MAHARISHI DAYANAND UNIVERSITY

Delhi Global Institute of Technology

Operating System Lab

Subject Code : LC-CSE-212G

Name : Bazgha Razi

Registration Number : 191380214

Roll Number :

INDEX

Program 1

Aim: What is unix?

UNIX

The UNIX operating system is a set of programs that act as a link between the computer and the user.

The computer programs that allocate the system resources and coordinate all the details of the computer's internals is called the operating system or kernel.

Users communicate with the kernel through a program known as the shell. The shell is a command line interpreter; it translates commands entered by the user and converts them into a language that is understood by the kernel.

• Unix was originally developed in 1969 by a group of AT&T employees at Bell Labs, including Ken Thompson, Dennis Ritchie, Douglas McIlroy, and Joe Ossanna.

• There are various Unix variants available in the market. Solaris Unix, AIX, UP Unix and BSD are few examples. Linux is also a flavour of Unix which is freely available.

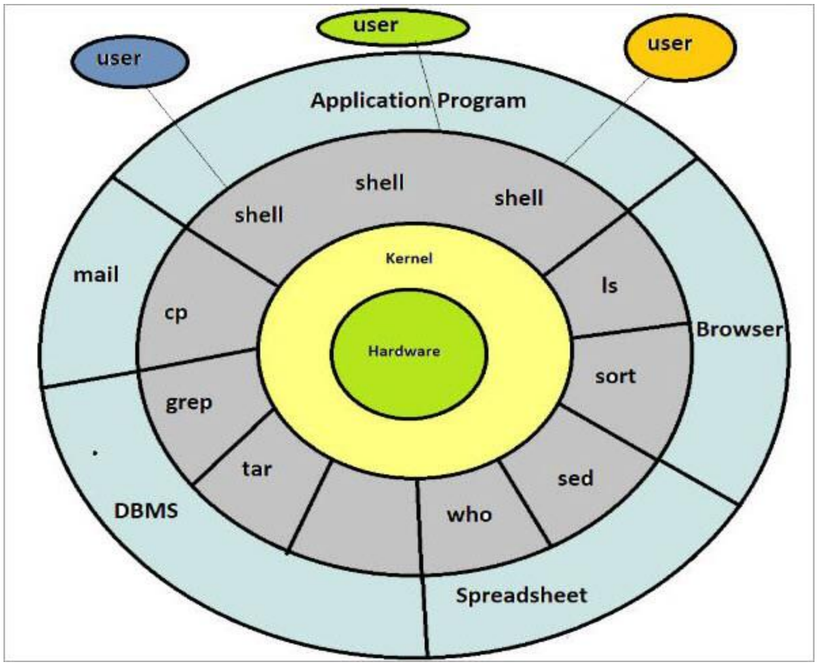
• Several people can use a UNIX computer at the same time; hence UNIX is called a multiuser system.

• A user can also run multiple programs at the same time; hence UNIX is called multitasking.

Program 2

Aim: Structure of UNIX

Structure of UNIX



The main concept that unites all the versions of Unix is the following four basics −

- **Kernel** − The kernel is the heart of the operating system. It interacts with the hardware and most of the tasks like memory management, task scheduling and file management.
- **Shell** − The shell is the utility that processes your requests. When you type in a command at your terminal, the shell interprets the command and calls the program that you want. The shell uses standard syntax for all commands. C Shell, Bourne Shell and Korn Shell are the most famous shells which are available with most of the Unix variants.
- **Commands and Utilities** − There are various commands and utilities which you can make use of in your day to day activities. **cp**, **mv**, **cat** and **grep**, etc. are few examples of commands and utilities. There are over 250 standard commands plus numerous others provided through 3rd party software. All the commands come along with various options.
- **Files and Directories** − All the data of Unix is organized into files. All files are then organized into directories. These directories are further organized into a tree-like structure called the **filesystem**.

Program 3

Aim: Difference between UNIX and Windows

DIFFERENCE BETWEEN
UNIX & WINDOWS

NO:	UNIX	WINDOWS
1	It is a Free Source OS.	It is a licensed OS.
2	UNIX uses daemons.	Windows has service processes.
3	UNIX is more secure.	Windows is more vulnerable.
4	Unix is much better at handling multiple tasks for a single user or for multiple users than windows.	Windows is inferior in this regard.
5	UNIX preferred by programmers for its more flexible nature.	Windows preferred by less sophisticated users.
6	When a new process is created by a UNIX application, it becomes a child of the process that created it.	Windows processes on the other hand do not share a hierarchical relationship.
7	It uses the Unix file system (UFS) also known as Fast File System (FFS)	Windows uses the FAT32 and NTFS File system.

Program 4

Aim: Commands of UNIX.

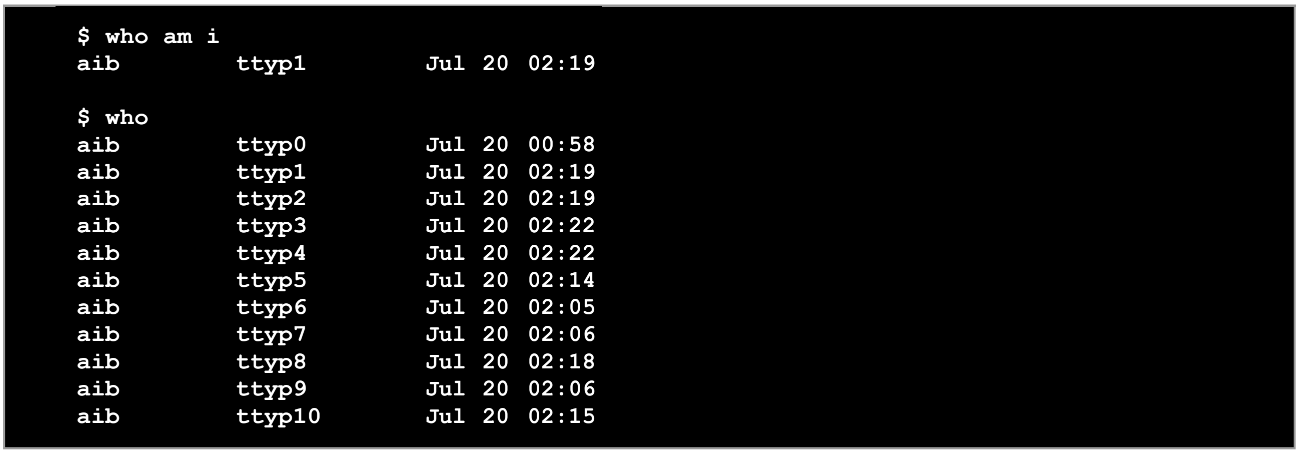
BASIC

UNIX COMMANDS

The various commands are:

who

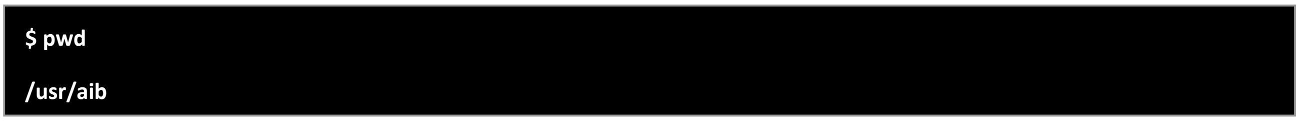
who : Shows other nodes connected to same server. who am I : Shows your node.



```
$ who am i
aib        ttyp1         Jul 20 02:19

$ who
aib        ttyp0         Jul 20 00:58
aib        ttyp1         Jul 20 02:19
aib        ttyp2         Jul 20 02:19
aib        ttyp3         Jul 20 02:22
aib        ttyp4         Jul 20 02:22
aib        ttyp5         Jul 20 02:14
aib        ttyp6         Jul 20 02:05
aib        ttyp7         Jul 20 02:06
aib        ttyp8         Jul 20 02:18
aib        ttyp9         Jul 20 02:06
aib        ttyp10        Jul 20 02:15
```

pwd

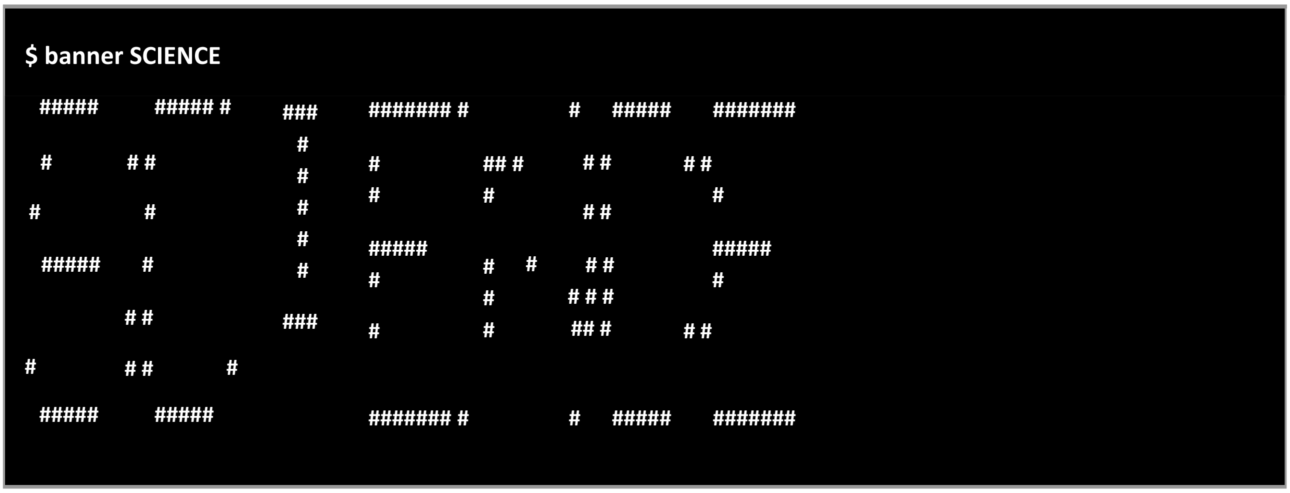
pwd : Present Working Directory



```
$ pwd

/usr/aib
```

