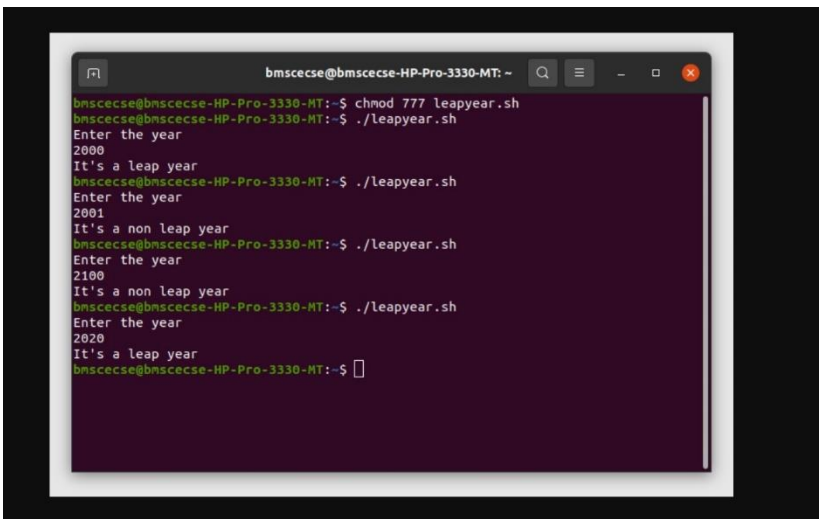
LAB PROGRAMS:

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:

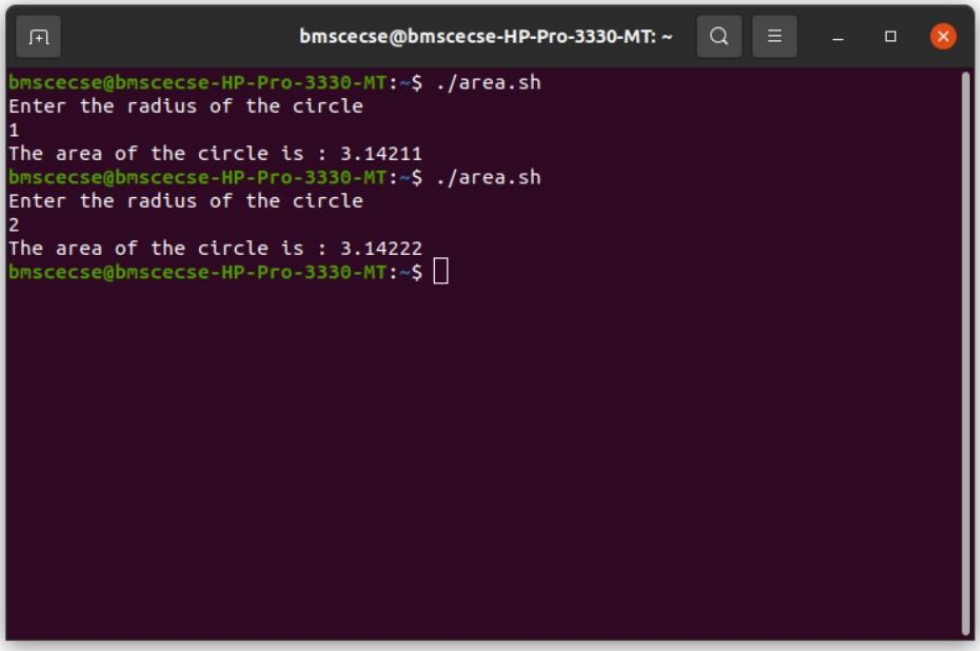


PROGRAM2: Shell script to find the area of a circle

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:



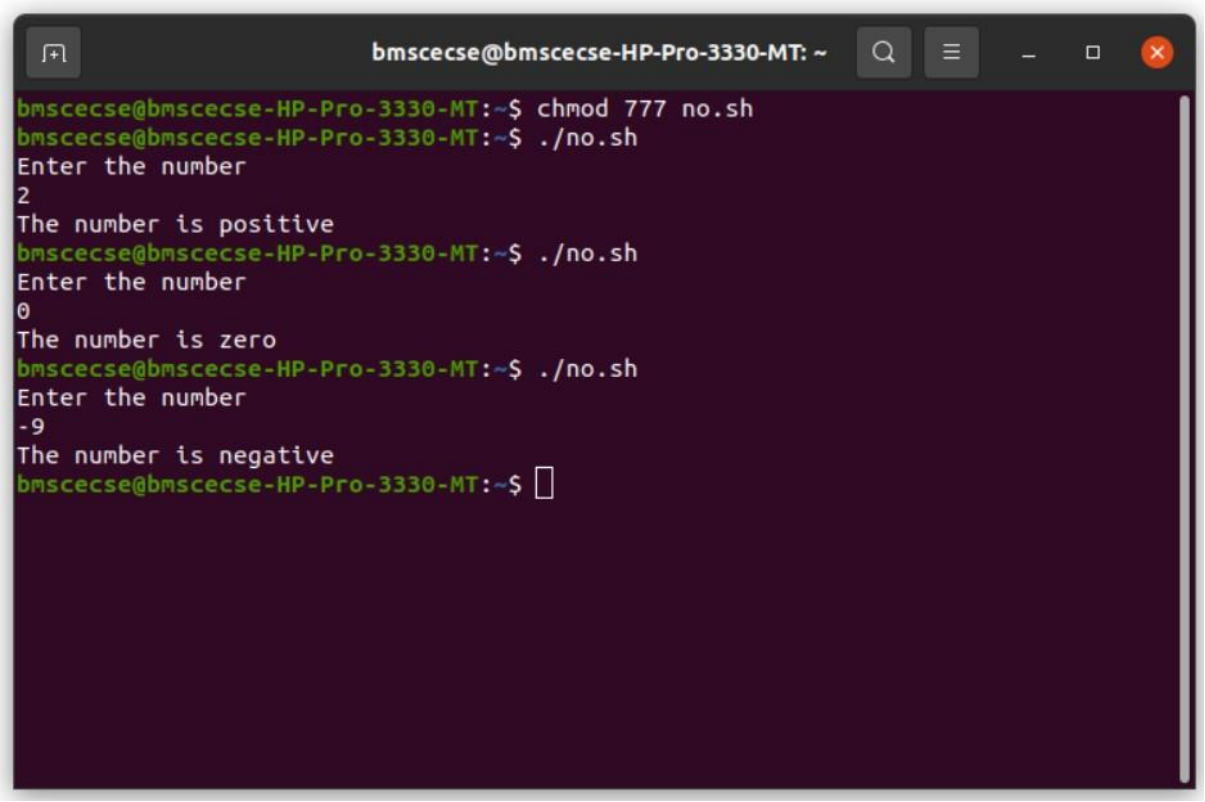
PROGRAM3: Shell script to check whether the number is zero/ positive/ negative

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:



