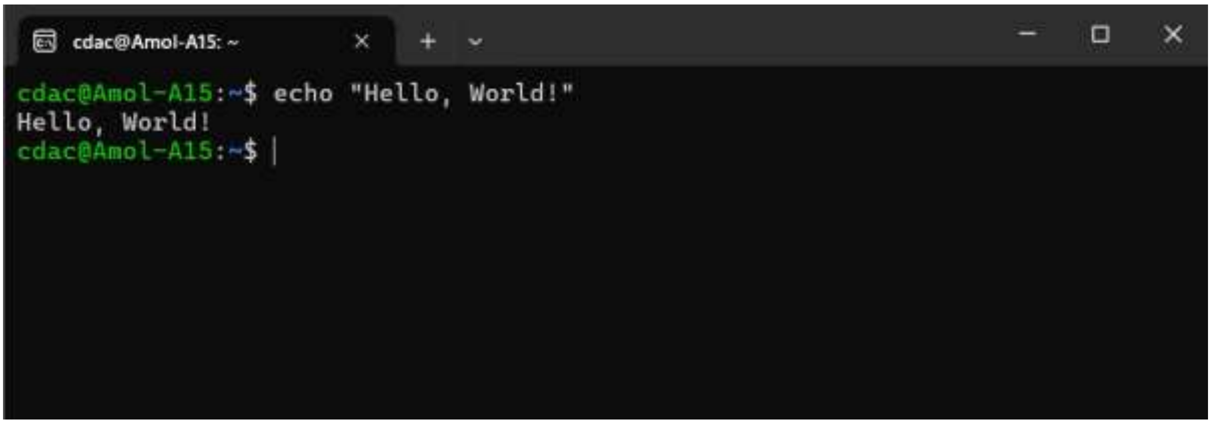
Concepts of Operating System

Assignment 2

Part A

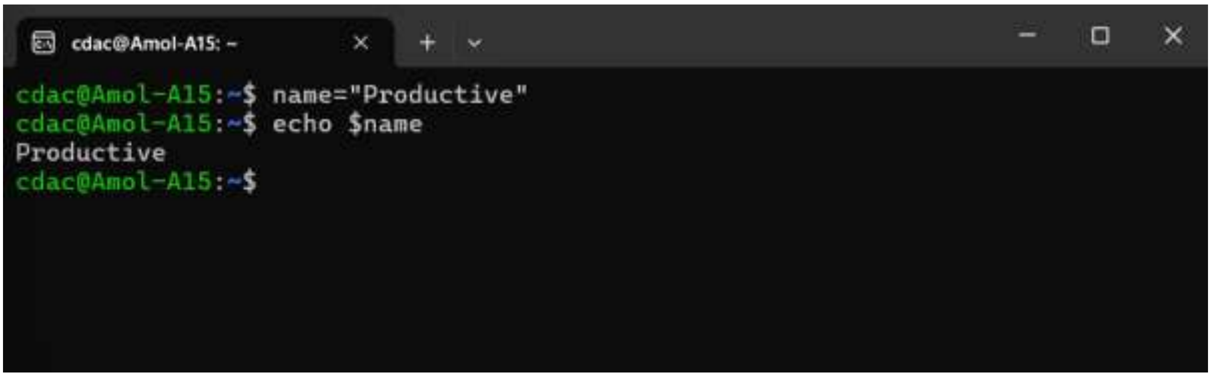
What will the following commands do?

1. echo "Hello, World!"
   Display message on screen
   Hello, World



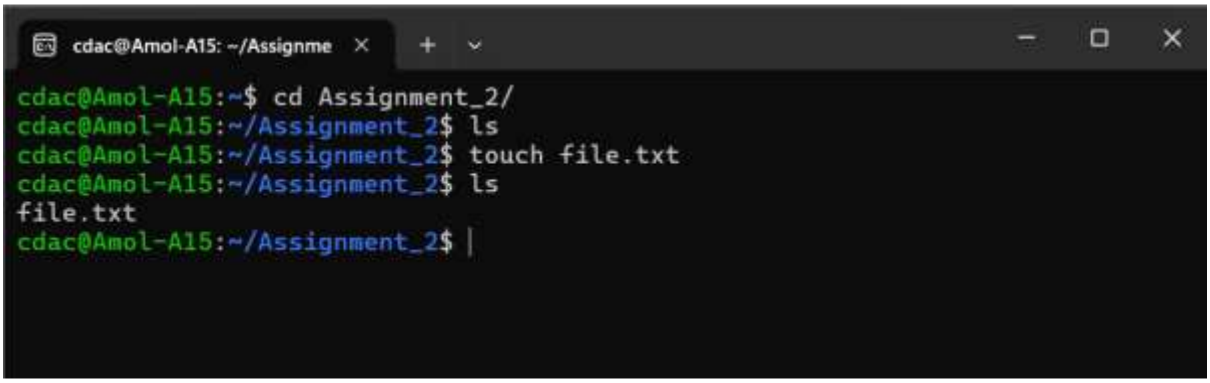
```
cdac@Amol-A15:~$ echo "Hello, World!"
Hello, World!
cdac@Amol-A15:~$ |
```

2. name="Productive"
   Create a variable with the value as string {Productive}



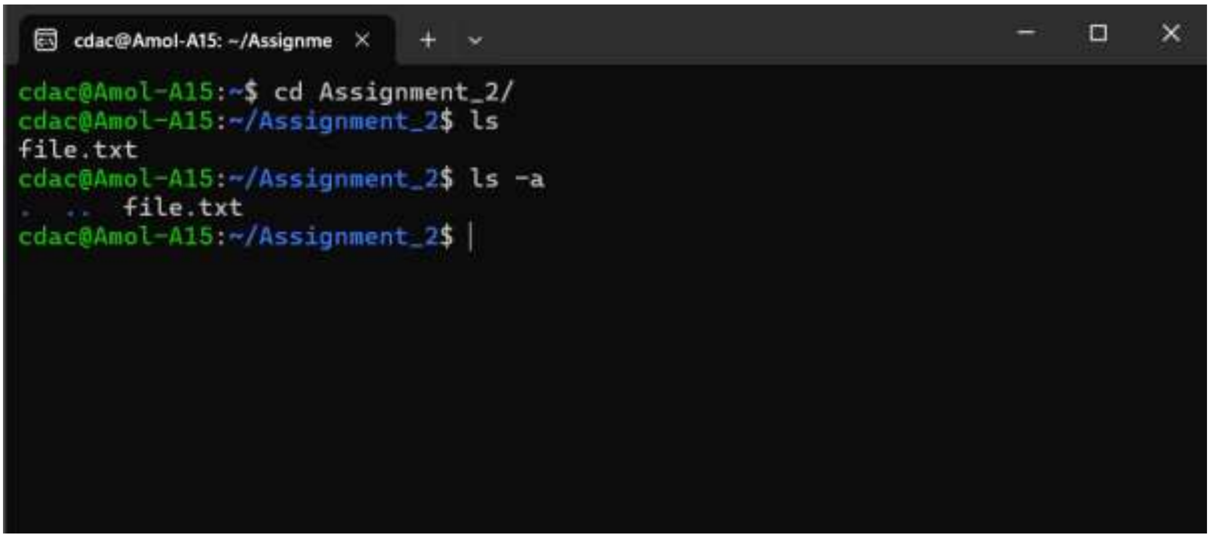
```
cdac@Amol-A15:~$ name="Productive"
cdac@Amol-A15:~$ echo $name
Productive
cdac@Amol-A15:~$
```

3. touch file.txt
   is use to create a new file



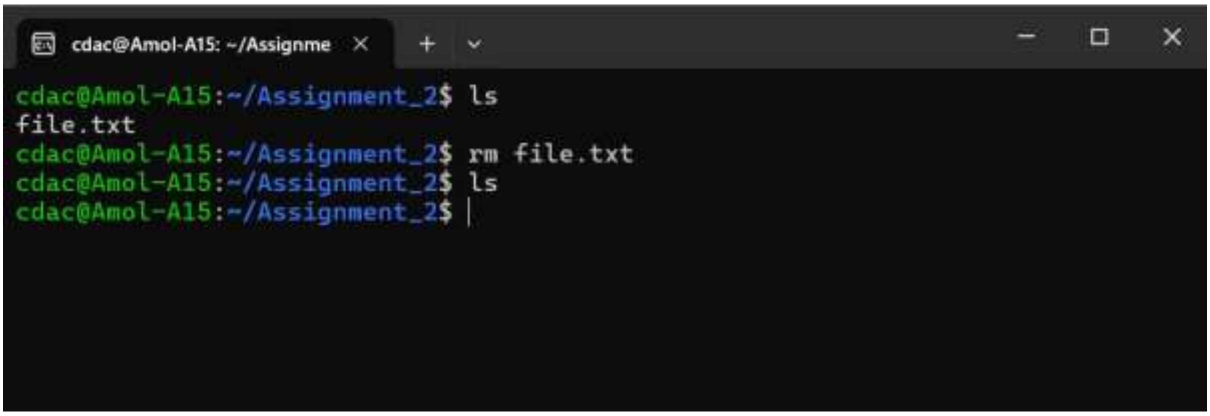
```
cdac@Amol-A15:~$ cd Assignment_2/
cdac@Amol-A15:~/Assignment_2$ ls
cdac@Amol-A15:~/Assignment_2$ touch file.txt
cdac@Amol-A15:~/Assignment_2$ ls
file.txt
cdac@Amol-A15:~/Assignment_2$ |
```

4. ls -a
   list directory contents
   -a is use to not ignore entries starting with { . }



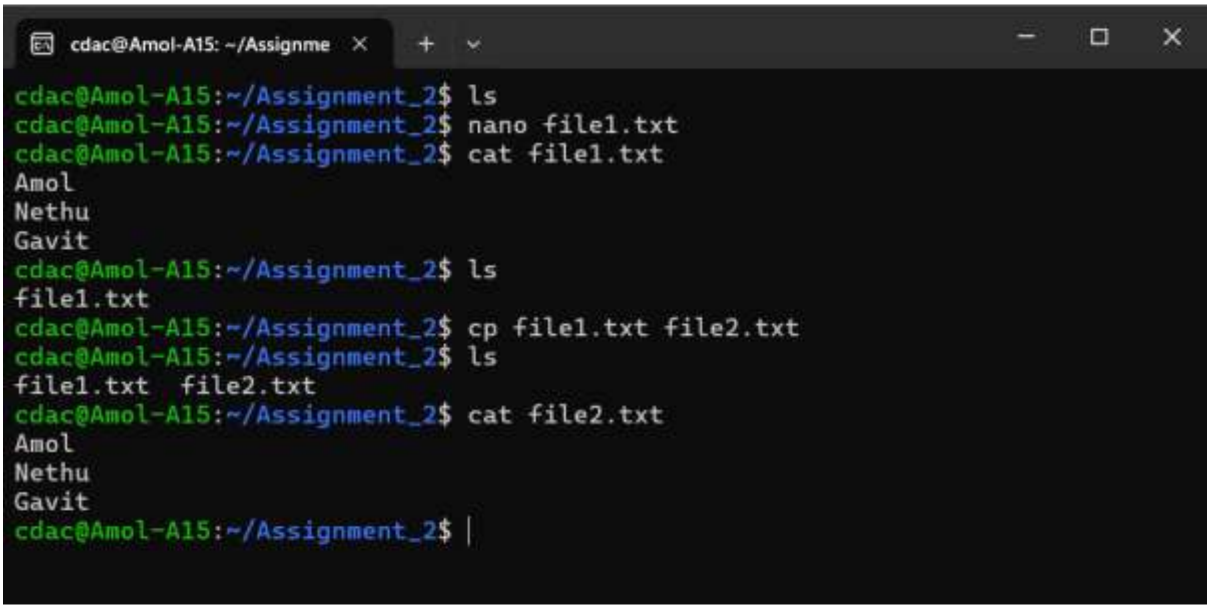
```
cdac@Amol-A15:~$ cd Assignment_2/
cdac@Amol-A15:~/Assignment_2$ ls
file.txt
cdac@Amol-A15:~/Assignment_2$ ls -a
.  ..  file.txt
cdac@Amol-A15:~/Assignment_2$ |
```