Banner

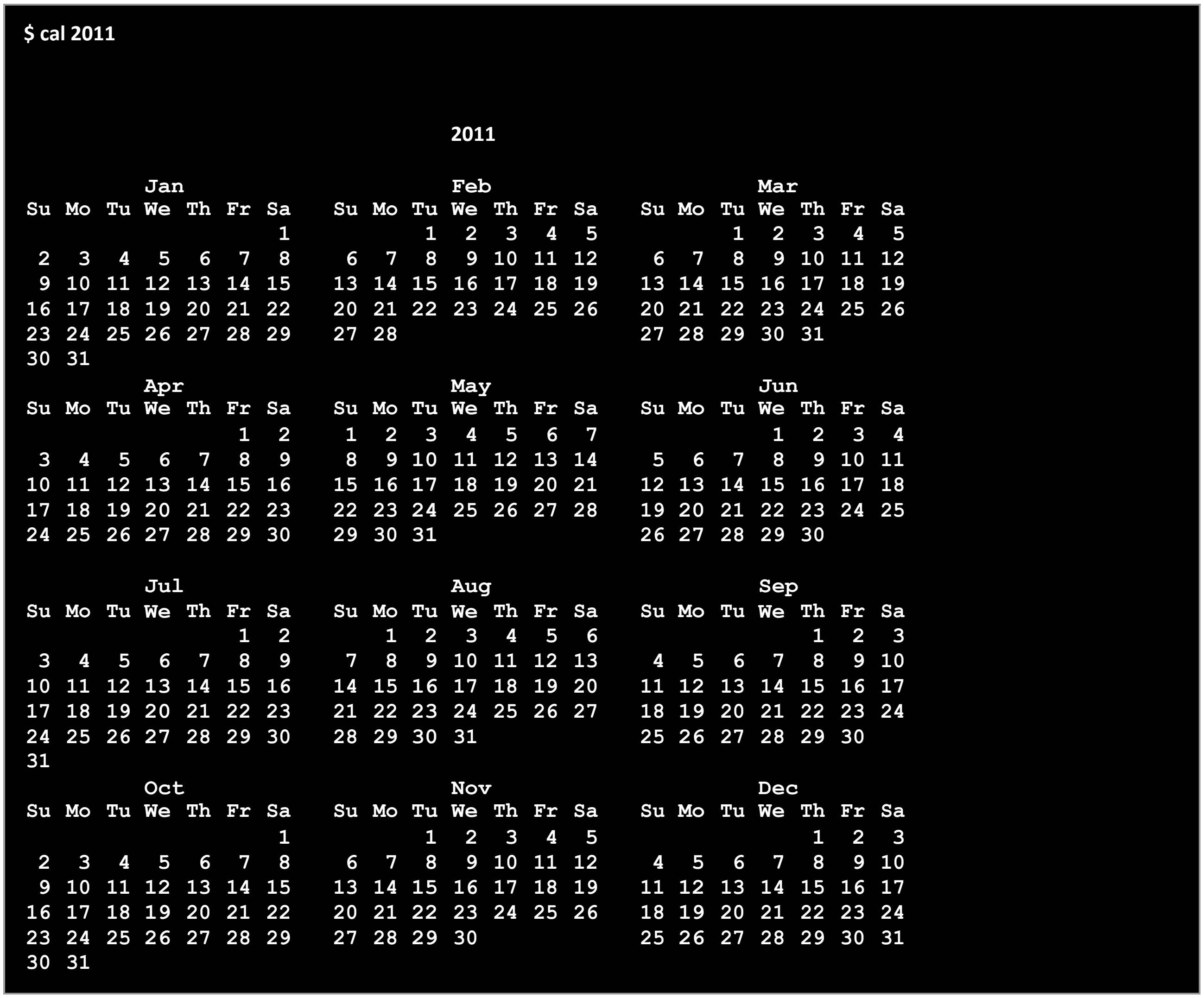
banner <name> : display <name> in a banner-like format



```
$ banner SCIENCE

 #####     ##### #    ###    ####### #        #   #####    #######
                       #
 #       # #           #     #         ## #     # #     # #
                             #         #                   #
#         #            #                        # #
                       #
                             #####                         #####
 #####    #            #               #   #    # #
                             #                             #
                                       #      # # #
         # #          ###
                             #         #      ## #      # #
#        # #     #
 #####     #####             ####### #        #   #####    #######
```

cal

cal : Calender



```
$ cal 2011


                                  2011

        Jan                    Feb                    Mar
Su Mo Tu We Th Fr Sa   Su Mo Tu We Th Fr Sa   Su Mo Tu We Th Fr Sa
                   1          1  2  3  4  5          1  2  3  4  5
 2  3  4  5  6  7  8    6  7  8  9 10 11 12    6  7  8  9 10 11 12
 9 10 11 12 13 14 15   13 14 15 16 17 18 19   13 14 15 16 17 18 19
16 17 18 19 20 21 22   20 21 22 23 24 25 26   20 21 22 23 24 25 26
23 24 25 26 27 28 29   27 28                  27 28 29 30 31
30 31
        Apr                    May                    Jun
Su Mo Tu We Th Fr Sa   Su Mo Tu We Th Fr Sa   Su Mo Tu We Th Fr Sa
                1  2    1  2  3  4  5  6  7             1  2  3  4
 3  4  5  6  7  8  9    8  9 10 11 12 13 14    5  6  7  8  9 10 11
10 11 12 13 14 15 16   15 16 17 18 19 20 21   12 13 14 15 16 17 18
17 18 19 20 21 22 23   22 23 24 25 26 27 28   19 20 21 22 23 24 25
24 25 26 27 28 29 30   29 30 31               26 27 28 29 30

        Jul                    Aug                    Sep
Su Mo Tu We Th Fr Sa   Su Mo Tu We Th Fr Sa   Su Mo Tu We Th Fr Sa
                1  2       1  2  3  4  5  6                1  2  3
 3  4  5  6  7  8  9    7  8  9 10 11 12 13    4  5  6  7  8  9 10
10 11 12 13 14 15 16   14 15 16 17 18 19 20   11 12 13 14 15 16 17
17 18 19 20 21 22 23   21 22 23 24 25 26 27   18 19 20 21 22 23 24
24 25 26 27 28 29 30   28 29 30 31            25 26 27 28 29 30
31
        Oct                    Nov                    Dec
Su Mo Tu We Th Fr Sa   Su Mo Tu We Th Fr Sa   Su Mo Tu We Th Fr Sa
                   1          1  2  3  4  5                1  2  3
 2  3  4  5  6  7  8    6  7  8  9 10 11 12    4  5  6  7  8  9 10
 9 10 11 12 13 14 15   13 14 15 16 17 18 19   11 12 13 14 15 16 17
16 17 18 19 20 21 22   20 21 22 23 24 25 26   18 19 20 21 22 23 24
23 24 25 26 27 28 29   27 28 29 30            25 26 27 28 29 30 31
30 31
```

date

date +% _ : Shows date, with specified formats

```
$date

Wed Jul 20 02:26:01 IST 2011


$ date +%H <-- Hours (Time Capital) 02

$ date +%M <-- Minutes 26

$ date +%S <-- Seconds 19


$ date

Wed Jul 20 02:26:31 IST 2011


$ date +%d <-- Day (Date LowerCase) 20

$ date +%m <-- Month 07

$ date +%y <-- Year (two digits) 11

$ date +%Y <-- Year (all four digits)

2011
```

touch

touch <filename> : Creates files without data

cat

cat > (existing/new filename) : Write data to file (also creates)
cat < (existing filename) : Read data from file

```
$ touch FOLM1 FOLM2 FOLM3

$ cat > FOLM2
Yosh.

$ cat > FOLM1
DOOM DOT.

$ cat < FOLM2
Yosh.
```