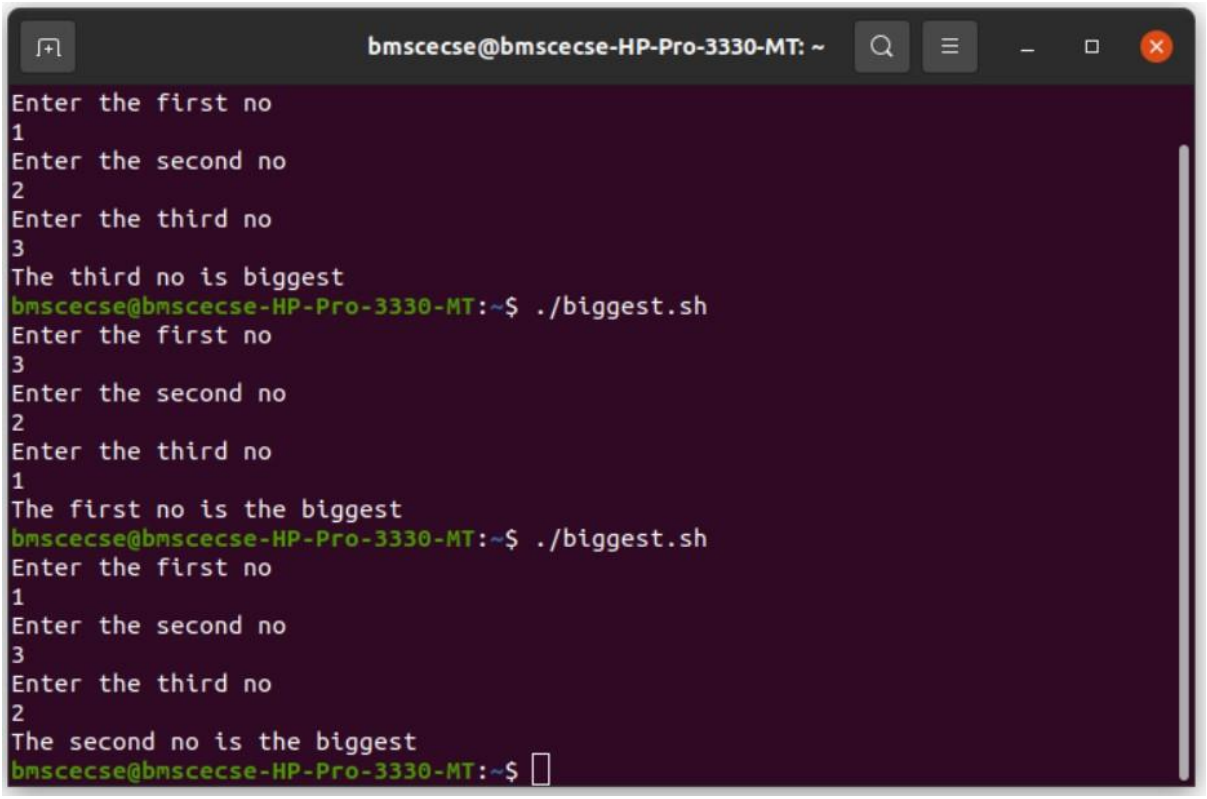
PROGRAM4: Shell script to find the biggest of three numbers

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:



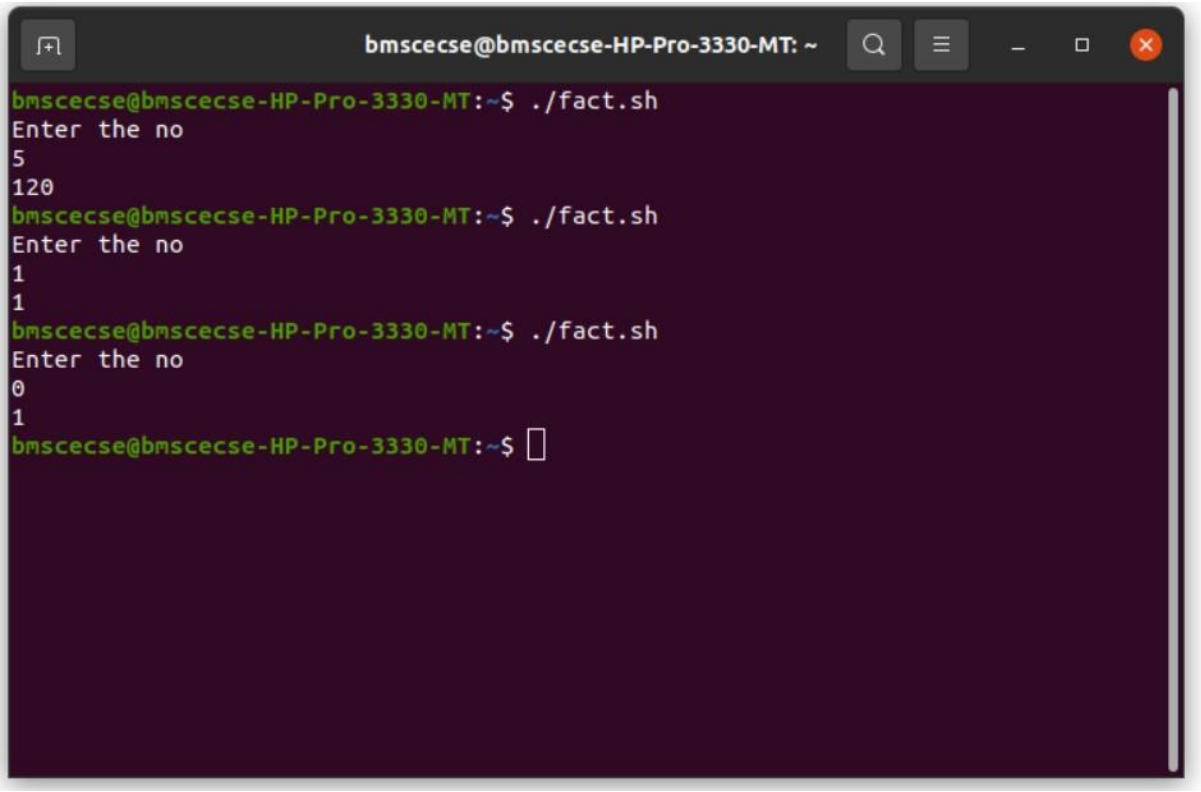
PROGRAM5: Shell script to find the factorial of a number

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:

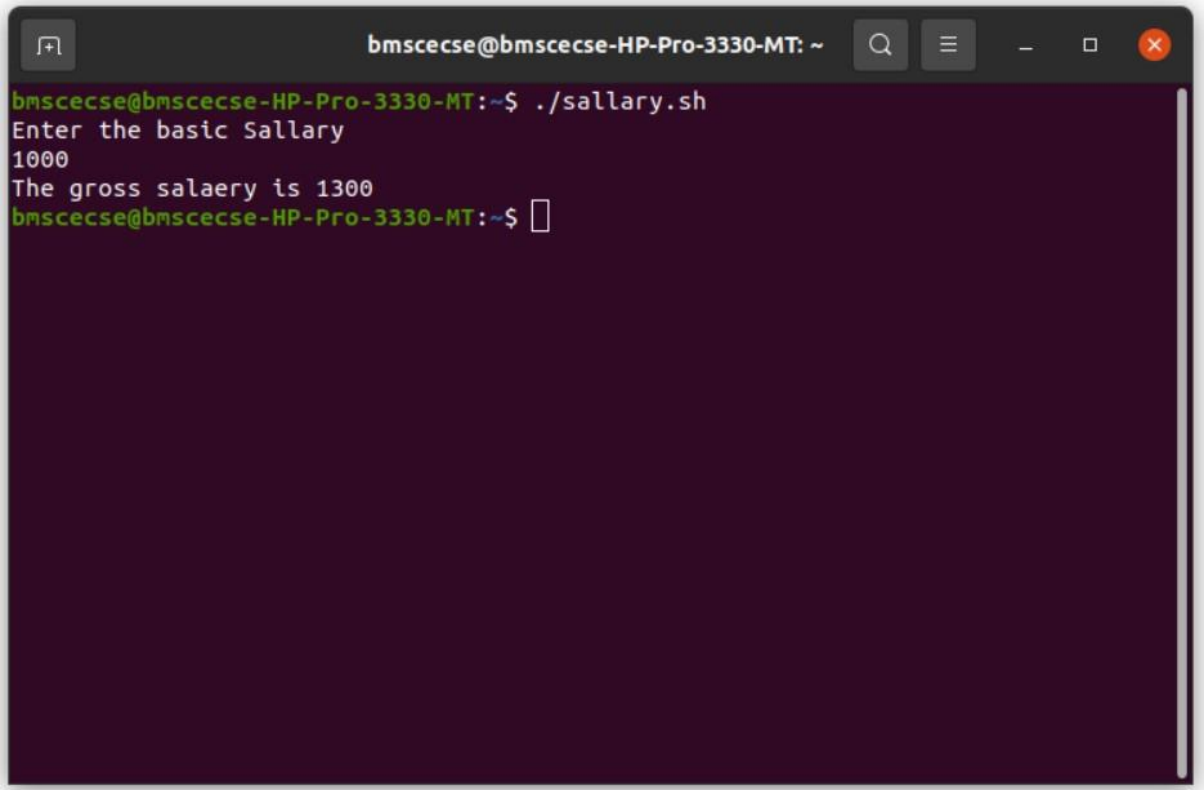


PROGRAM6: Shell script to compute the gross salary of an employee

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:

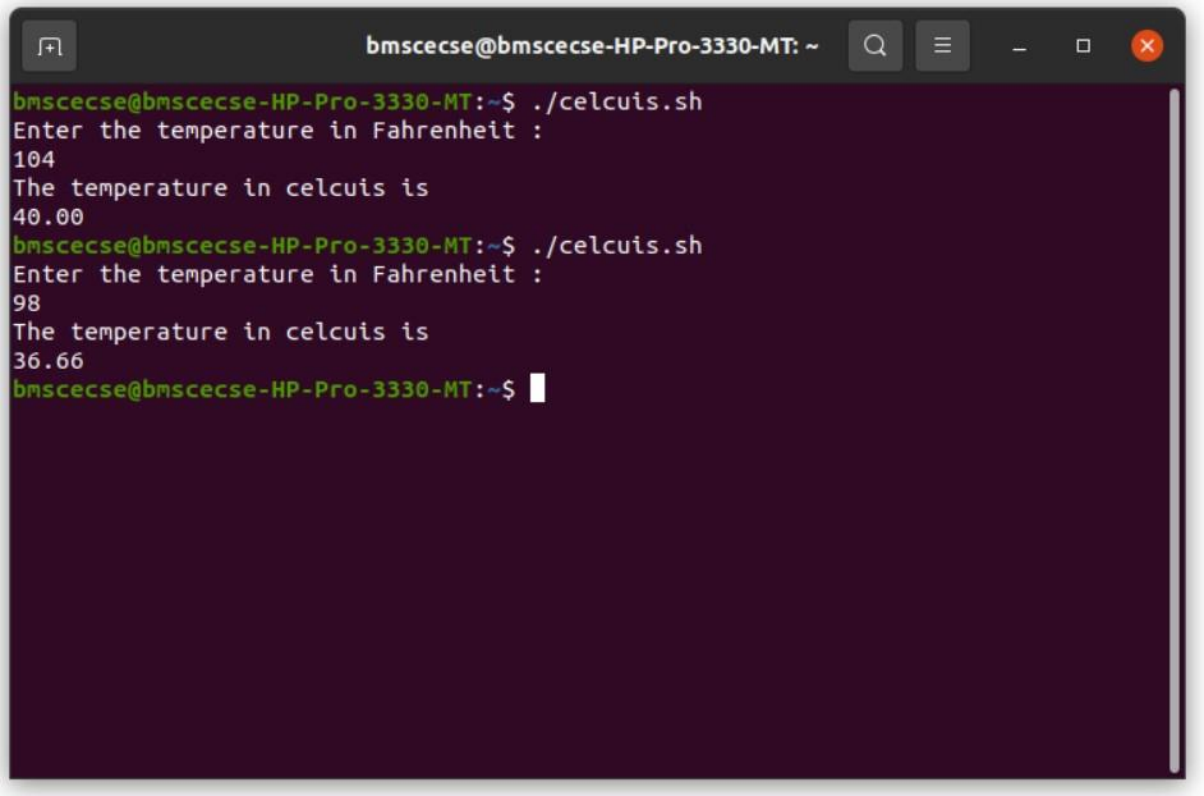


PROGRAM7: Shell script to convert the temperature Fahrenheit to Celsius

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:

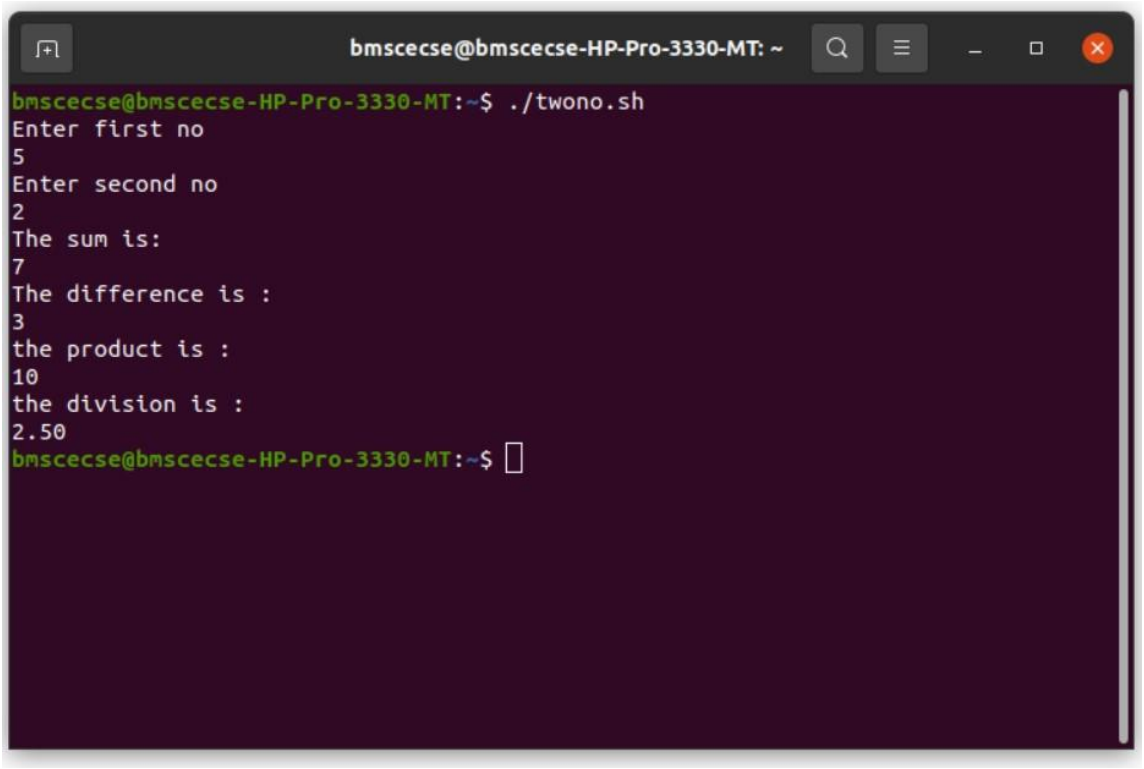


PROGRAM8: Shell script to perform arithmetic operations on given two numbers

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:

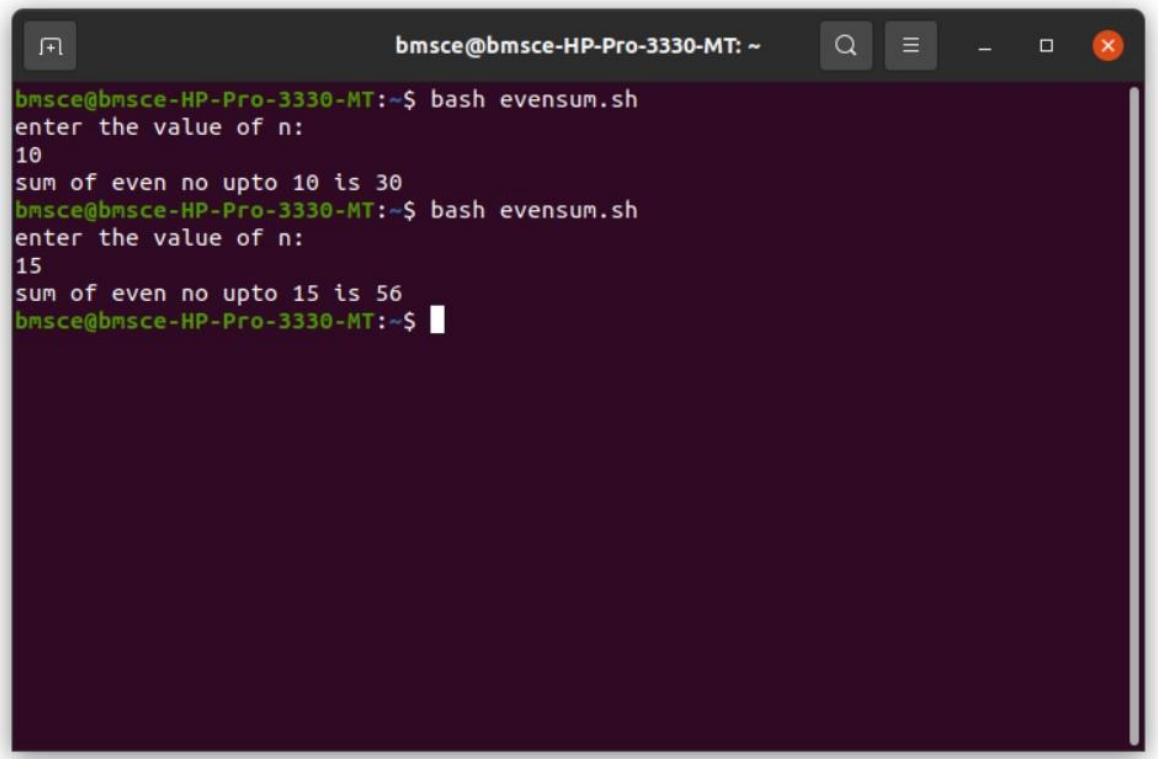


PROGRAM9: Shell script to find the sum of even numbers upto n

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:

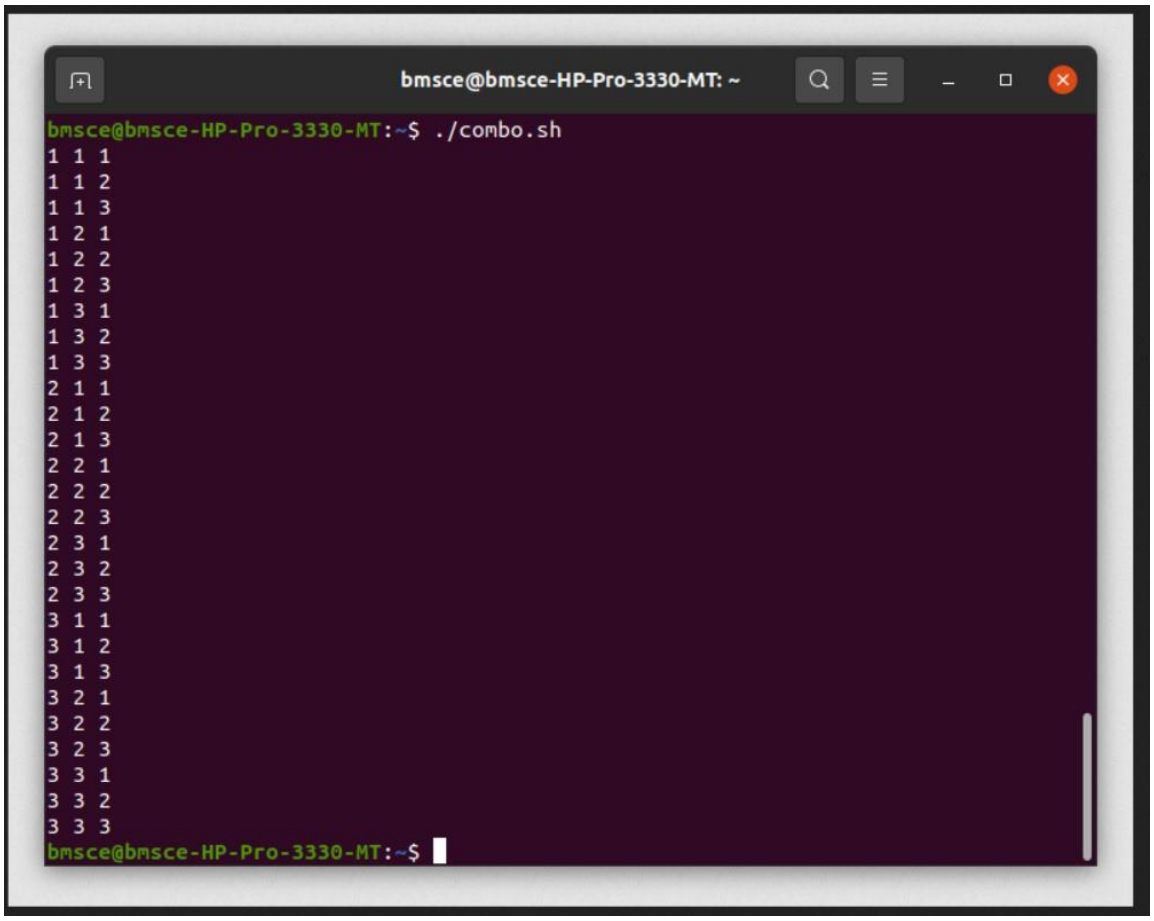


PROGRAM10: Shell script to print the combinations of numbers 123

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:

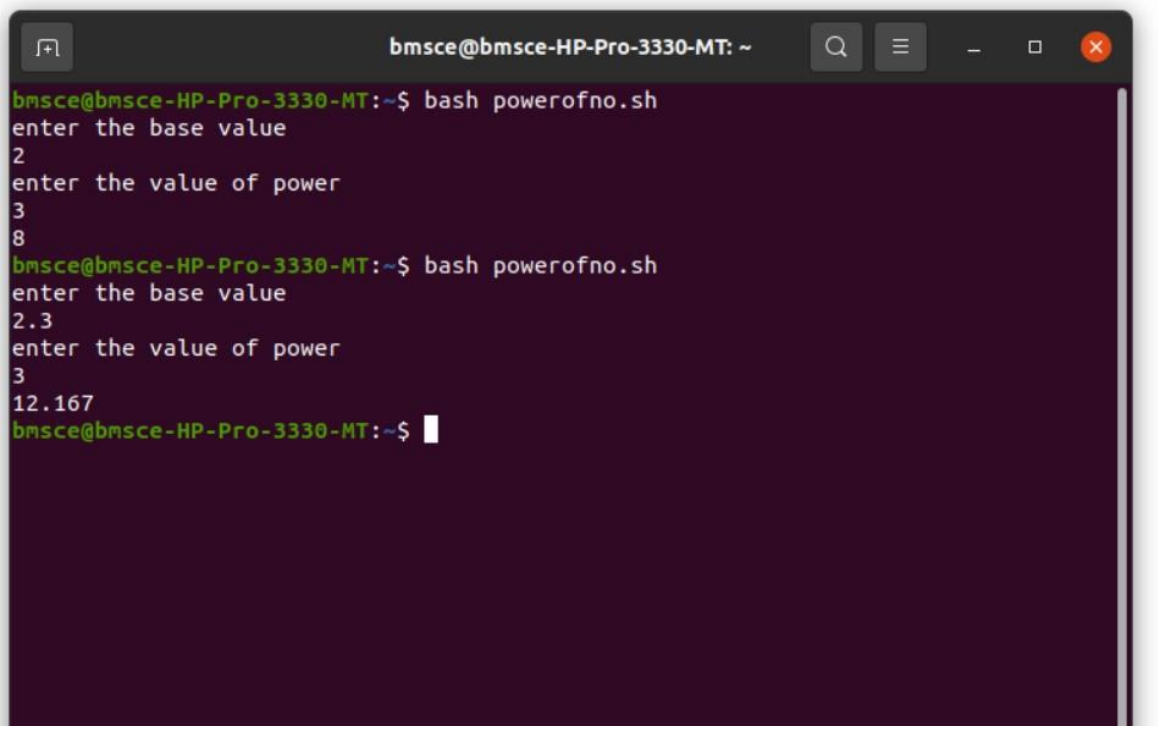


PROGRAM11: Shell script to find the power of a number

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:

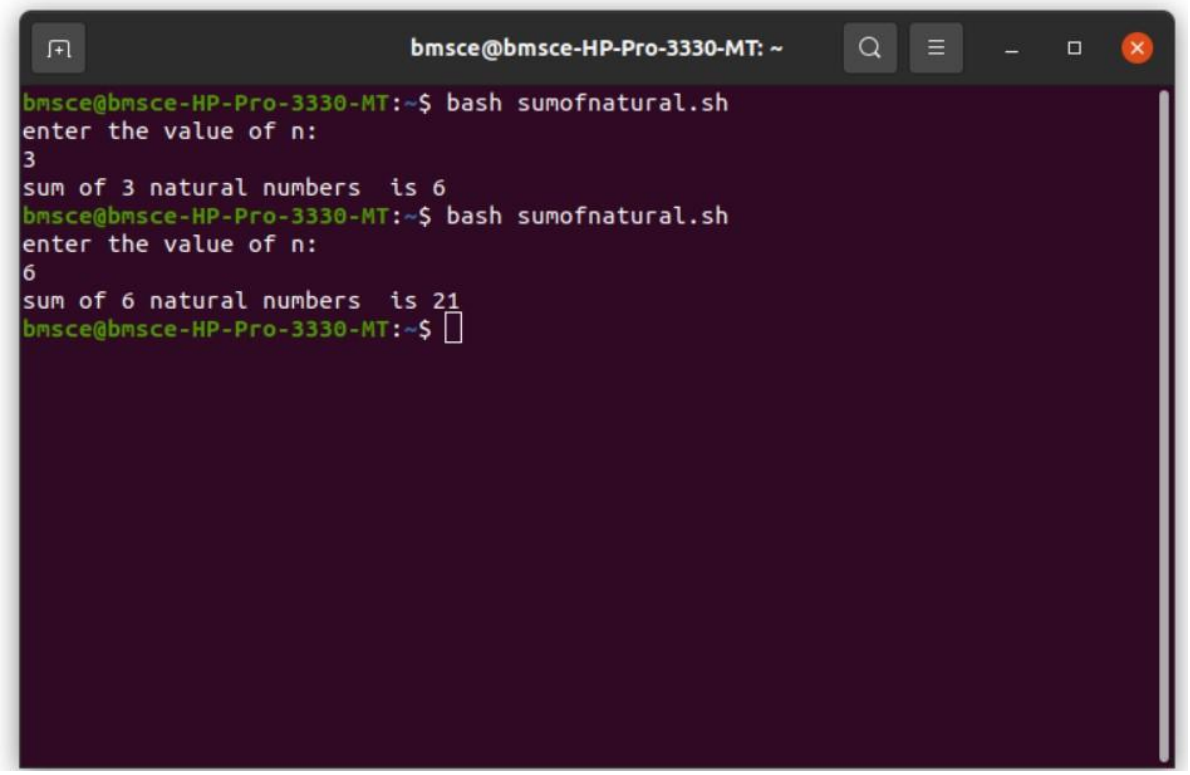


PROGRAM12: Shell script to find the sum of n natural numbers

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:



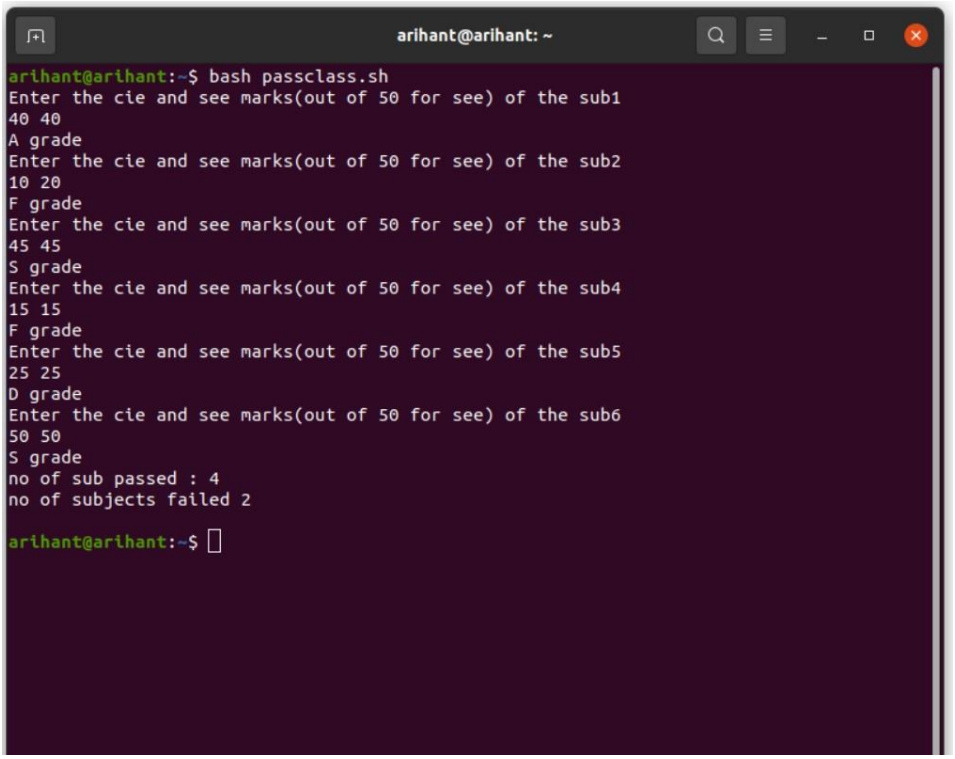
PROGRAM13: Shell script to display the pass class of a student

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:

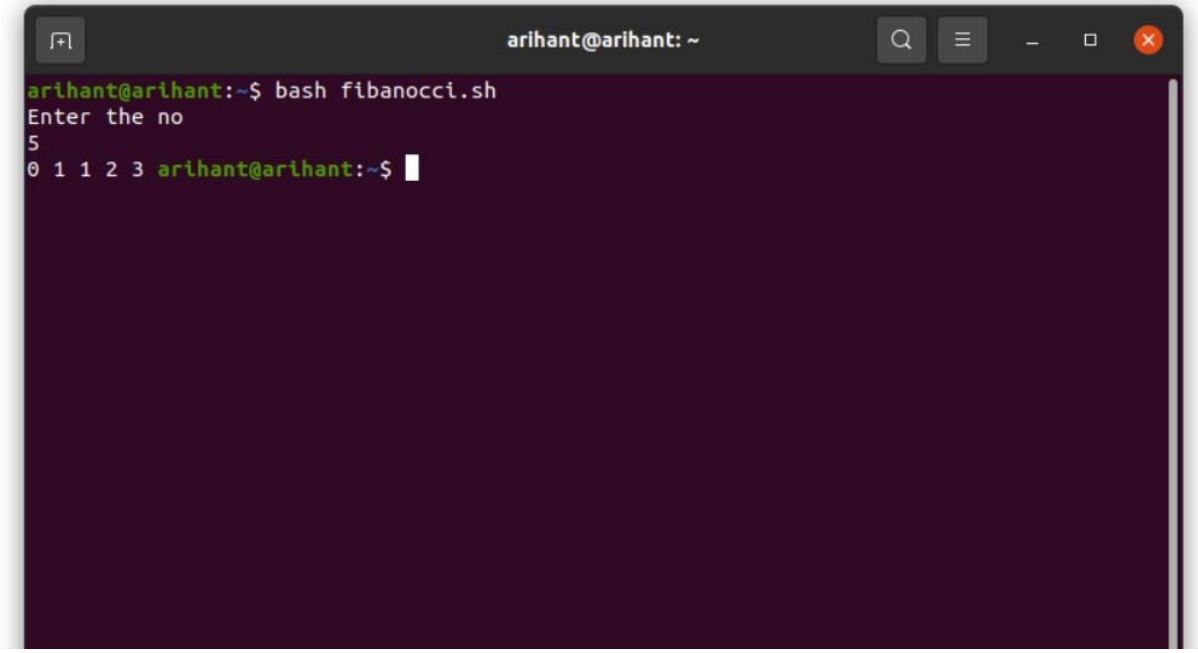


PROGRAM14: Shell script to find the Fibonacci series up to n

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:

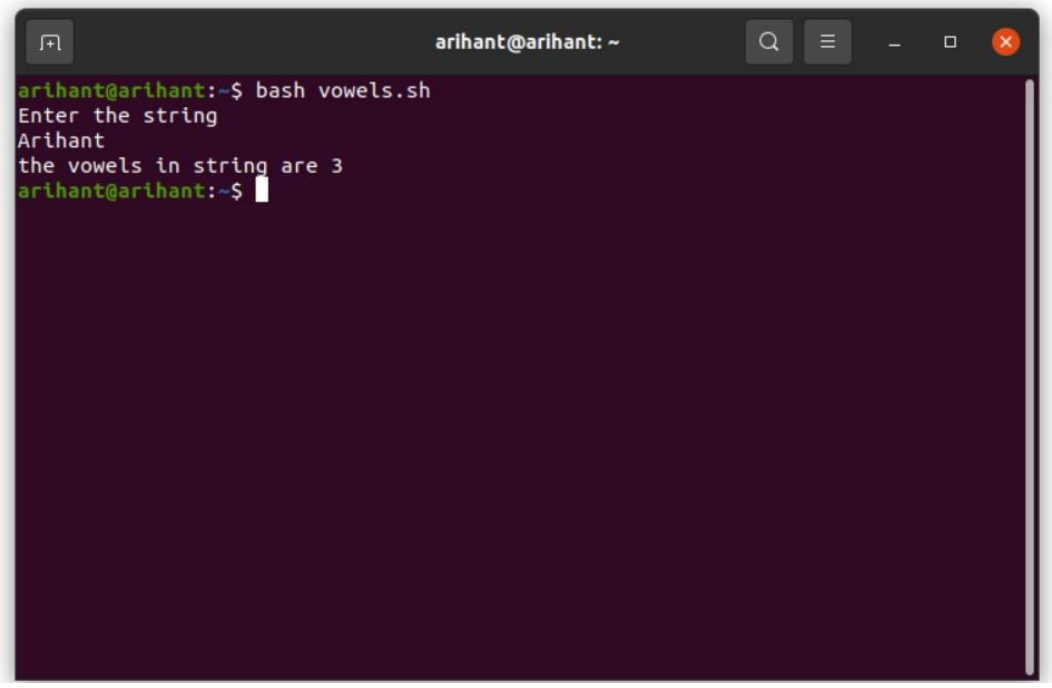


PROGRAM15: Shell script to count the number of vowels of a string

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:

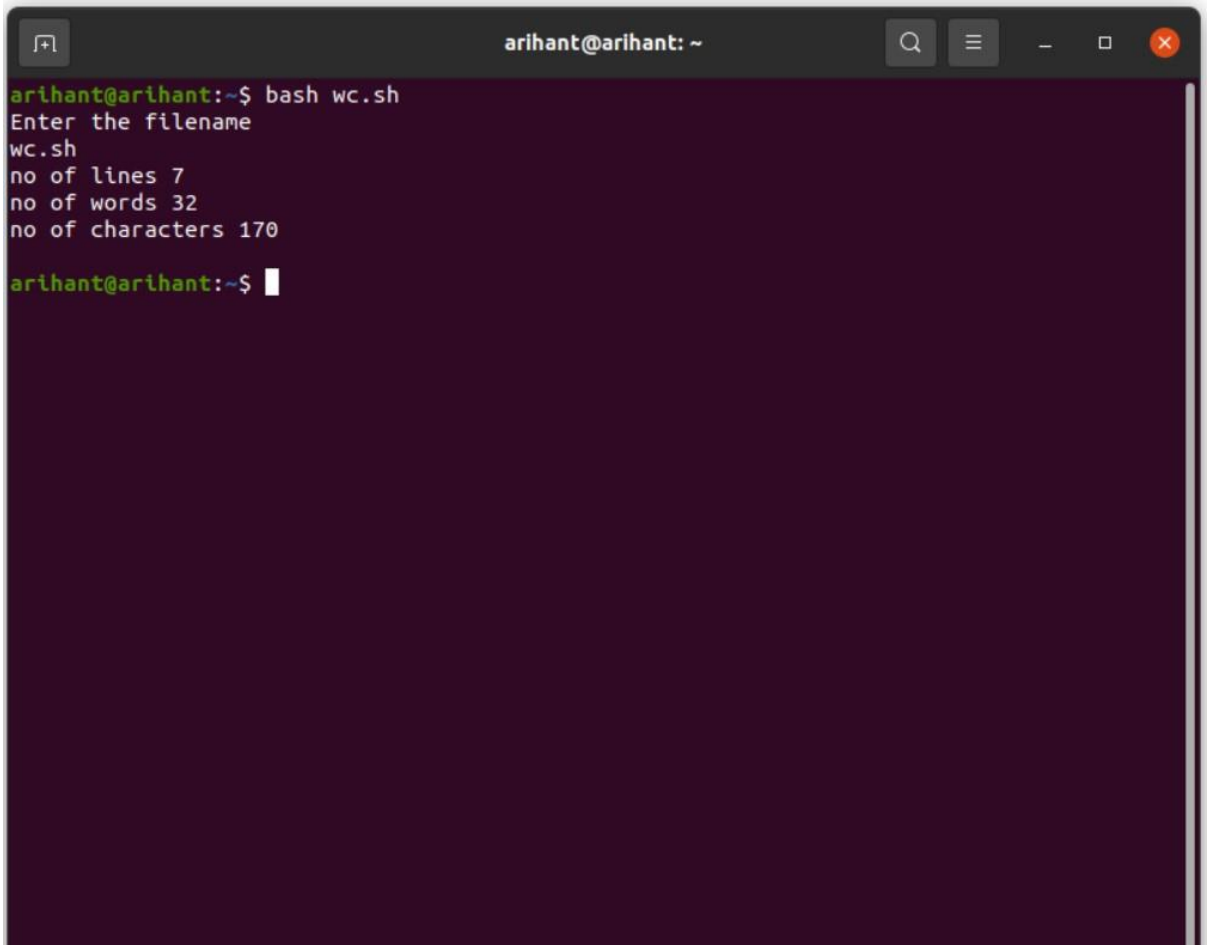


PROGRAM16: Shell script to check number of lines, words, characters in a file

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:

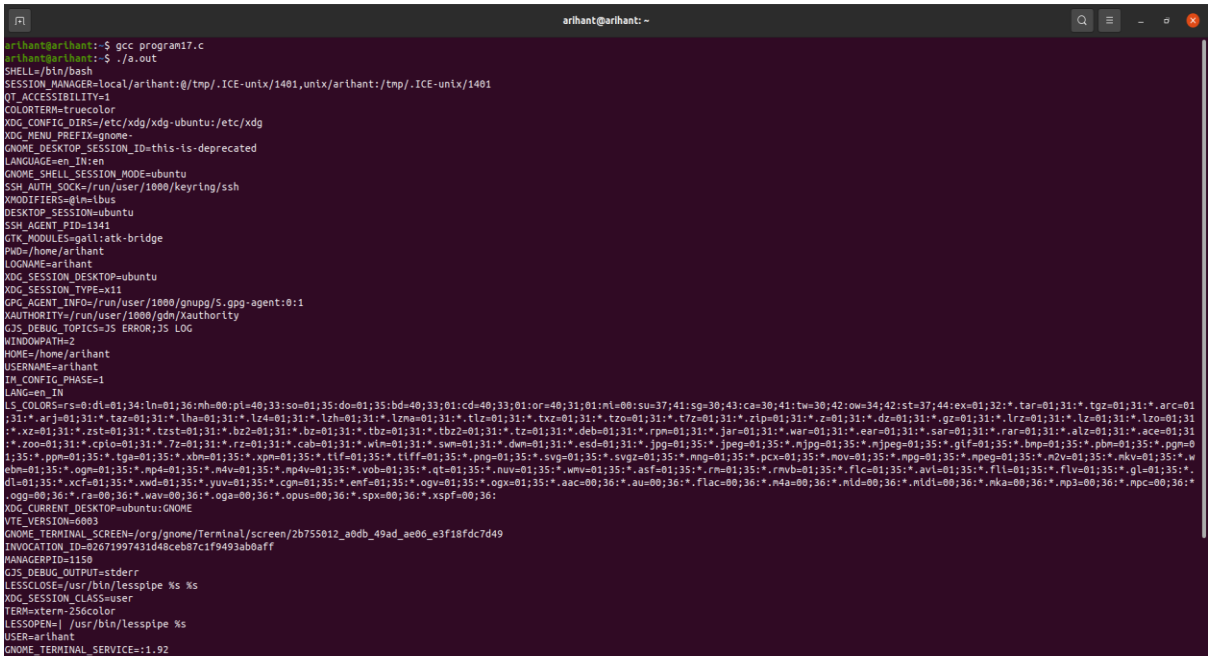


PROGRAM17: Write a C/C++ program to that outputs the contents of its Environment list