5. rm file.txt
   remove files or directories



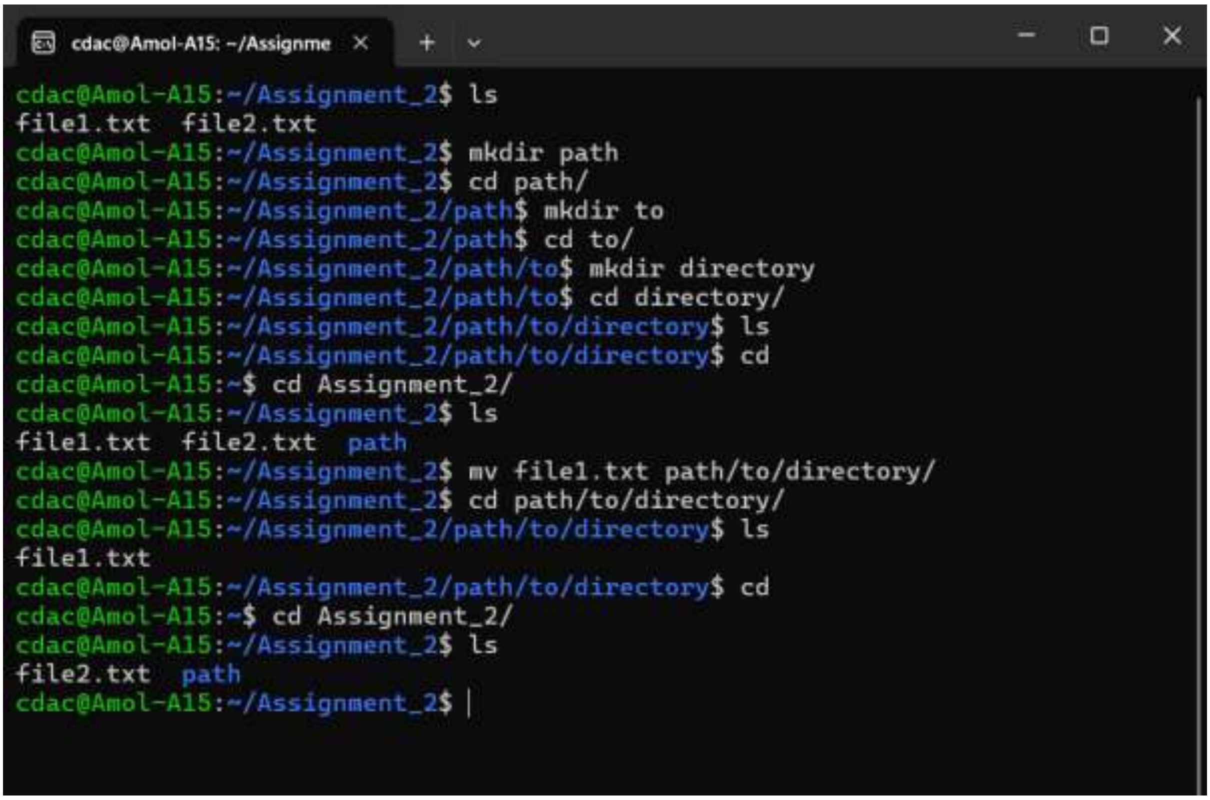
```
cdac@Amol-A15:~/Assignment_2$ ls
file.txt
cdac@Amol-A15:~/Assignment_2$ rm file.txt
cdac@Amol-A15:~/Assignment_2$ ls
cdac@Amol-A15:~/Assignment_2$ |
```

6. cp file1.txt file2.txt
   copy files and directories
   used to copy the contains of file1.txt to file2.txt



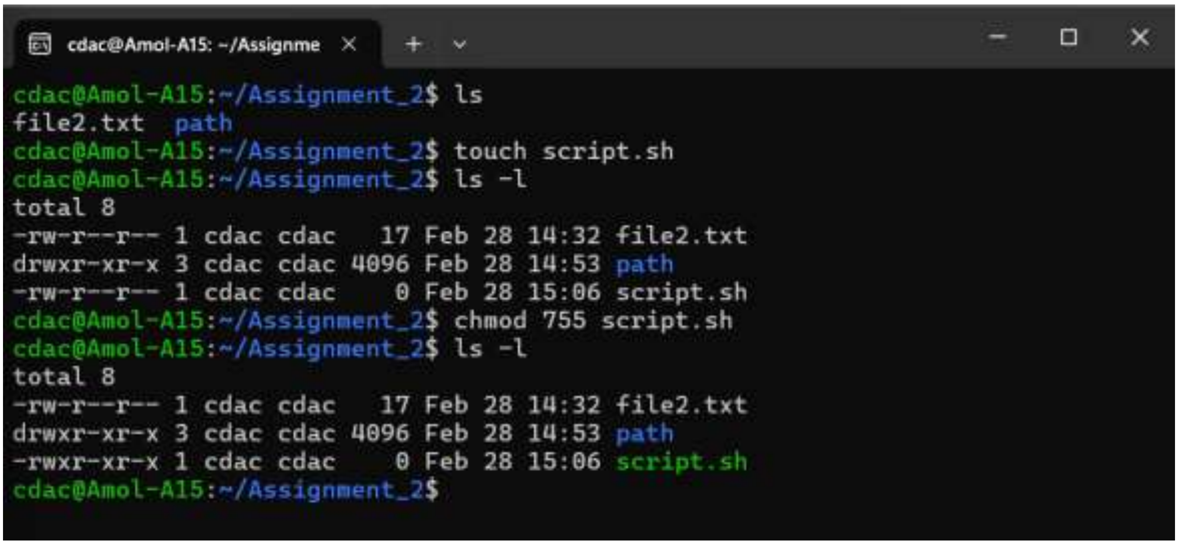
```
cdac@Amol-A15:~/Assignment_2$ ls
cdac@Amol-A15:~/Assignment_2$ nano file1.txt
cdac@Amol-A15:~/Assignment_2$ cat file1.txt
Amol
Nethu
Gavit
cdac@Amol-A15:~/Assignment_2$ ls
file1.txt
cdac@Amol-A15:~/Assignment_2$ cp file1.txt file2.txt
cdac@Amol-A15:~/Assignment_2$ ls
file1.txt  file2.txt
cdac@Amol-A15:~/Assignment_2$ cat file2.txt
Amol
Nethu
Gavit
cdac@Amol-A15:~/Assignment_2$ |
```

7. mv file1.txt /path/to/directory/
   move files



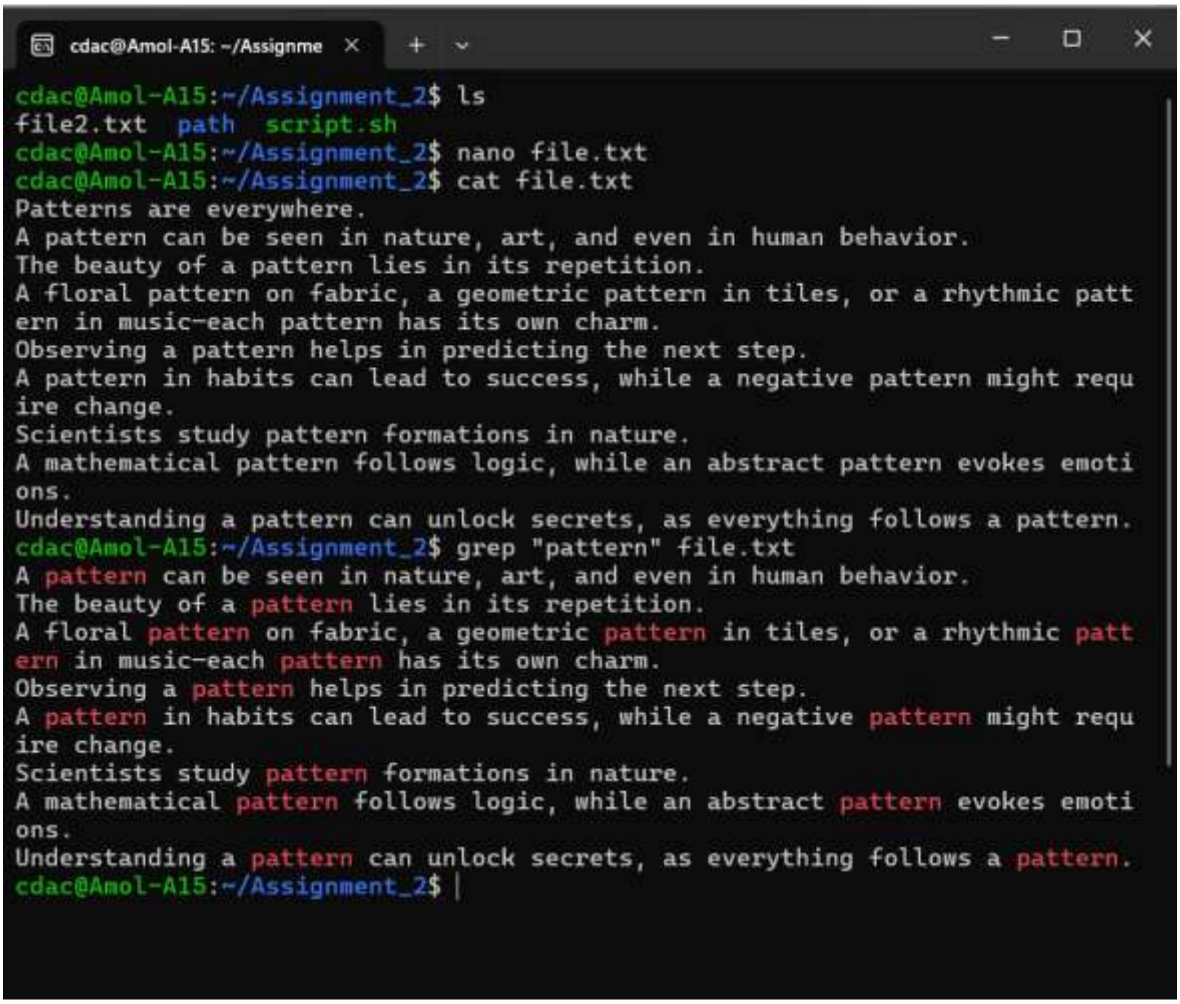
```
cdac@Amol-A15:~/Assignment_2$ ls
file1.txt  file2.txt
cdac@Amol-A15:~/Assignment_2$ mkdir path
cdac@Amol-A15:~/Assignment_2$ cd path/
cdac@Amol-A15:~/Assignment_2/path$ mkdir to
cdac@Amol-A15:~/Assignment_2/path$ cd to/
cdac@Amol-A15:~/Assignment_2/path/to$ mkdir directory
cdac@Amol-A15:~/Assignment_2/path/to$ cd directory/
cdac@Amol-A15:~/Assignment_2/path/to/directory$ ls
cdac@Amol-A15:~/Assignment_2/path/to/directory$ cd
cdac@Amol-A15:~$ cd Assignment_2/
cdac@Amol-A15:~/Assignment_2$ ls
file1.txt  file2.txt  path
cdac@Amol-A15:~/Assignment_2$ mv file1.txt path/to/directory/
cdac@Amol-A15:~/Assignment_2$ cd path/to/directory/
cdac@Amol-A15:~/Assignment_2/path/to/directory$ ls
file1.txt
cdac@Amol-A15:~/Assignment_2/path/to/directory$ cd
cdac@Amol-A15:~$ cd Assignment_2/
cdac@Amol-A15:~/Assignment_2$ ls
file2.txt  path
cdac@Amol-A15:~/Assignment_2$ |
```

8. chmod 755 script.sh
   change file permissions
   Default file permissions are: u=rw, g=r, o=r { 644}
   Change it to: { 755 } means u=rwx, g=rx, o=rx



```
cdac@Amol-A15:~/Assignment_2$ ls
file2.txt  path
cdac@Amol-A15:~/Assignment_2$ touch script.sh
cdac@Amol-A15:~/Assignment_2$ ls -l
total 8
-rw-r--r-- 1 cdac cdac   17 Feb 28 14:32 file2.txt
drwxr-xr-x 3 cdac cdac 4096 Feb 28 14:53 path
-rw-r--r-- 1 cdac cdac    0 Feb 28 15:06 script.sh
cdac@Amol-A15:~/Assignment_2$ chmod 755 script.sh
cdac@Amol-A15:~/Assignment_2$ ls -l
total 8
-rw-r--r-- 1 cdac cdac   17 Feb 28 14:32 file2.txt
drwxr-xr-x 3 cdac cdac 4096 Feb 28 14:53 path
-rwxr-xr-x 1 cdac cdac    0 Feb 28 15:06 script.sh
cdac@Amol-A15:~/Assignment_2$
```

9. grep "pattern" file.txt
   print lines that match patterns
   here we use “pattern” word to search in the { file.txt }



```
cdac@Amol-A15:~/Assignment_2$ ls
file2.txt  path  script.sh
cdac@Amol-A15:~/Assignment_2$ nano file.txt
cdac@Amol-A15:~/Assignment_2$ cat file.txt
Patterns are everywhere.
A pattern can be seen in nature, art, and even in human behavior.
The beauty of a pattern lies in its repetition.
A floral pattern on fabric, a geometric pattern in tiles, or a rhythmic patt
ern in music—each pattern has its own charm.
Observing a pattern helps in predicting the next step.
A pattern in habits can lead to success, while a negative pattern might requ
ire change.
Scientists study pattern formations in nature.
A mathematical pattern follows logic, while an abstract pattern evokes emoti
ons.
Understanding a pattern can unlock secrets, as everything follows a pattern.
cdac@Amol-A15:~/Assignment_2$ grep "pattern" file.txt
A pattern can be seen in nature, art, and even in human behavior.
The beauty of a pattern lies in its repetition.
A floral pattern on fabric, a geometric pattern in tiles, or a rhythmic patt
ern in music—each pattern has its own charm.
Observing a pattern helps in predicting the next step.
A pattern in habits can lead to success, while a negative pattern might requ
ire change.
Scientists study pattern formations in nature.
A mathematical pattern follows logic, while an abstract pattern evokes emoti
ons.
Understanding a pattern can unlock secrets, as everything follows a pattern.
cdac@Amol-A15:~/Assignment_2$ |
```

10. kill PID

    This option specifies the process ID of the process to be killed.