mv

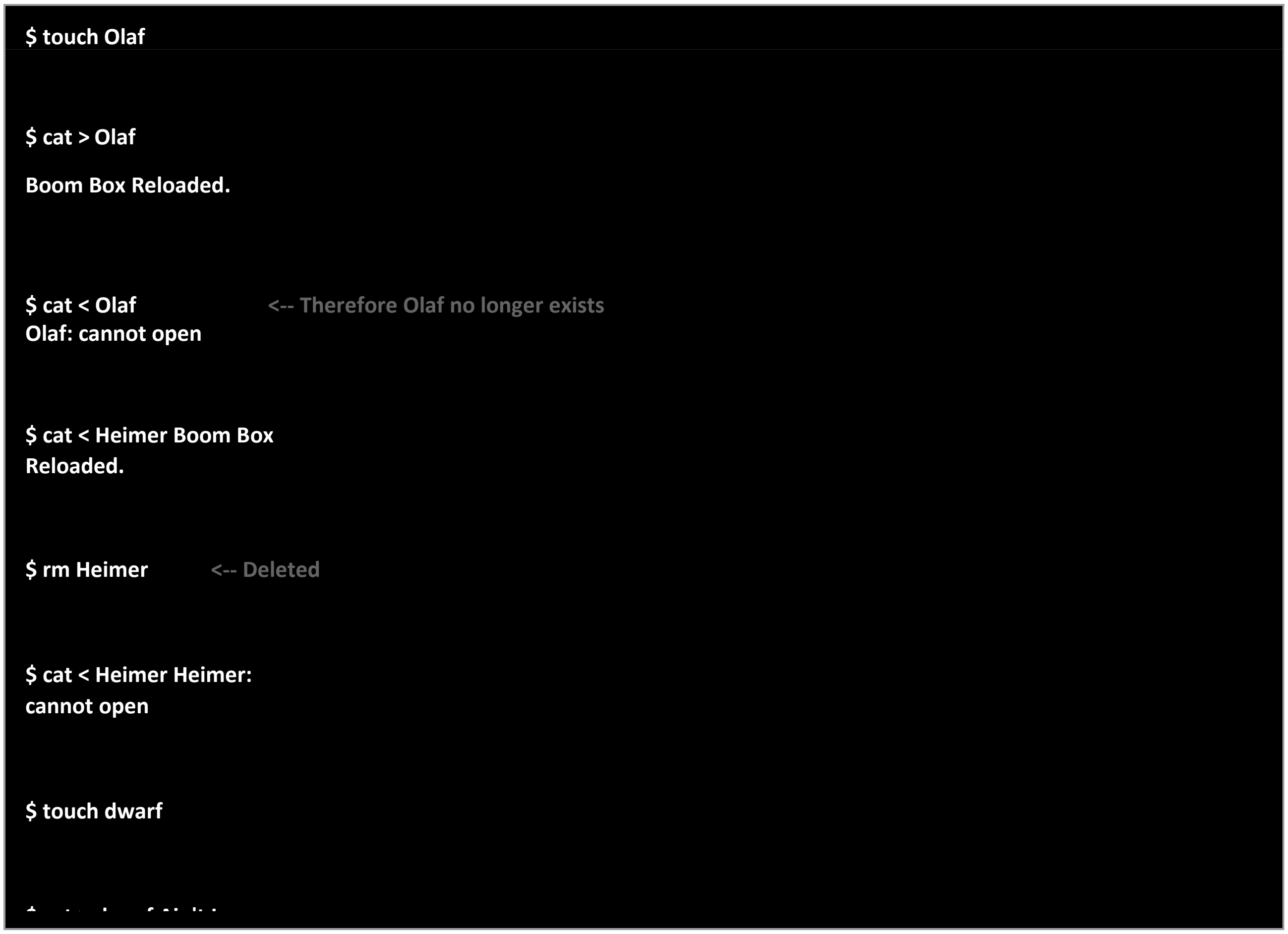
mv <filename1> <filename2> : Renames file from 1 to 2

cp

cp <filename1> <filename2> : Copies from one to another

rm

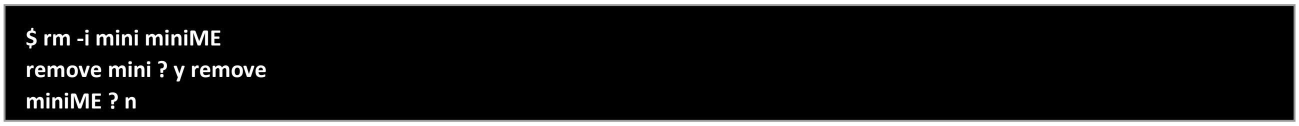
rm <filename1> : deletes file



```
$ touch Olaf


$ cat > Olaf

Boom Box Reloaded.



$ cat < Olaf                <-- Therefore Olaf no longer exists
Olaf: cannot open


$ cat < Heimer Boom Box
Reloaded.


$ rm Heimer          <-- Deleted


$ cat < Heimer Heimer:
cannot open


$ touch dwarf


[illegible]
```

-i

<command> -i <files> : With Permission



```
$ rm -i mini miniME
remove mini ? y remove
miniME ? n
```

wc

wc <filename> : Word Count (lines, words, characters)

-l : only lines
-w : only words
-c : only characters
-wl : words & lines
-wc : words & characters
-lc : lines & characters

```
$ cat < PUMA2 Jump &
move it. Jump & move
it.. Jump & move it... Yo!


$ wc PUMA2

           4           13            55 PUMA2

    $ wc -w PUMA2

           13 PUMA2

     $ wc -c PUMA2
```