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:



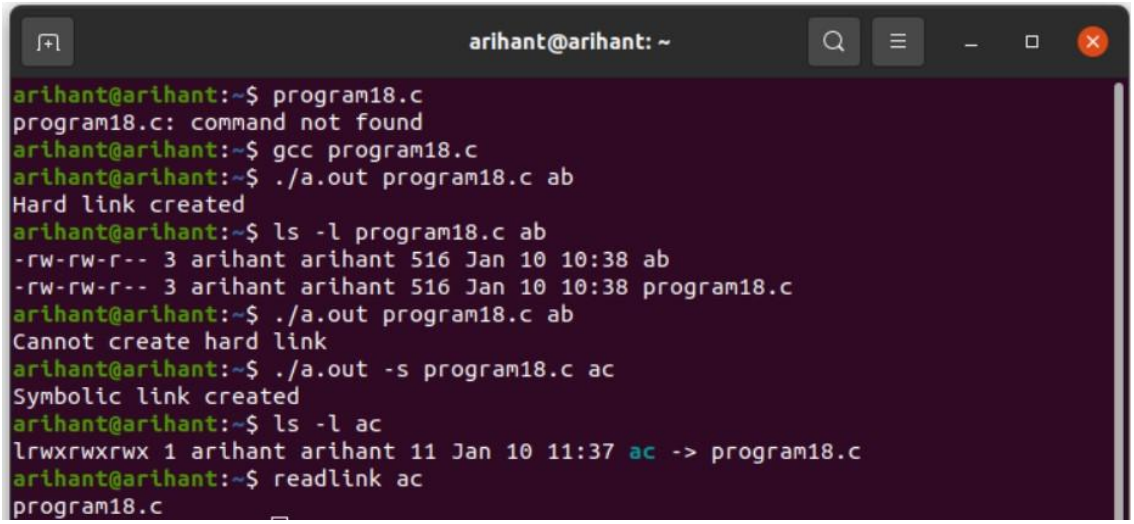
PROGRAM18: Write a C/C++ program to emulate the unix ln command

CODE:

```
#include<stdio.h>

#include<sys/types.h>

#include<unistd.h>

#include<string.h>
```

```
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:



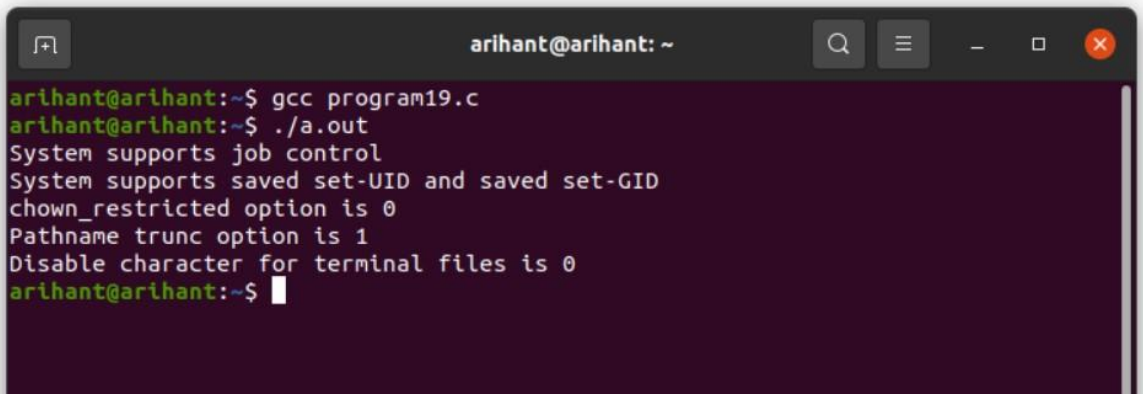
PROGRAM19: Write a C/C++ POSIX compliant program that prints the POSIX defined configuration options supported on any given system using feature test macros.

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:



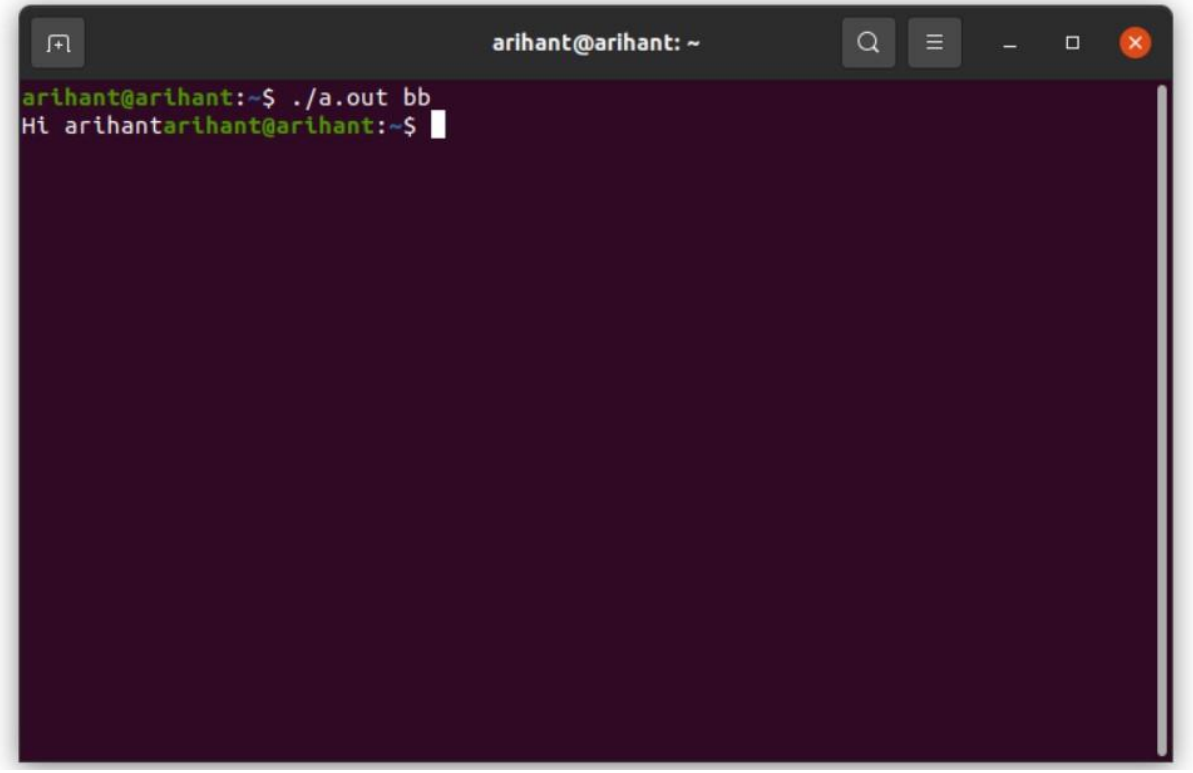
PROGRAM20: Write a C/C++ program which demonstrates interprocess communication between a reader process and a writer process. Use mkfifo, open, read, write and close APIs in your program.

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:


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