11. mkdir mydir && cd mydir && touch file.txt && echo "Hello, World!" > file.txt && cat file.txt

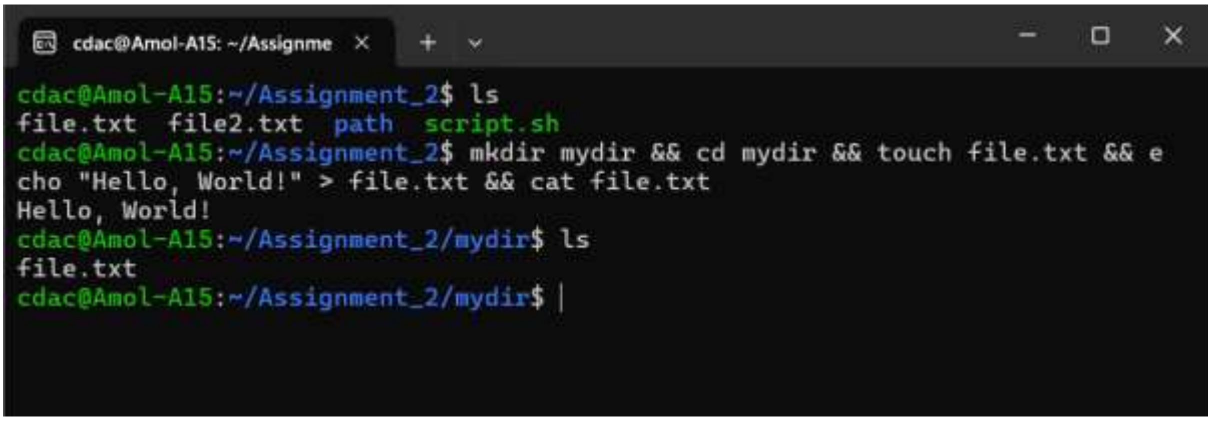
    i) Crate the directory name mydir.

    ii) Change the directory to mydir.

    iii) Create the new file with name as “file.txt” in current directory.

    iv) Then append the output of echo “Hello, World!” in the file.txt.

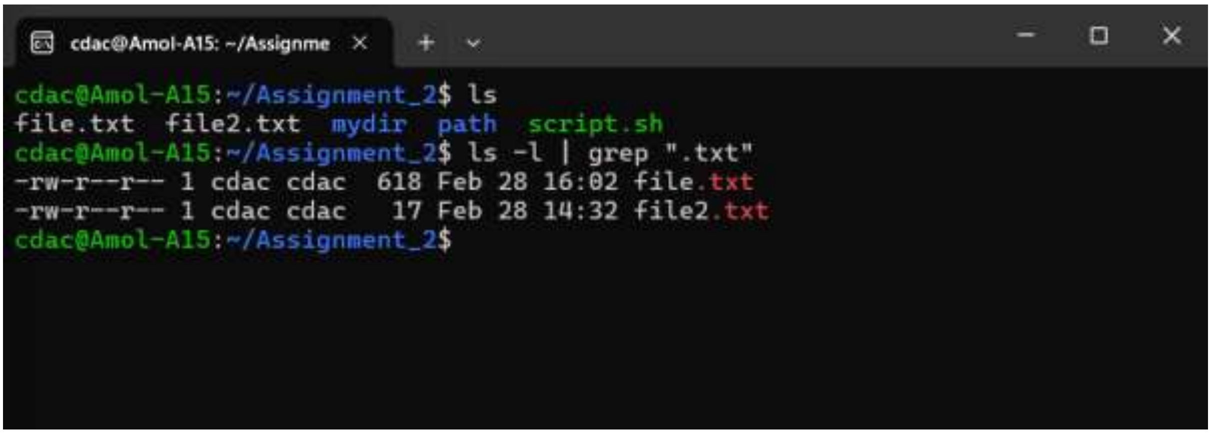
    v) Display the contains of the file.txt.



```
cdac@Amol-A15:~/Assignment_2$ ls
file.txt  file2.txt  path  script.sh
cdac@Amol-A15:~/Assignment_2$ mkdir mydir && cd mydir && touch file.txt && e
cho "Hello, World!" > file.txt && cat file.txt
Hello, World!
cdac@Amol-A15:~/Assignment_2/mydir$ ls
file.txt
cdac@Amol-A15:~/Assignment_2/mydir$ |
```

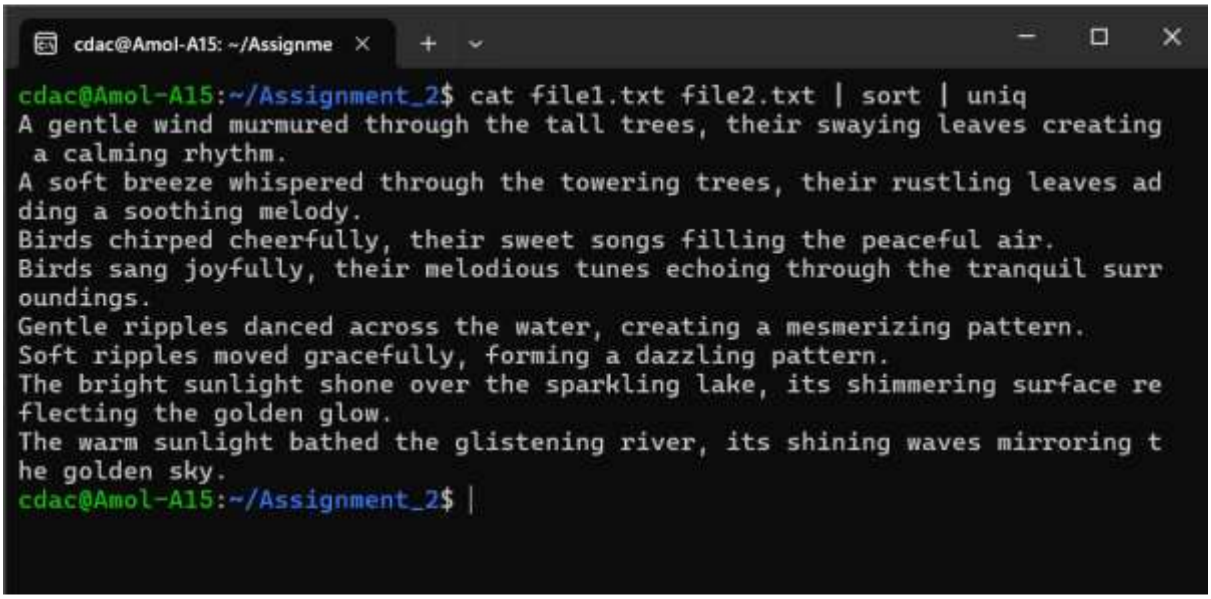
12. ls -l | grep ".txt"

    { | } this piping symbol is used to pass a program's output into another program's input.



```
cdac@Amol-A15:~/Assignment_2$ ls
file.txt  file2.txt  mydir  path  script.sh
cdac@Amol-A15:~/Assignment_2$ ls -l | grep ".txt"
-rw-r--r-- 1 cdac cdac  618 Feb 28 16:02 file.txt
-rw-r--r-- 1 cdac cdac   17 Feb 28 14:32 file2.txt
cdac@Amol-A15:~/Assignment_2$
```

13. cat file1.txt file2.txt | sort | uniq
    i) Print the contains of the both file
    ii) Then the output (contain of both files) is sort all lines according to the alphabetical order.
    iii) Then exclude all repeated lines.

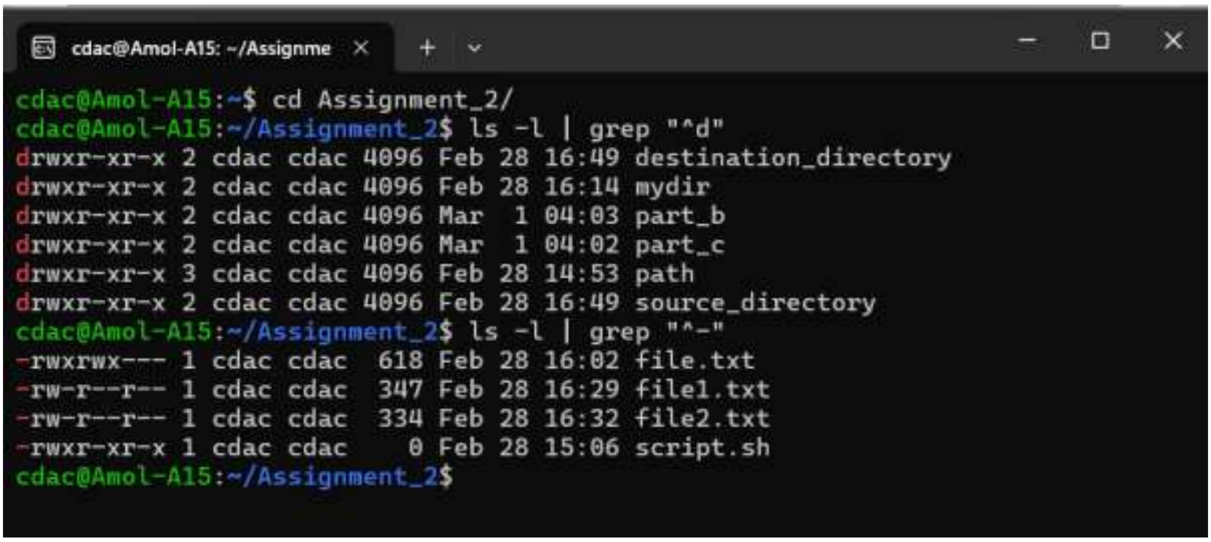


```
cdac@Amol-A15:~/Assignment_2$ cat file1.txt file2.txt | sort | uniq
A gentle wind murmured through the tall trees, their swaying leaves creating
 a calming rhythm.
A soft breeze whispered through the towering trees, their rustling leaves ad
ding a soothing melody.
Birds chirped cheerfully, their sweet songs filling the peaceful air.
Birds sang joyfully, their melodious tunes echoing through the tranquil surr
oundings.
Gentle ripples danced across the water, creating a mesmerizing pattern.
Soft ripples moved gracefully, forming a dazzling pattern.
The bright sunlight shone over the sparkling lake, its shimmering surface re
flecting the golden glow.
The warm sunlight bathed the glistening river, its shining waves mirroring t
he golden sky.
cdac@Amol-A15:~/Assignment_2$ |
```

14. ls -l | grep "^d"

ls -l is use to print the metadata of all directories as well as files present in the current directory.

With grep we can filter out the { “^d” } only directories. or files by using { “^-” }.

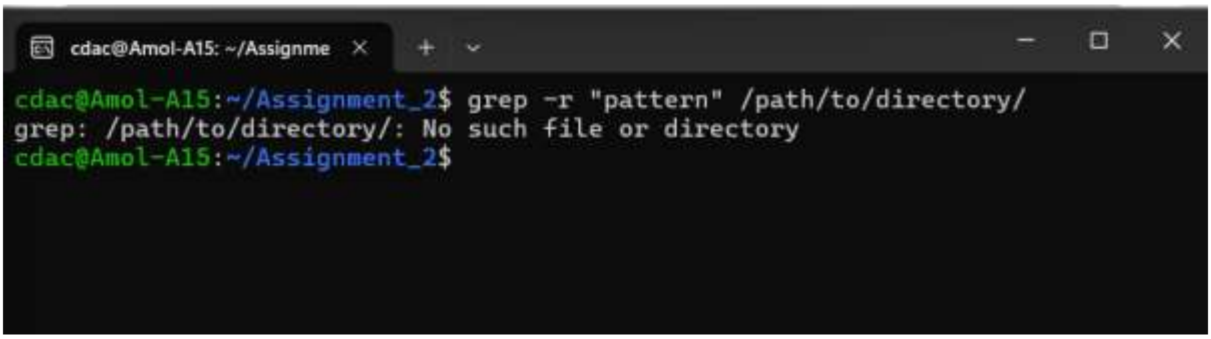


```
cdac@Amol-A15:~$ cd Assignment_2/
cdac@Amol-A15:~/Assignment_2$ ls -l | grep "^d"
drwxr-xr-x 2 cdac cdac 4096 Feb 28 16:49 destination_directory
drwxr-xr-x 2 cdac cdac 4096 Feb 28 16:14 mydir
drwxr-xr-x 2 cdac cdac 4096 Mar  1 04:03 part_b
drwxr-xr-x 2 cdac cdac 4096 Mar  1 04:02 part_c
drwxr-xr-x 3 cdac cdac 4096 Feb 28 14:53 path
drwxr-xr-x 2 cdac cdac 4096 Feb 28 16:49 source_directory
cdac@Amol-A15:~/Assignment_2$ ls -l | grep "^-"
-rwxrwx--- 1 cdac cdac  618 Feb 28 16:02 file.txt
-rw-r--r-- 1 cdac cdac  347 Feb 28 16:29 file1.txt
-rw-r--r-- 1 cdac cdac  334 Feb 28 16:32 file2.txt
-rwxr-xr-x 1 cdac cdac    0 Feb 28 15:06 script.sh
cdac@Amol-A15:~/Assignment_2$
```

15. grep -r "pattern" /path/to/directory/

Here grep command is used recursively to search for given pattern “pattern” in the given directory path.