Head

head -n <filename> : displays the first n lines of a file

Tail

tail -n <filename> : displays the last n lines of a file

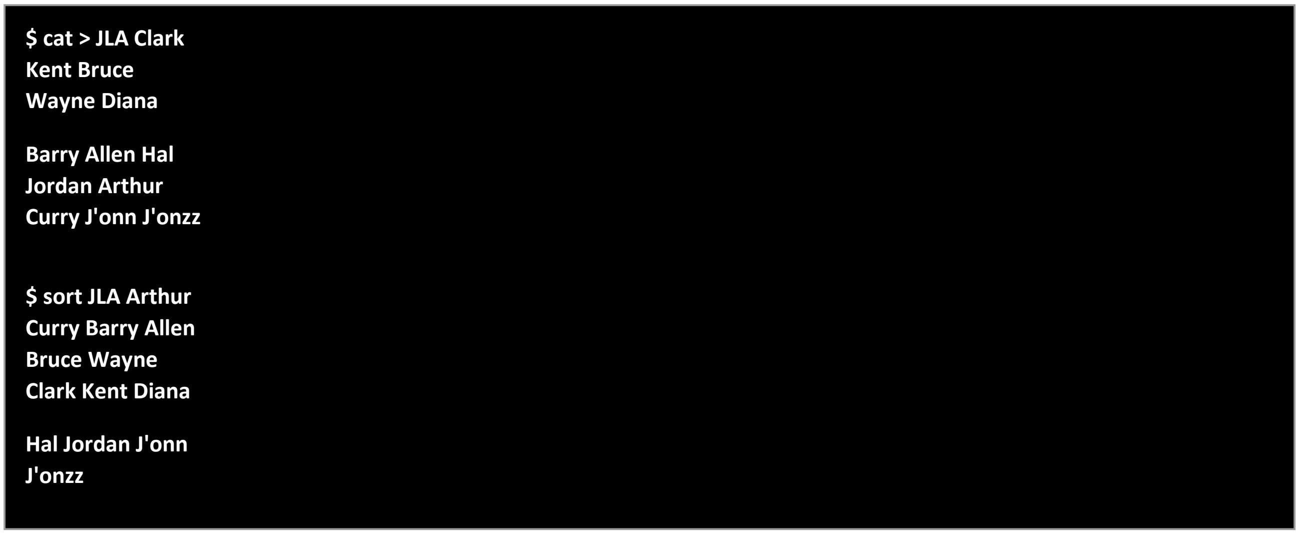
```
$ cat < PUMA2 Jump &
move it. Jump & move
it.. Jump & move it... Yo!


$ head -1 PUMA2
Jump & move it.


$ tail -2 PUMA2 Jump &
move it... Yo!
```

Sort

sort <filename> : Sorts files taking each line as separate entity
sort -r <filename> : reverse order (descending)
sort -o<to_filename> <from_filename> : save sorted file

sort <filename> <filename2> : Combines both and sorts it all.
sort -o<TO_file> <FROM_file1> <FROM_file2> : Saves the combined and sorted.

-m : treats each file as an entity instead of the lines within it
-u : avoids repetition when common entities from 2 files repeat



```
$ cat > JLA Clark
Kent Bruce
Wayne Diana

Barry Allen Hal
Jordan Arthur
Curry J'onn J'onzz


$ sort JLA Arthur
Curry Barry Allen
Bruce Wayne
Clark Kent Diana

Hal Jordan J'onn
J'onzz
```

Program 5

Aim: VI Editor

VI EDITOR

Vi editor, is a widely-used and popular UNIX-based text editor.

Like most UNIX system interfaces and other text editors, it lets you control the system by using the keyboard rather than a combination of mouse selections and keystrokes. The succinctness of the interface makes it highly useful for people who work at a computer all day, especially programmers entering or manipulating language statements.

There are two modes in the Vi Editor.

1. Command Mode
2. Insertion Mode

The editor begins in command mode, where the cursor movement and text deletion and pasting occur. Insertion mode begins upon entering an insertion or change command. [ESC] returns the editor to command mode. Most commands execute as soon as you type them except for "colon" commands which execute when you press the return key.

ENTERING THE INSERT MODE

a append text, after the cursor;

i insert text, before the cursor;

R enter Overtype Mode;

A append text, after end of line;

I insert text, before first non-whitespace character;

o open new line below cursor in Insert Mode;

O open new line above cursor in Insert Mode;

EXITING THE INSERT MODE

ESC Exit insertion mode and return to command mode

Program 6

Aim: Shell Scripting

SHELL SCRIPTING

A shell script is a text file that contains a sequence of commands for a UNIX-based operating system. It's called a shell script because it combines into a single file (script) a sequence of commands that would otherwise have to be presented to the system from a keyboard one at a time.

A shell script is usually created for command sequences for which a user has a repeated need. You initiate the sequence of commands in the shell script by simply entering the name of the shell script on a command line.

Often, writing a shell script is much quicker than writing the equivalent code in other programming languages. The many advantages include easy program or file selection, quick start, and interactive debugging. A shell script can be used to provide a sequencing and decision-making linkage around existing programs, and for moderately-sized scripts the absence of a compilation step is an advantage.

On the other hand, shell scripting is prone to costly errors. Inadvertent typing errors such as rm -rf * / are folklore in the Unix community; a single extra space converts the command from one that deletes everything in the sub-directories to one which deletes everything - and also tries to delete everything in the root directory. Similar problems can transform cp and mv into dangerous weapons, and misuse of the > redirect can delete the contents of a file. This is made more problematic by the fact that many UNIX commands differ in name by only one letter: cp, cd, dd, df, etc.

Another significant disadvantage is the slow execution speed and the need to launch a new process for almost every shell command executed. When a script's job can be accomplished by setting up a pipeline in which efficient filter commands perform most of the work, the slowdown is mitigated, but a complex script is typically several orders of magnitude slower than a conventional compiled program that performs an equivalent task.

LOOPING SYNTAX

FOR LOOP

Syntax:

```
for { <variable_name> } in { <list of variable> } do
                    <commands>
    done
```

WHILE LOOP

Syntax:

```
    while [ <condition> ] do
                              <command1>
                              <command2>
```

UNTIL LOOP

Syntax:

```
    until cond
                        do
                              Statement(s) to be executed until command is true done
```

DECISION MAKING SYNTAX

SWITCH CASE

It is a control statement used for decision making.

Syntax:

```
case $<variable> in


<option1>)

<statement>

;;
```

IF CONDITION

It is a control statement used for decision making.

