}

close(fd);

}

```

OUTPUT:

A terminal window with a dark background and a light gray title bar. The title bar contains the text 'arihant@arihant: ~' and standard window control icons (search, menu, close). The terminal shows a command prompt 'arihant@arihant:~\$' followed by the command './a.out bb'. The output 'Hi arihant' is displayed on the next line, followed by another prompt 'arihant@arihant:~\$' with a cursor. A vertical scrollbar is visible on the right side of the terminal area.

```
arihant@arihant:~$ ./a.out bb
Hi arihant
arihant@arihant:~$
```