But there is no such file present in our directory.

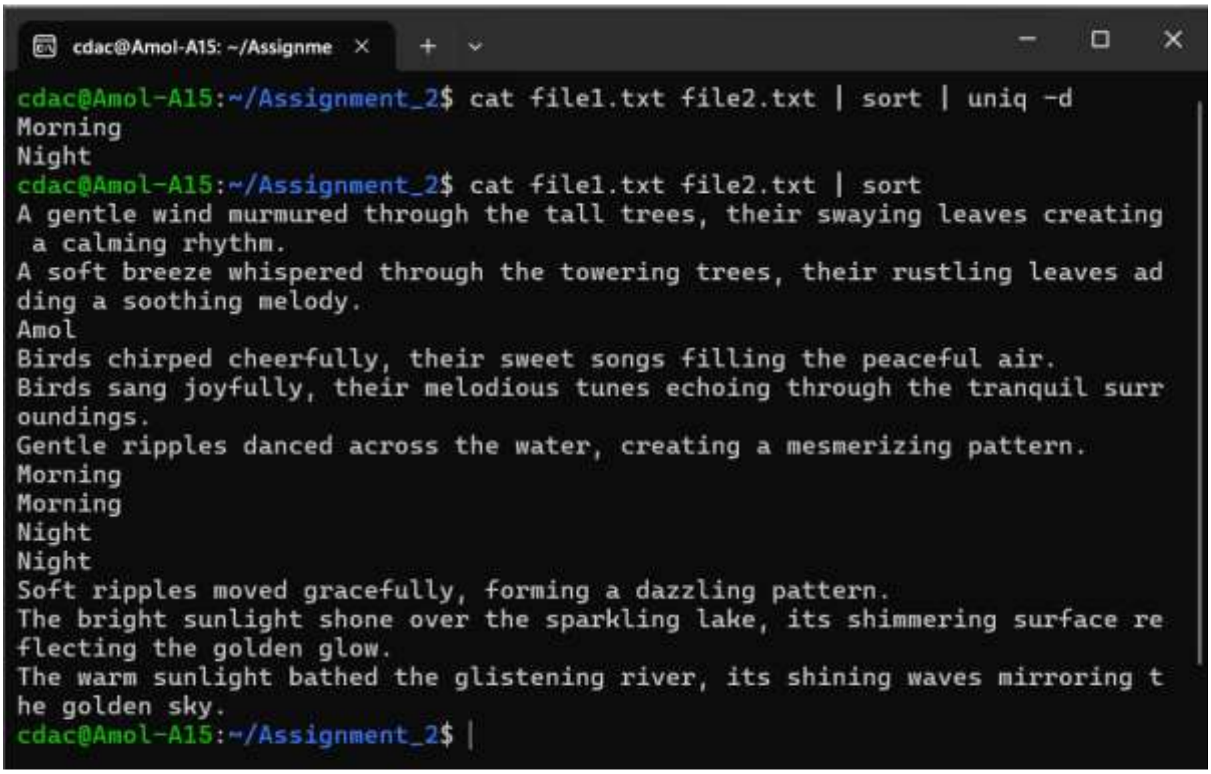


```
cdac@Amol-A15:~/Assignment_2$ grep -r "pattern" /path/to/directory/
grep: /path/to/directory/: No such file or directory
cdac@Amol-A15:~/Assignment_2$
```

16. cat file1.txt file2.txt | sort | uniq –d

we are combining the contains of the file1.txt and file2.txt then sorting them alphabetically.

Then by using uniq -d command we are printing only duplicate values.



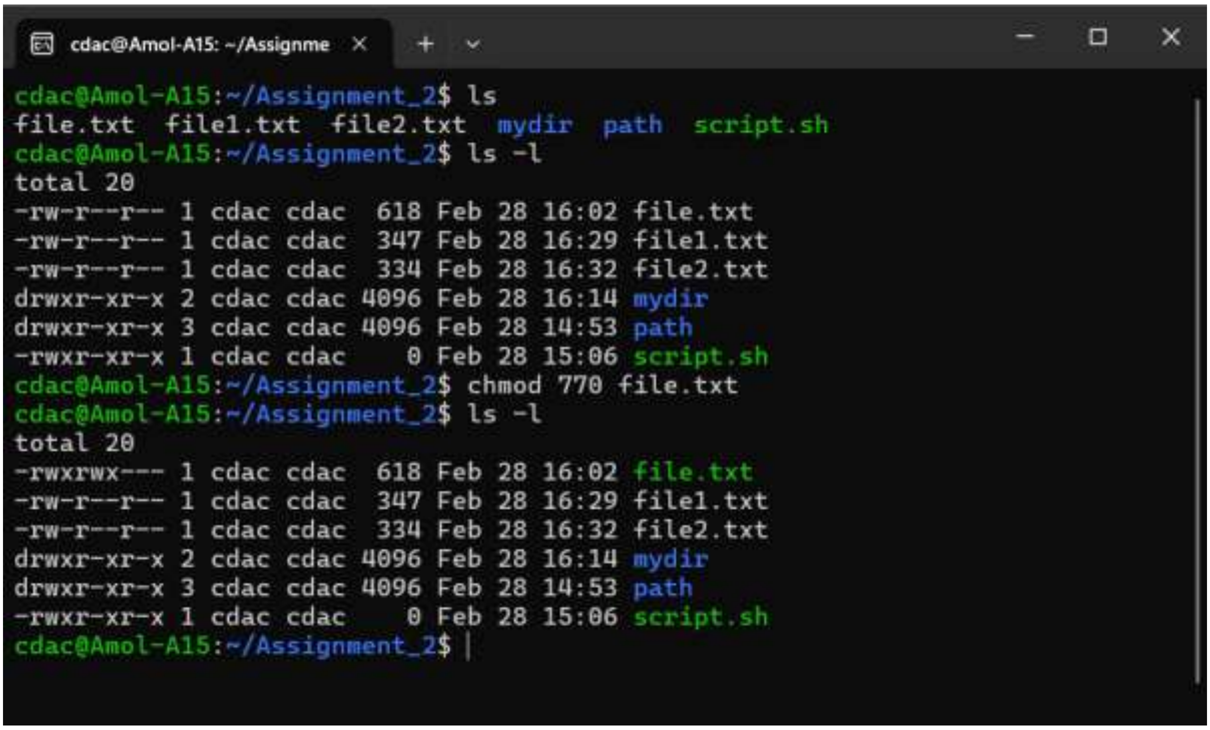
```
cdac@Amol-A15:~/Assignment_2$ cat file1.txt file2.txt | sort | uniq -d
Morning
Night
cdac@Amol-A15:~/Assignment_2$ cat file1.txt file2.txt | sort
A gentle wind murmured through the tall trees, their swaying leaves creating
 a calming rhythm.
A soft breeze whispered through the towering trees, their rustling leaves ad
ding a soothing melody.
Amol
Birds chirped cheerfully, their sweet songs filling the peaceful air.
Birds sang joyfully, their melodious tunes echoing through the tranquil surr
oundings.
Gentle ripples danced across the water, creating a mesmerizing pattern.
Morning
Morning
Night
Night
Soft ripples moved gracefully, forming a dazzling pattern.
The bright sunlight shone over the sparkling lake, its shimmering surface re
flecting the golden glow.
The warm sunlight bathed the glistening river, its shining waves mirroring t
he golden sky.
cdac@Amol-A15:~/Assignment_2$ |
```

17. chmod 644 file.txt

    change file permissions

    Default file permissions are: u=rw, g=r, o=r { 644}

    Change it to: { 770 } means u=rwx, g=rwx, o=---



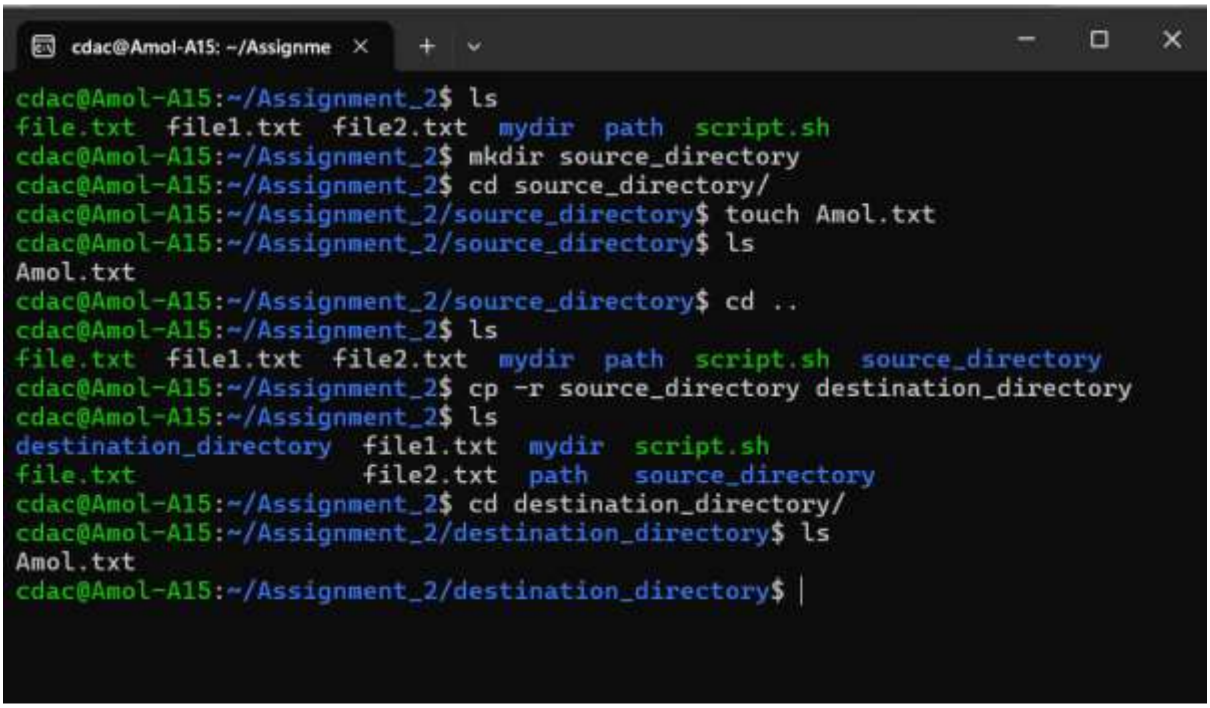
```
cdac@Amol-A15:~/Assignment_2$ ls
file.txt  file1.txt  file2.txt  mydir  path  script.sh
cdac@Amol-A15:~/Assignment_2$ ls -l
total 20
-rw-r--r-- 1 cdac cdac  618 Feb 28 16:02 file.txt
-rw-r--r-- 1 cdac cdac  347 Feb 28 16:29 file1.txt
-rw-r--r-- 1 cdac cdac  334 Feb 28 16:32 file2.txt
drwxr-xr-x 2 cdac cdac 4096 Feb 28 16:14 mydir
drwxr-xr-x 3 cdac cdac 4096 Feb 28 14:53 path
-rwxr-xr-x 1 cdac cdac    0 Feb 28 15:06 script.sh
cdac@Amol-A15:~/Assignment_2$ chmod 770 file.txt
cdac@Amol-A15:~/Assignment_2$ ls -l
total 20
-rwxrwx--- 1 cdac cdac  618 Feb 28 16:02 file.txt
-rw-r--r-- 1 cdac cdac  347 Feb 28 16:29 file1.txt
-rw-r--r-- 1 cdac cdac  334 Feb 28 16:32 file2.txt
drwxr-xr-x 2 cdac cdac 4096 Feb 28 16:14 mydir
drwxr-xr-x 3 cdac cdac 4096 Feb 28 14:53 path
-rwxr-xr-x 1 cdac cdac    0 Feb 28 15:06 script.sh
cdac@Amol-A15:~/Assignment_2$ |
```