Syntax:

```
if [ $<variable> <logical operator> $<variable> then

<statement> fi
```

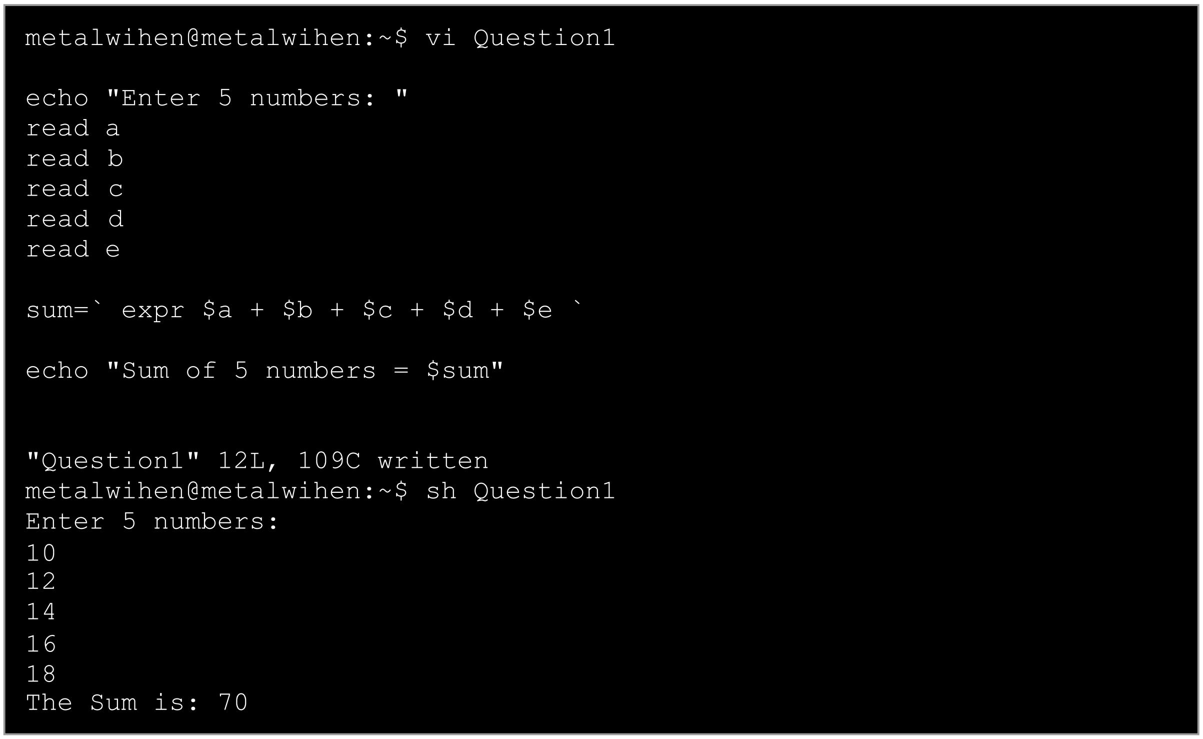
(IF - ELSE IF)

```
if [ $<variable> <logical operator> $<variable> ] then

<statement>

elif [$<variable> <logical operator> $<variable> ] then

<statement>
else

<statement> fi
```

Program 7

Aim: Find Sum of 5 Natural Numbers

FIND SUM OF 5 NATURAL NUMBERS

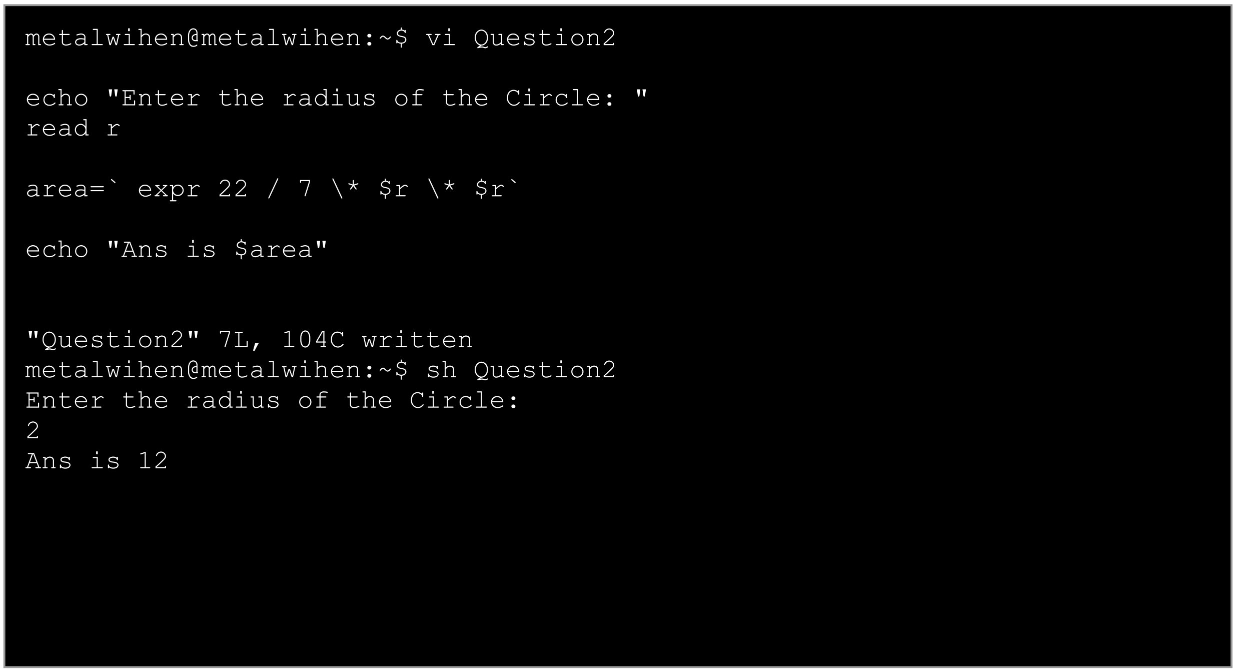


```
metalwihen@metalwihen:~$ vi Question1

echo "Enter 5 numbers: "
read a
read b
read c
read d
read e

sum=` expr $a + $b + $c + $d + $e `

echo "Sum of 5 numbers = $sum"


"Question1" 12L, 109C written
metalwihen@metalwihen:~$ sh Question1
Enter 5 numbers:
10
12
14
16
18
The Sum is: 70
```