18. cp -r source_directory destination_directory

    copy files and directories

    -r is use to copy directories recursively

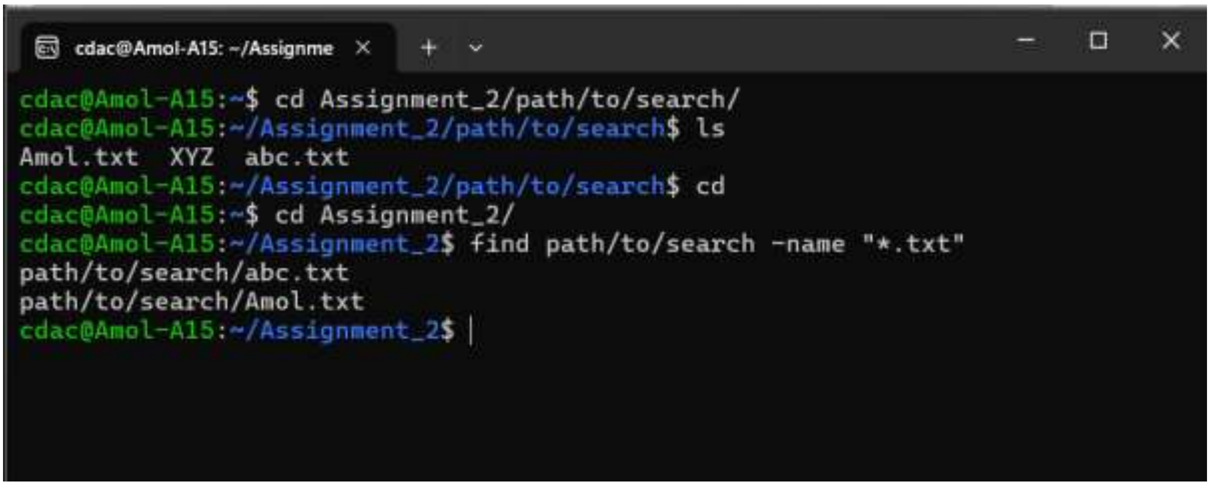
    Here recursively means all the contains of the source_directory are copied to destination_directory.



```
cdac@Amol-A15:~/Assignment_2$ ls
file.txt  file1.txt  file2.txt  mydir  path  script.sh
cdac@Amol-A15:~/Assignment_2$ mkdir source_directory
cdac@Amol-A15:~/Assignment_2$ cd source_directory/
cdac@Amol-A15:~/Assignment_2/source_directory$ touch Amol.txt
cdac@Amol-A15:~/Assignment_2/source_directory$ ls
Amol.txt
cdac@Amol-A15:~/Assignment_2/source_directory$ cd ..
cdac@Amol-A15:~/Assignment_2$ ls
file.txt  file1.txt  file2.txt  mydir  path  script.sh  source_directory
cdac@Amol-A15:~/Assignment_2$ cp -r source_directory destination_directory
cdac@Amol-A15:~/Assignment_2$ ls
destination_directory  file1.txt  mydir  script.sh
file.txt               file2.txt  path   source_directory
cdac@Amol-A15:~/Assignment_2$ cd destination_directory/
cdac@Amol-A15:~/Assignment_2/destination_directory$ ls
Amol.txt
cdac@Amol-A15:~/Assignment_2/destination_directory$ |
```

19. find /path/to/search -name "*.txt"

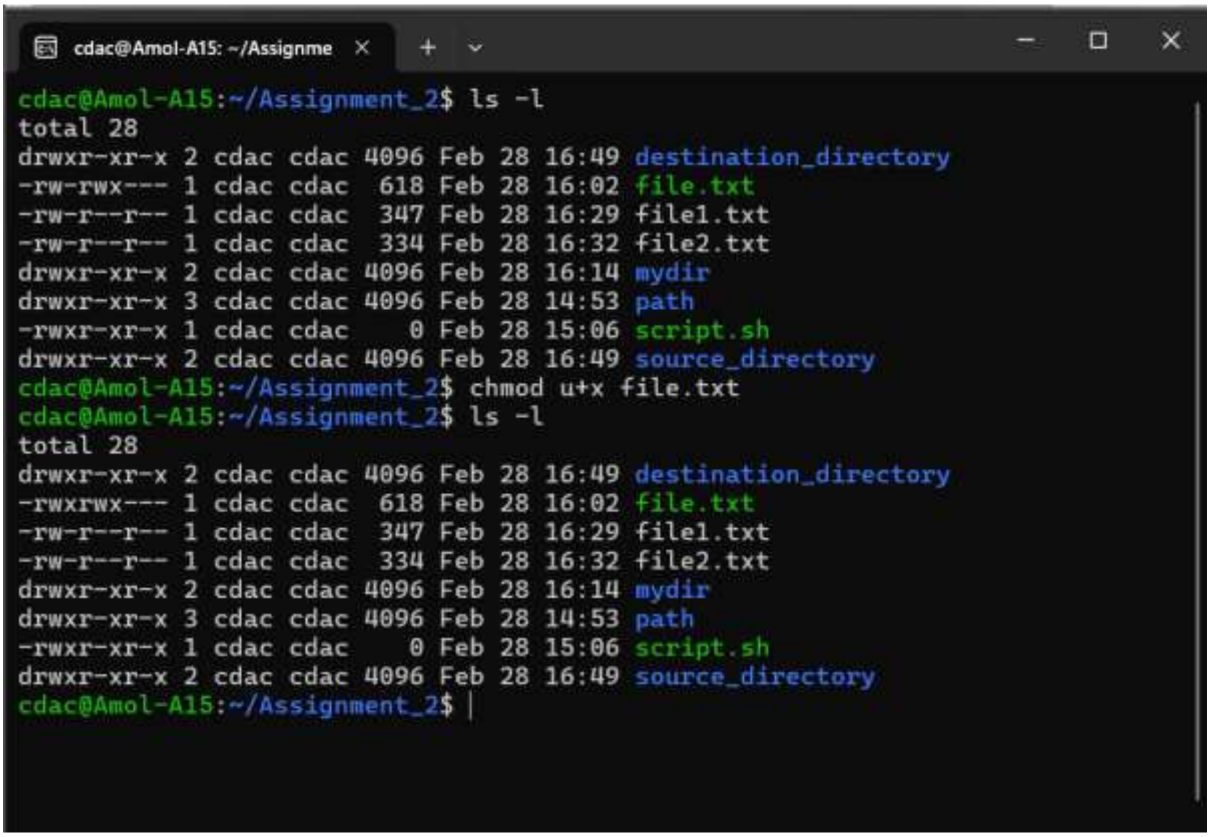
    use to find the { .txt } file name in the given directory path.



```
cdac@Amol-A15:~$ cd Assignment_2/path/to/search/
cdac@Amol-A15:~/Assignment_2/path/to/search$ ls
Amol.txt  XYZ  abc.txt
cdac@Amol-A15:~/Assignment_2/path/to/search$ cd
cdac@Amol-A15:~$ cd Assignment_2/
cdac@Amol-A15:~/Assignment_2$ find path/to/search -name "*.txt"
path/to/search/abc.txt
path/to/search/Amol.txt
cdac@Amol-A15:~/Assignment_2$ |
```

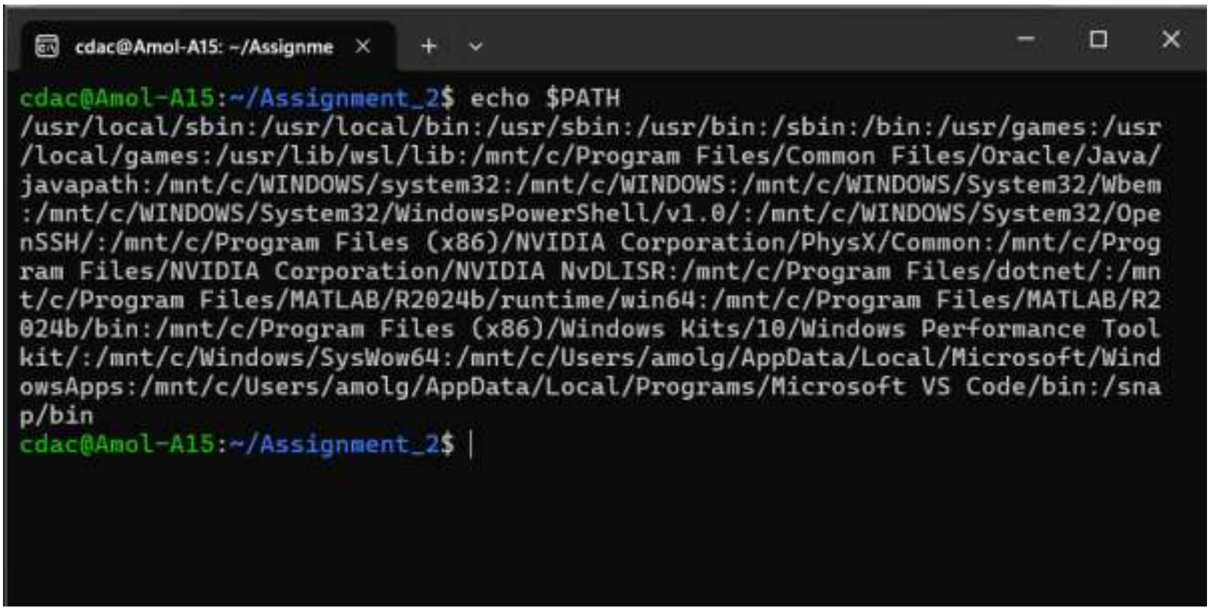
20. chmod u+x file.txt

    Initially user didn’t have the permission to execute the file, but by using the{ u+x } we give permission to execute the file.txt to user.



```
cdac@Amol-A15:~/Assignment_2$ ls -l
total 28
drwxr-xr-x 2 cdac cdac 4096 Feb 28 16:49 destination_directory
-rw-rwx--- 1 cdac cdac  618 Feb 28 16:02 file.txt
-rw-r--r-- 1 cdac cdac  347 Feb 28 16:29 file1.txt
-rw-r--r-- 1 cdac cdac  334 Feb 28 16:32 file2.txt
drwxr-xr-x 2 cdac cdac 4096 Feb 28 16:14 mydir
drwxr-xr-x 3 cdac cdac 4096 Feb 28 14:53 path
-rwxr-xr-x 1 cdac cdac    0 Feb 28 15:06 script.sh
drwxr-xr-x 2 cdac cdac 4096 Feb 28 16:49 source_directory
cdac@Amol-A15:~/Assignment_2$ chmod u+x file.txt
cdac@Amol-A15:~/Assignment_2$ ls -l
total 28
drwxr-xr-x 2 cdac cdac 4096 Feb 28 16:49 destination_directory
-rwxrwx--- 1 cdac cdac  618 Feb 28 16:02 file.txt
-rw-r--r-- 1 cdac cdac  347 Feb 28 16:29 file1.txt
-rw-r--r-- 1 cdac cdac  334 Feb 28 16:32 file2.txt
drwxr-xr-x 2 cdac cdac 4096 Feb 28 16:14 mydir
drwxr-xr-x 3 cdac cdac 4096 Feb 28 14:53 path
-rwxr-xr-x 1 cdac cdac    0 Feb 28 15:06 script.sh
drwxr-xr-x 2 cdac cdac 4096 Feb 28 16:49 source_directory
cdac@Amol-A15:~/Assignment_2$ |
```

21. echo $PATH

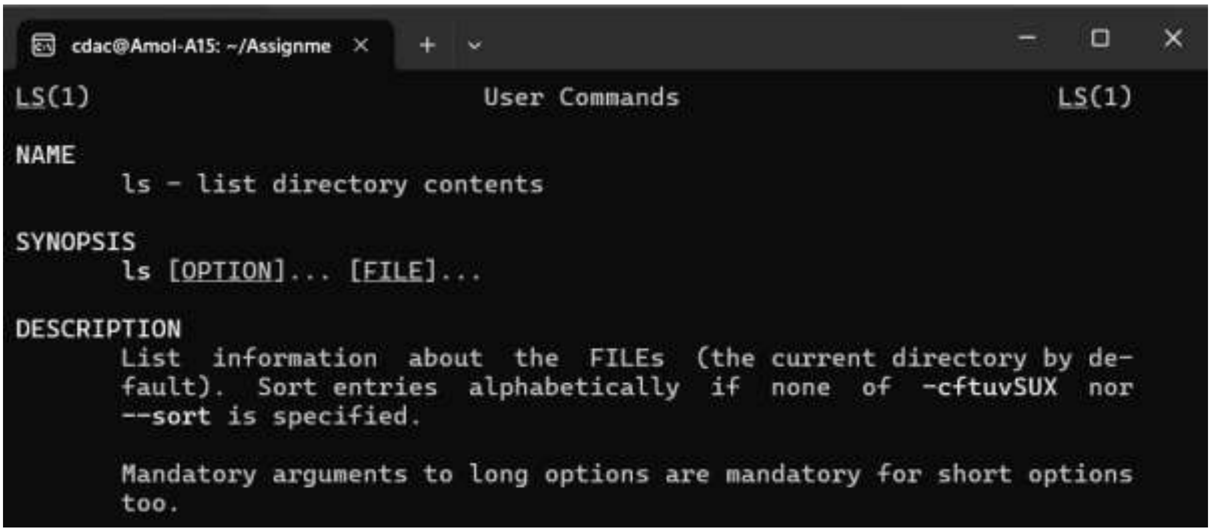


```
cdac@Amol-A15:~/Assignment_2$ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr
/local/games:/usr/lib/wsl/lib:/mnt/c/Program Files/Common Files/Oracle/Java/
javapath:/mnt/c/WINDOWS/system32:/mnt/c/WINDOWS:/mnt/c/WINDOWS/System32/Wbem
:/mnt/c/WINDOWS/System32/WindowsPowerShell/v1.0/:/mnt/c/WINDOWS/System32/Ope
nSSH/:/mnt/c/Program Files (x86)/NVIDIA Corporation/PhysX/Common:/mnt/c/Prog
ram Files/NVIDIA Corporation/NVIDIA NvDLISR:/mnt/c/Program Files/dotnet/:/mn
t/c/Program Files/MATLAB/R2024b/runtime/win64:/mnt/c/Program Files/MATLAB/R2
024b/bin:/mnt/c/Program Files (x86)/Windows Kits/10/Windows Performance Tool
kit/:/mnt/c/Windows/SysWow64:/mnt/c/Users/amolg/AppData/Local/Microsoft/Wind
owsApps:/mnt/c/Users/amolg/AppData/Local/Programs/Microsoft VS Code/bin:/sna
p/bin
cdac@Amol-A15:~/Assignment_2$ |
```

Part B

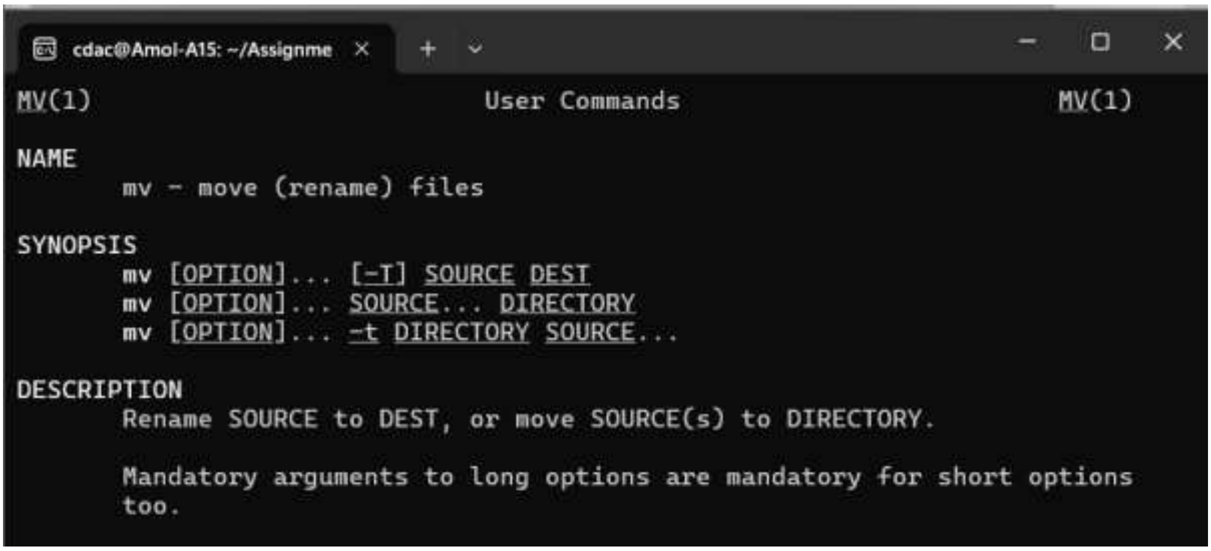
Identify True or False:

1. ls is used to list files and directories in a directory.
   This statement is **true**, Check the manual of ls.



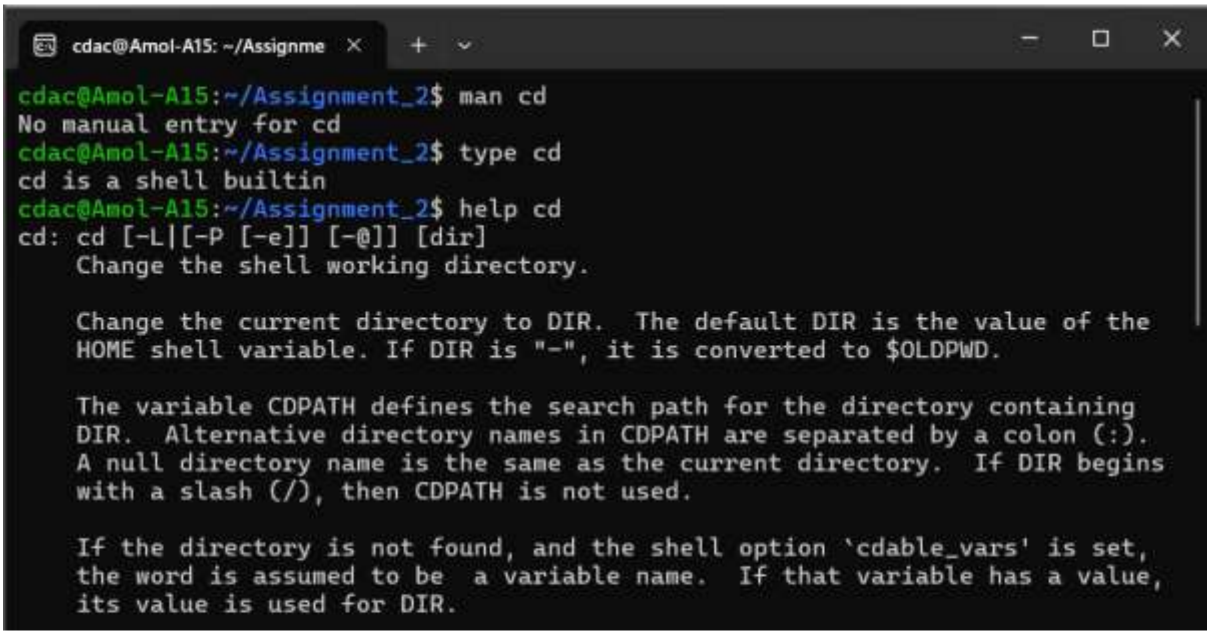
```
LS(1)                        User Commands                        LS(1)

NAME
       ls - list directory contents

SYNOPSIS
       ls [OPTION]... [FILE]...

DESCRIPTION
       List  information  about  the  FILEs  (the current directory by de-
       fault).  Sort entries  alphabetically  if  none  of  -cftuvSUX  nor
       --sort is specified.

       Mandatory arguments to long options are mandatory for short options
       too.
```

2. mv is used to move files and directories.
   This statement is **true**, Check the manual of mv.
   Also, this command is use to rename files.



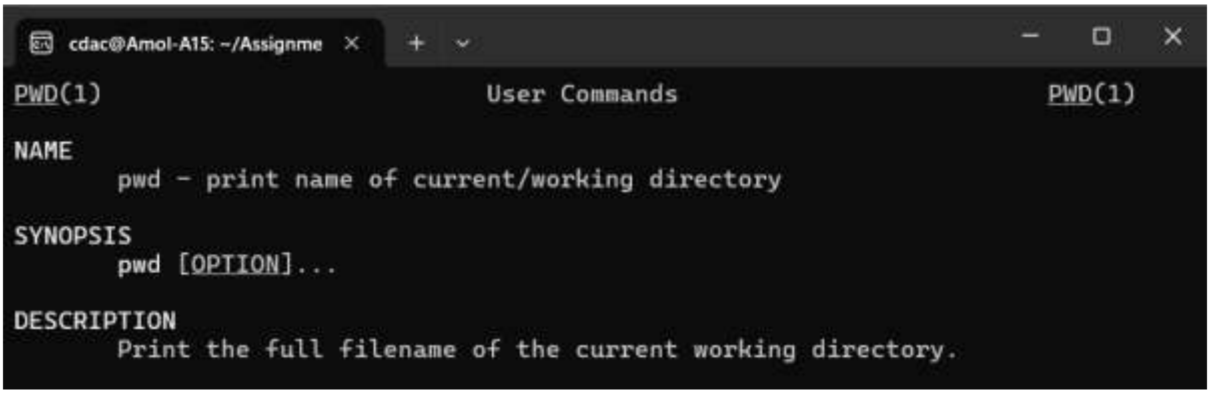
```
MV(1)                        User Commands                        MV(1)

NAME
       mv - move (rename) files

SYNOPSIS
       mv [OPTION]... [-T] SOURCE DEST
       mv [OPTION]... SOURCE... DIRECTORY
       mv [OPTION]... -t DIRECTORY SOURCE...

DESCRIPTION
       Rename SOURCE to DEST, or move SOURCE(s) to DIRECTORY.

       Mandatory arguments to long options are mandatory for short options
       too.
```

3. cd is used to copy files and directories.
   This statement is **false**, cd command is use to change the directory.
   {cd is builtin shell command therefor we can't call it's manual to need to use help for the manual to print}



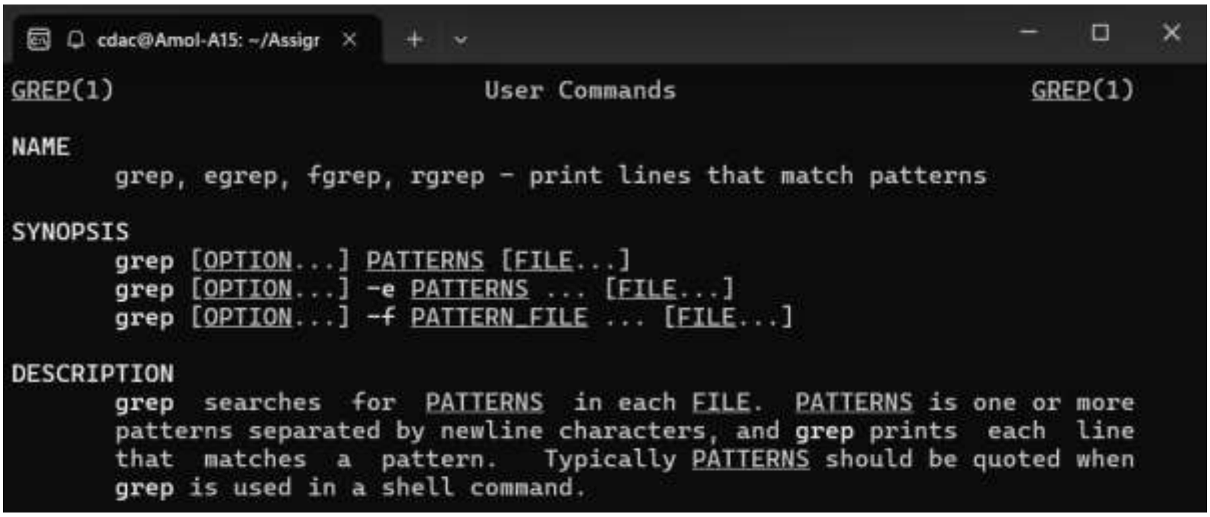
```
cdac@Amol-A15:~/Assignment_2$ man cd
No manual entry for cd
cdac@Amol-A15:~/Assignment_2$ type cd
cd is a shell builtin
cdac@Amol-A15:~/Assignment_2$ help cd
cd: cd [-L|[-P [-e]] [-@]] [dir]
    Change the shell working directory.

    Change the current directory to DIR.  The default DIR is the value of the
    HOME shell variable. If DIR is "-", it is converted to $OLDPWD.

    The variable CDPATH defines the search path for the directory containing
    DIR.  Alternative directory names in CDPATH are separated by a colon (:).
    A null directory name is the same as the current directory.  If DIR begins
    with a slash (/), then CDPATH is not used.

    If the directory is not found, and the shell option `cdable_vars' is set,
    the word is assumed to be  a variable name.  If that variable has a value,
    its value is used for DIR.
```

4. pwd stands for "print working directory" and displays the current directory
   This statement is **true**, Check the manual of pwd.