Program 8

Aim: Find the area of circle

FIND THE AREA OF CIRCLE

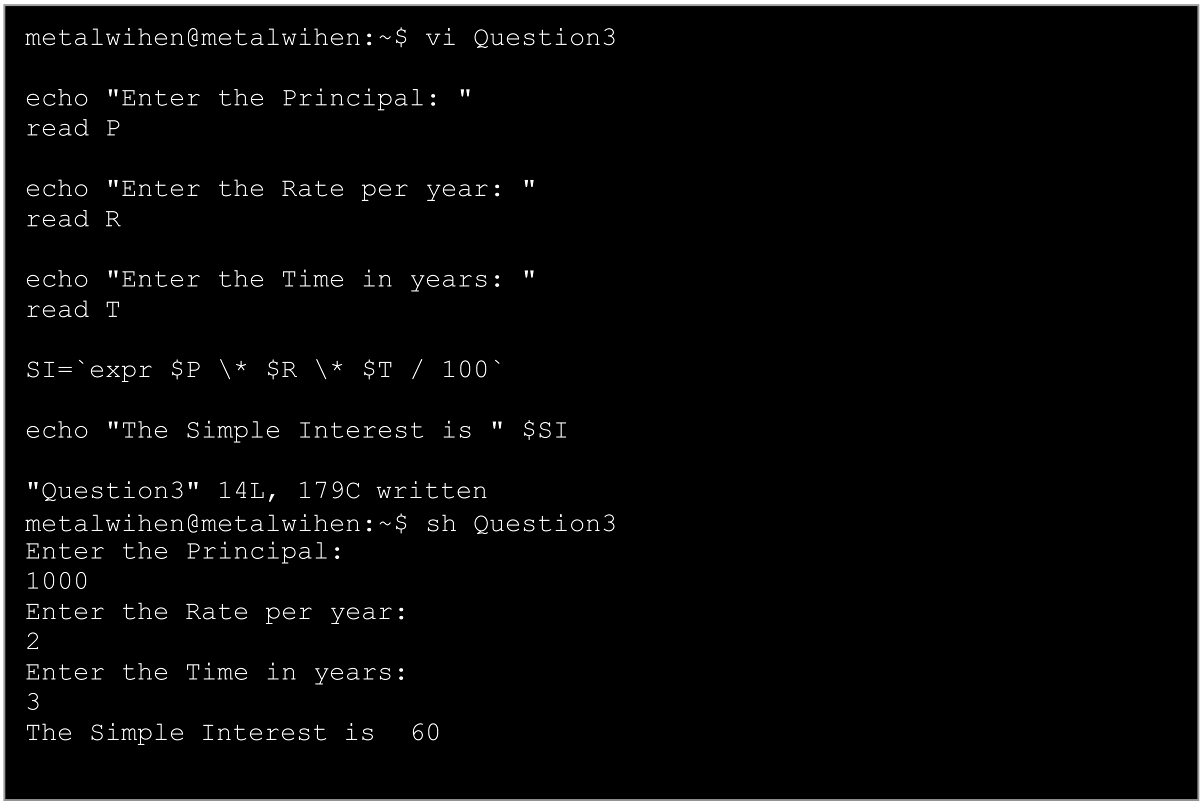


```
metalwihen@metalwihen:~$ vi Question2

echo "Enter the radius of the Circle: "
read r

area=` expr 22 / 7 \* $r \* $r`

echo "Ans is $area"


"Question2" 7L, 104C written
metalwihen@metalwihen:~$ sh Question2
Enter the radius of the Circle:
2
Ans is 12
```

Program 9

Aim: Calculate the simple interest

CALCULATE THE SIMPLE INTEREST

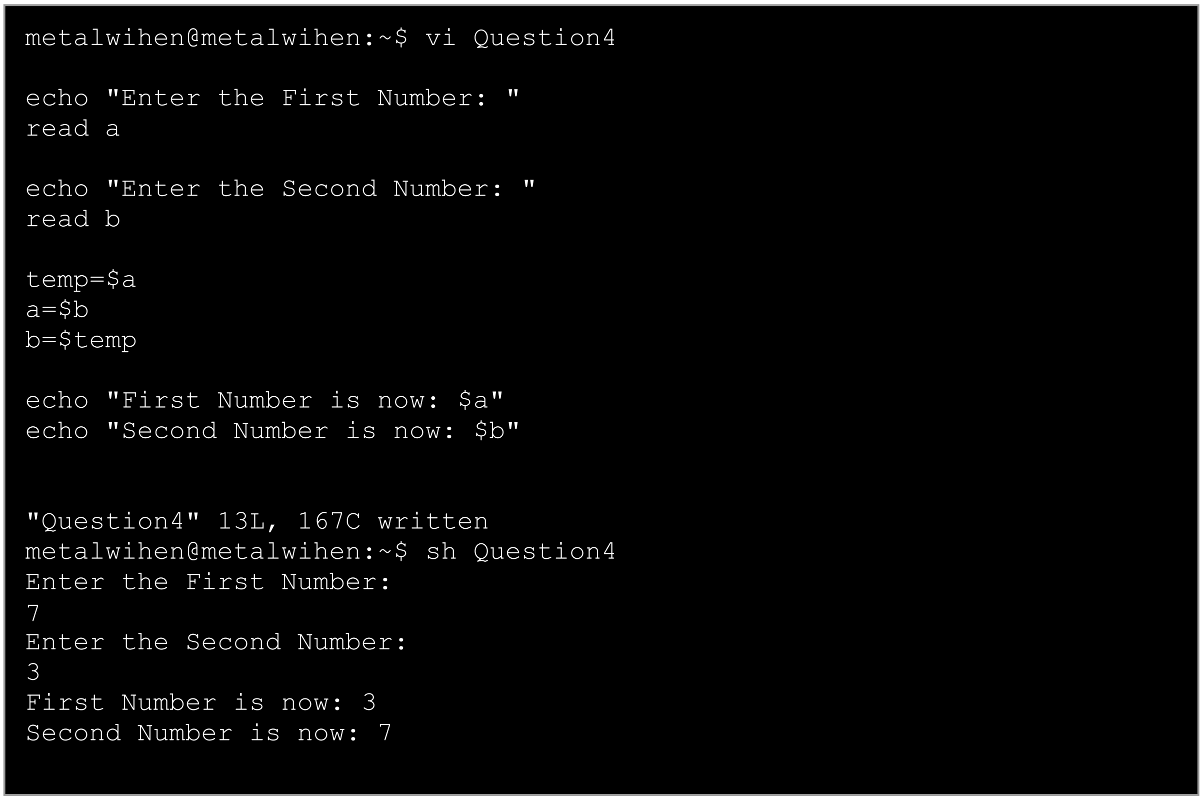


```
metalwihen@metalwihen:~$ vi Question3

echo "Enter the Principal: "
read P

echo "Enter the Rate per year: "
read R

echo "Enter the Time in years: "
read T

SI=`expr $P \* $R \* $T / 100`

echo "The Simple Interest is " $SI

"Question3" 14L, 179C written
metalwihen@metalwihen:~$ sh Question3
Enter the Principal:
1000
Enter the Rate per year:
2
Enter the Time in years:
3
The Simple Interest is  60
```

Program 10

Aim: Swap two numbers.