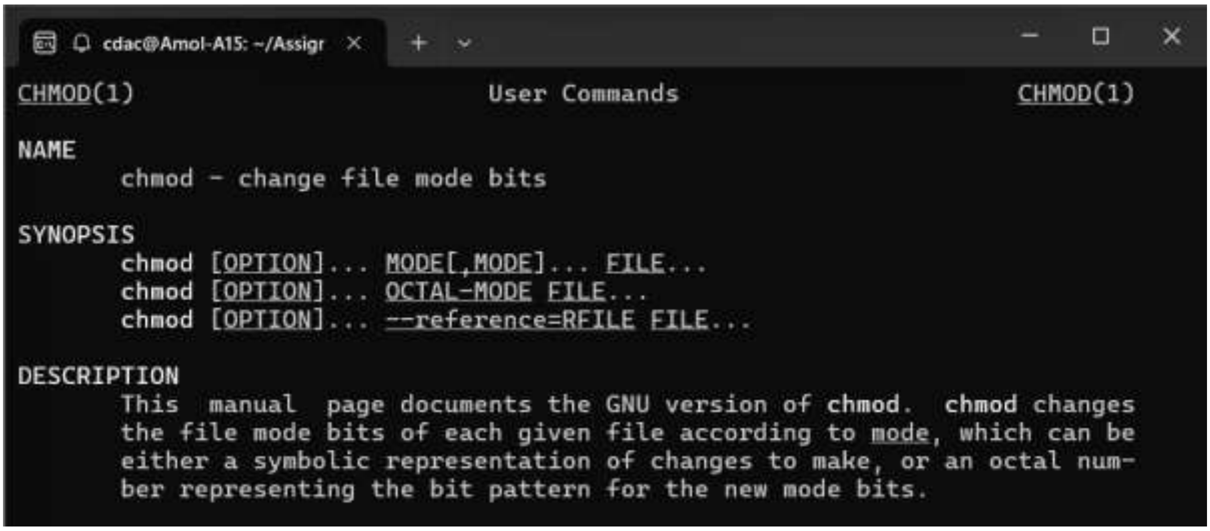
```
PWD(1)                          User Commands                          PWD(1)

NAME
       pwd - print name of current/working directory

SYNOPSIS
       pwd [OPTION]...

DESCRIPTION
       Print the full filename of the current working directory.
```

5. grep is used to search for patterns in files.
   This statement is **true**, Check the manual of grep.



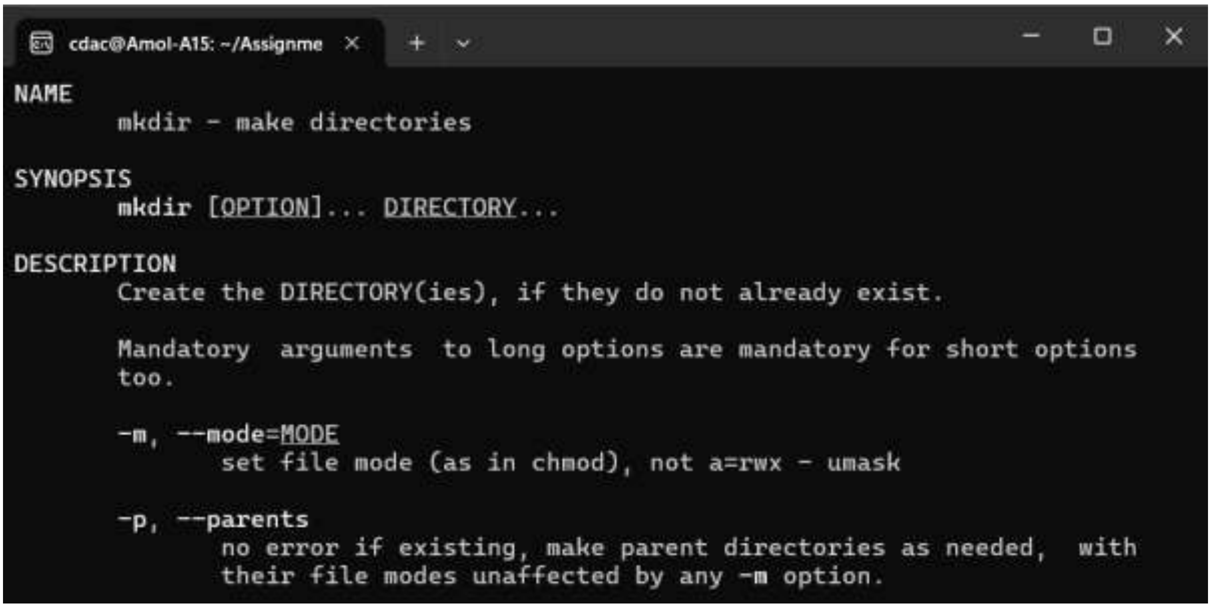
```
GREP(1)                         User Commands                         GREP(1)

NAME
       grep, egrep, fgrep, rgrep - print lines that match patterns

SYNOPSIS
       grep [OPTION...] PATTERNS [FILE...]
       grep [OPTION...] -e PATTERNS ... [FILE...]
       grep [OPTION...] -f PATTERN_FILE ... [FILE...]

DESCRIPTION
       grep  searches  for  PATTERNS  in each FILE.  PATTERNS is one or more
       patterns separated by newline characters, and grep prints  each  line
       that  matches  a  pattern.   Typically PATTERNS should be quoted when
       grep is used in a shell command.
```

6. chmod 755 file.txt gives read, write, and execute permissions to the owner, and read and execute permissions to group and others.
   This statement is **true**, Check the manual of chmod.



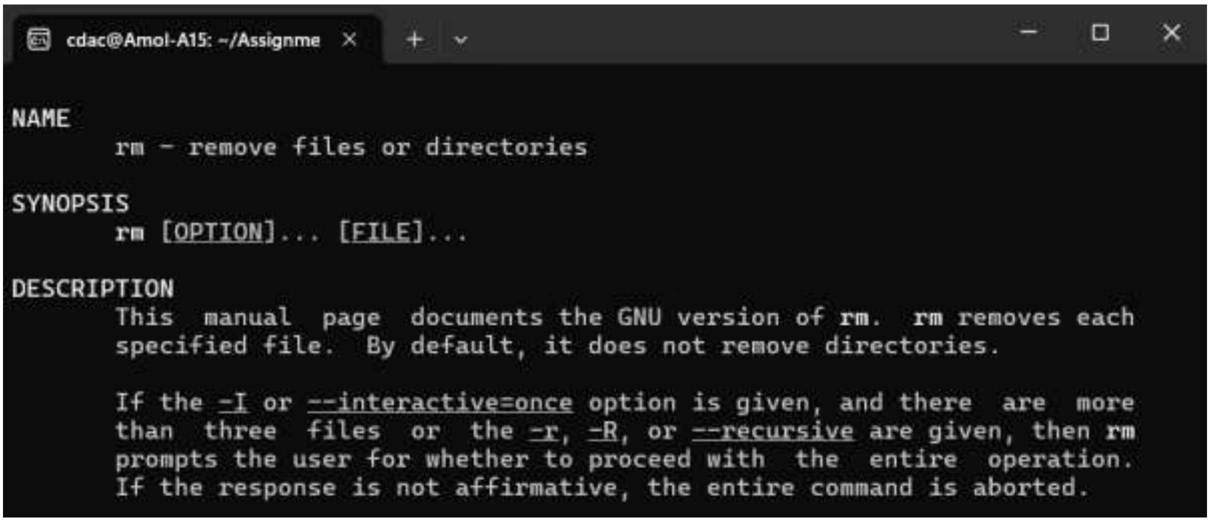
```
CHMOD(1)                    User Commands                    CHMOD(1)

NAME
       chmod - change file mode bits

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

DESCRIPTION
       This  manual  page documents the GNU version of chmod.  chmod changes
       the file mode bits of each given file according to mode, which can be
       either a symbolic representation of changes to make, or an octal num-
       ber representing the bit pattern for the new mode bits.
```

7. mkdir -p directory1/directory2 creates nested directories, creating directory2 inside directory1 if directory1 does not exist.
   This statement is **true**, Check the manual of mkdir.



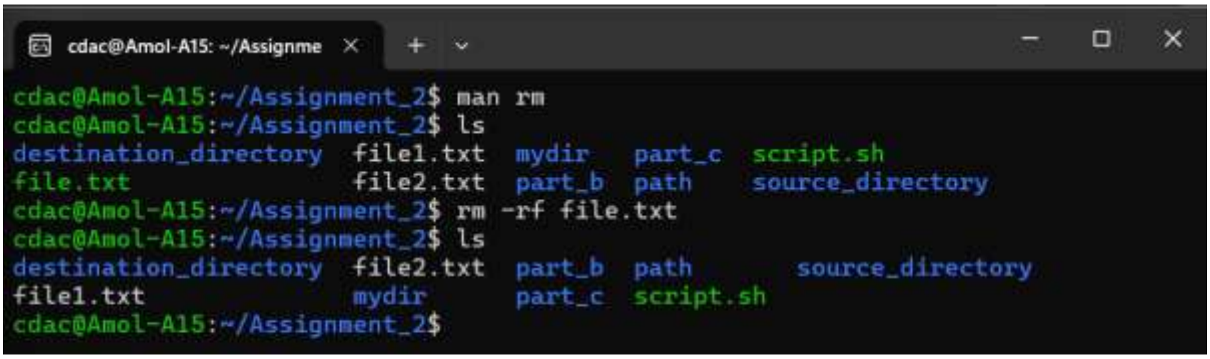
```
NAME
       mkdir - make directories

SYNOPSIS
       mkdir [OPTION]... DIRECTORY...

DESCRIPTION
       Create the DIRECTORY(ies), if they do not already exist.

       Mandatory  arguments  to long options are mandatory for short options
       too.

       -m, --mode=MODE
              set file mode (as in chmod), not a=rwx - umask

       -p, --parents
              no error if existing, make parent directories as needed,  with
              their file modes unaffected by any -m option.
```

8. rm -rf file.txt deletes a file forcefully without confirmation.
   This statement is **true**, Check the manual of rm.



```
NAME
       rm - remove files or directories

SYNOPSIS
       rm [OPTION]... [FILE]...

DESCRIPTION
       This  manual  page  documents the GNU version of rm.  rm removes each
       specified file.  By default, it does not remove directories.

       If the -I or --interactive=once option is given, and there  are  more
       than  three  files  or  the -r, -R, or --recursive are given, then rm
       prompts the user for whether to proceed with  the  entire  operation.
       If the response is not affirmative, the entire command is aborted.
```

Practical Implementation



```
cdac@Amol-A15:~/Assignment_2$ man rm
cdac@Amol-A15:~/Assignment_2$ ls
destination_directory  file1.txt  mydir   part_c  script.sh
file.txt               file2.txt  part_b  path    source_directory
cdac@Amol-A15:~/Assignment_2$ rm -rf file.txt
cdac@Amol-A15:~/Assignment_2$ ls
destination_directory  file2.txt  part_b  path       source_directory
file1.txt              mydir      part_c  script.sh
cdac@Amol-A15:~/Assignment_2$
```

Identify the Incorrect Commands:

1. chmodx is used to change file permissions.
   { chmod } this command is used to change file permissions.

2. cpy is used to copy files and directories.
   { cp } this command is used to copy files and directories.

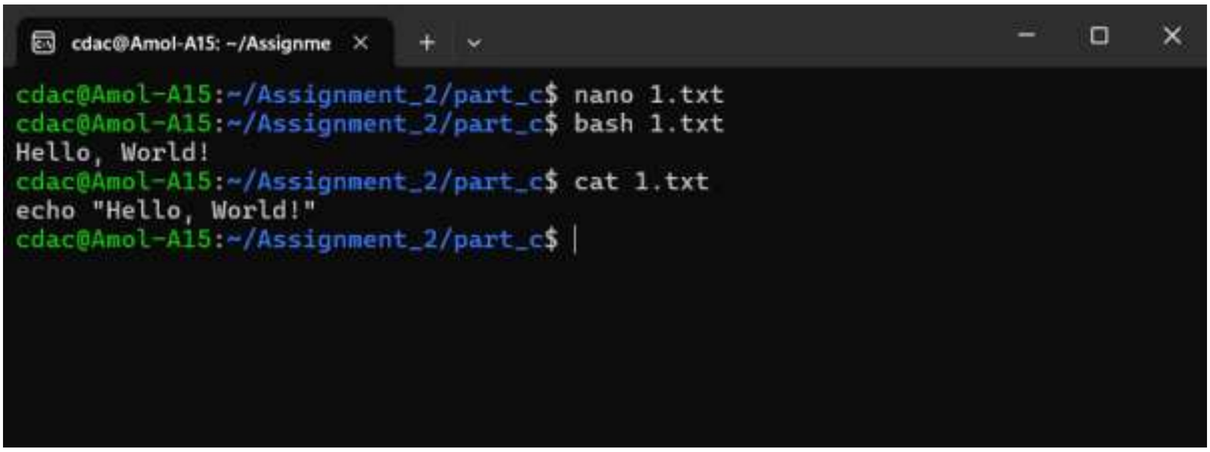
3. mkfile is used to create a new file.
   { touch } command is used to create new files.
   { mkdir } command is used to create new directories.