SWAP TWO NUMBERS

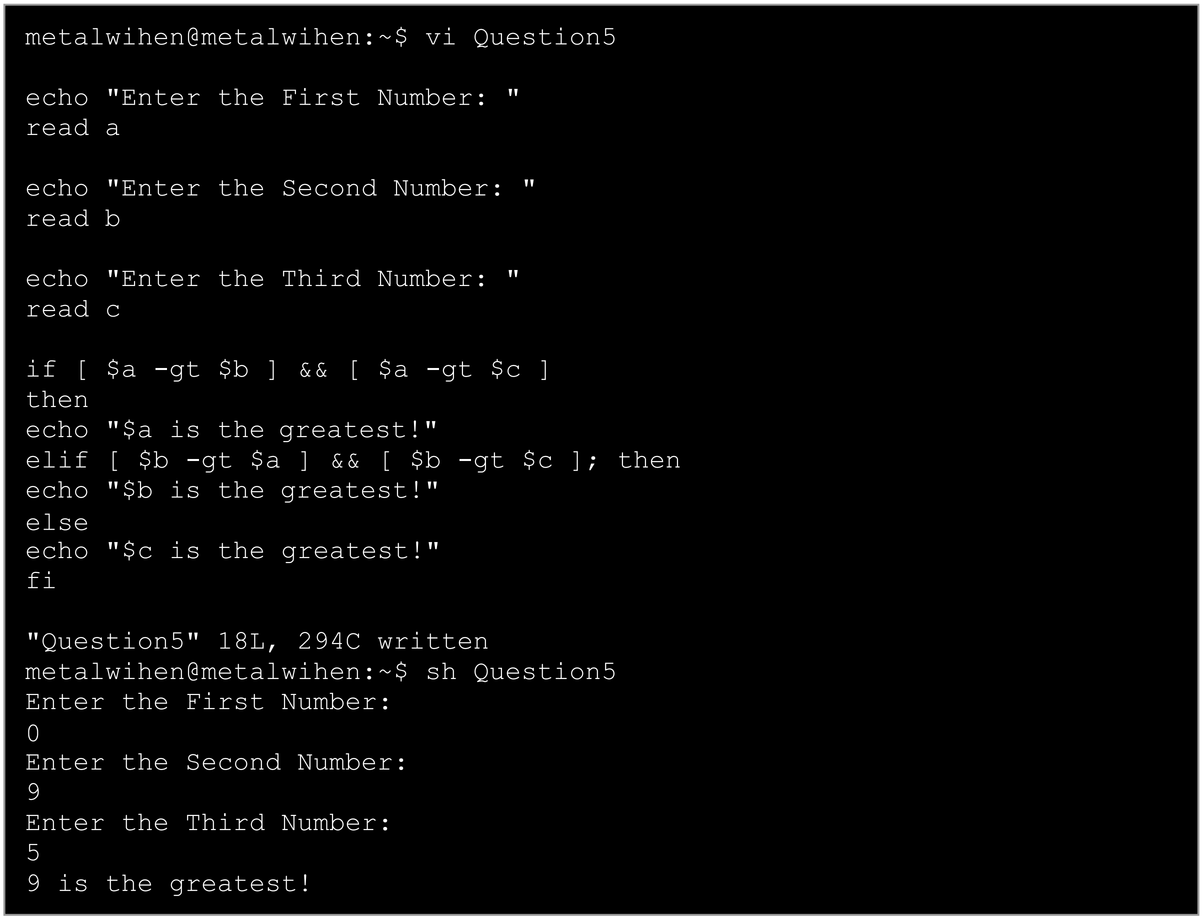


```
metalwihen@metalwihen:~$ vi Question4

echo "Enter the First Number: "
read a

echo "Enter the Second Number: "
read b

temp=$a
a=$b
b=$temp

echo "First Number is now: $a"
echo "Second Number is now: $b"


"Question4" 13L, 167C written
metalwihen@metalwihen:~$ sh Question4
Enter the First Number:
7
Enter the Second Number:
3
First Number is now: 3
Second Number is now: 7
```

Program 11

Aim: Find the largest number.

FIND THE LARGEST NUMBER



```
metalwihen@metalwihen:~$ vi Question5

echo "Enter the First Number: "
read a

echo "Enter the Second Number: "
read b

echo "Enter the Third Number: "
read c

if [ $a -gt $b ] && [ $a -gt $c ]
then
echo "$a is the greatest!"
elif [ $b -gt $a ] && [ $b -gt $c ]; then
echo "$b is the greatest!"
else
echo "$c is the greatest!"
fi

"Question5" 18L, 294C written
metalwihen@metalwihen:~$ sh Question5
Enter the First Number:
0
Enter the Second Number:
9
Enter the Third Number:
5
9 is the greatest!
```

Program 12

Aim: Check if number is a prime number.

CHECK IF NUMBER IS A PRIME NO

```
metalwihen@metalwihen:~$ vi Question12

echo "Enter the number:"
read n

flag=1
i=2

while [ $i -lt $n ]
do

rem=`expr $n % $i`

if [ $rem -eq 0 ]
then
flag=0
fi


i=`expr $i + 1`
done


if [ $flag -eq 0 ]
then
echo "Number is NOT Prime."
else
echo "Number is Prime."
fi

"Question13" 30L, 236C written
metalwihen@metalwihen:~$ sh Question13
Enter the number:
5
Number is Prime.

metalwihen@metalwihen:~$ sh Question13
Enter the number:
6
Number is NOT Prime.
```