4. catx is used to concatenate files.
   { cat } command is used to concatenate files.

5. rn is used to rename files
   { mv } command is used to rename files.

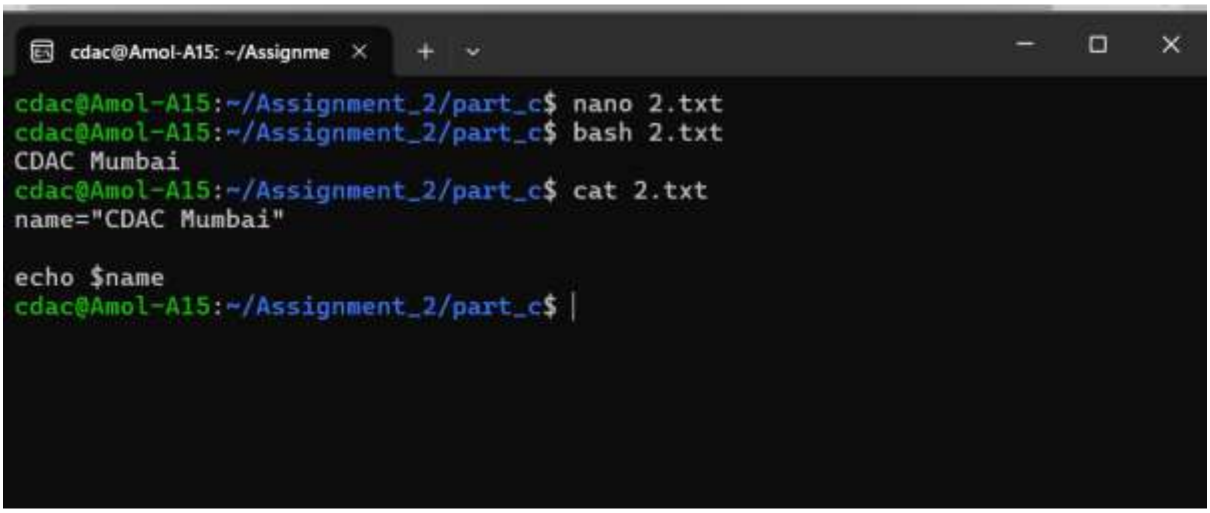
Part C

1) Write a shell script that prints "Hello, World!" to the terminal.



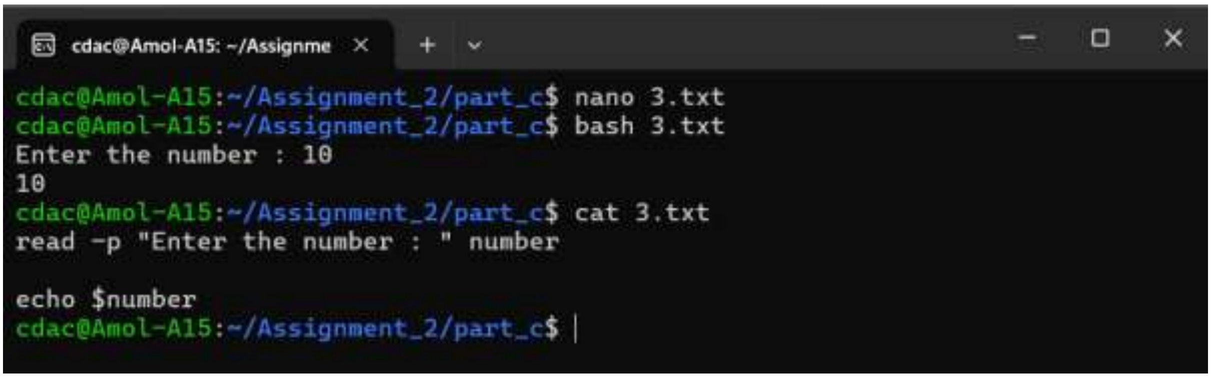
```
cdac@Amol-A15:~/Assignment_2/part_c$ nano 1.txt
cdac@Amol-A15:~/Assignment_2/part_c$ bash 1.txt
Hello, World!
cdac@Amol-A15:~/Assignment_2/part_c$ cat 1.txt
echo "Hello, World!"
cdac@Amol-A15:~/Assignment_2/part_c$ |
```

2) Declare a variable named "name" and assign the value "CDAC Mumbai" to it. Print the value of the variable.



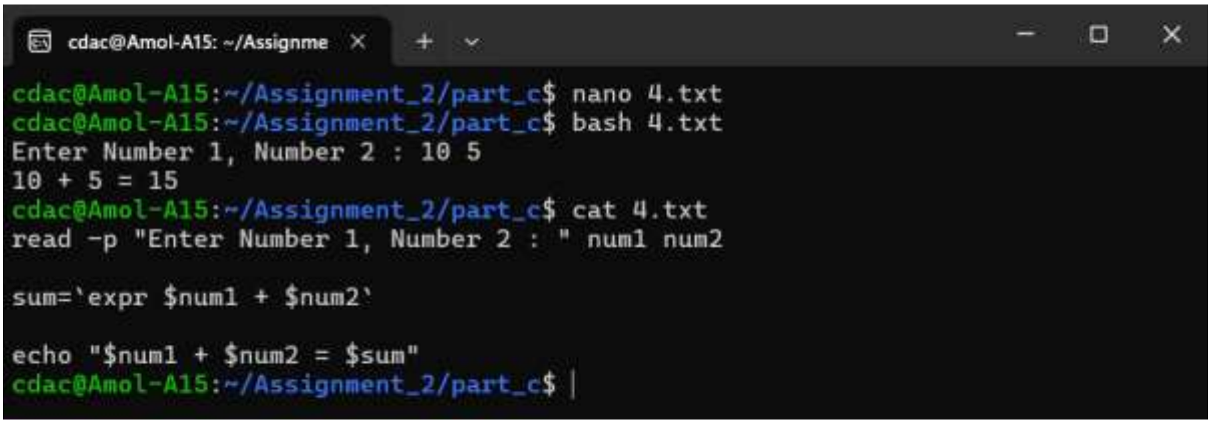
```
cdac@Amol-A15:~/Assignment_2/part_c$ nano 2.txt
cdac@Amol-A15:~/Assignment_2/part_c$ bash 2.txt
CDAC Mumbai
cdac@Amol-A15:~/Assignment_2/part_c$ cat 2.txt
name="CDAC Mumbai"

echo $name
cdac@Amol-A15:~/Assignment_2/part_c$ |
```

3) Write a shell script that takes a number as input from the user and prints it.



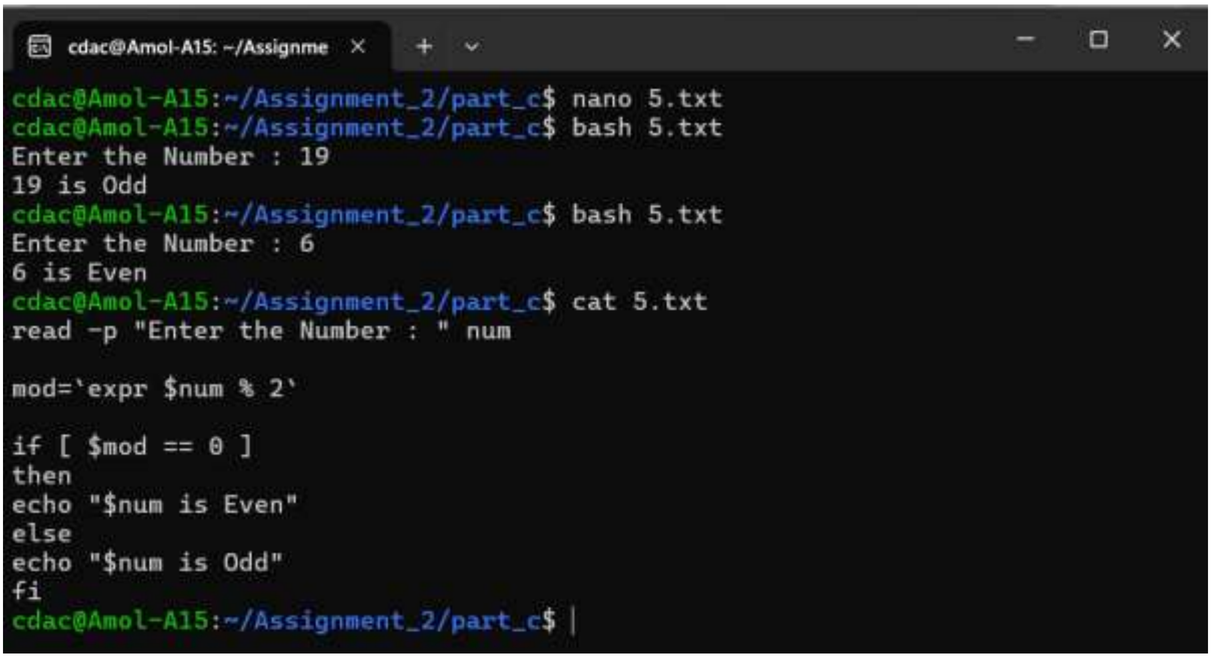
```
cdac@Amol-A15:~/Assignment_2/part_c$ nano 3.txt
cdac@Amol-A15:~/Assignment_2/part_c$ bash 3.txt
Enter the number : 10
10
cdac@Amol-A15:~/Assignment_2/part_c$ cat 3.txt
read -p "Enter the number : " number

echo $number
cdac@Amol-A15:~/Assignment_2/part_c$ |
```

4) Write a shell script that performs addition of two numbers (e.g., 5 and 3) and prints the result.



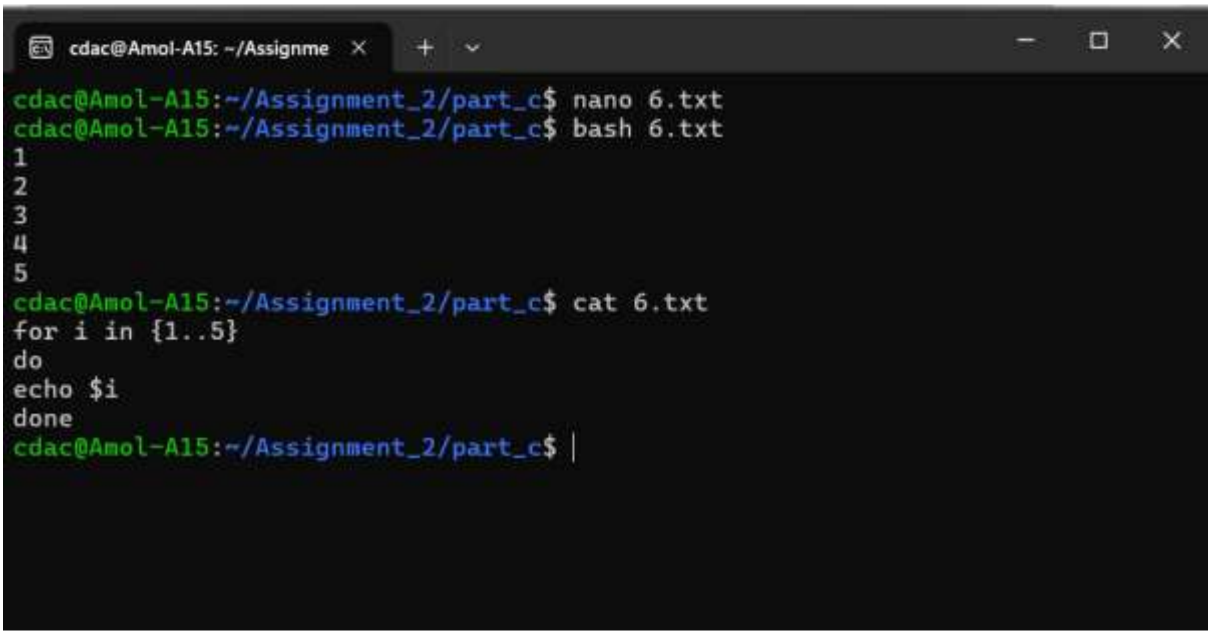
```
cdac@Amol-A15:~/Assignment_2/part_c$ nano 4.txt
cdac@Amol-A15:~/Assignment_2/part_c$ bash 4.txt
Enter Number 1, Number 2 : 10 5
10 + 5 = 15
cdac@Amol-A15:~/Assignment_2/part_c$ cat 4.txt
read -p "Enter Number 1, Number 2 : " num1 num2

sum=`expr $num1 + $num2`

echo "$num1 + $num2 = $sum"
cdac@Amol-A15:~/Assignment_2/part_c$ |
```

5) Write a shell script that takes a number as input and prints "Even" if it is even, otherwise prints "Odd".



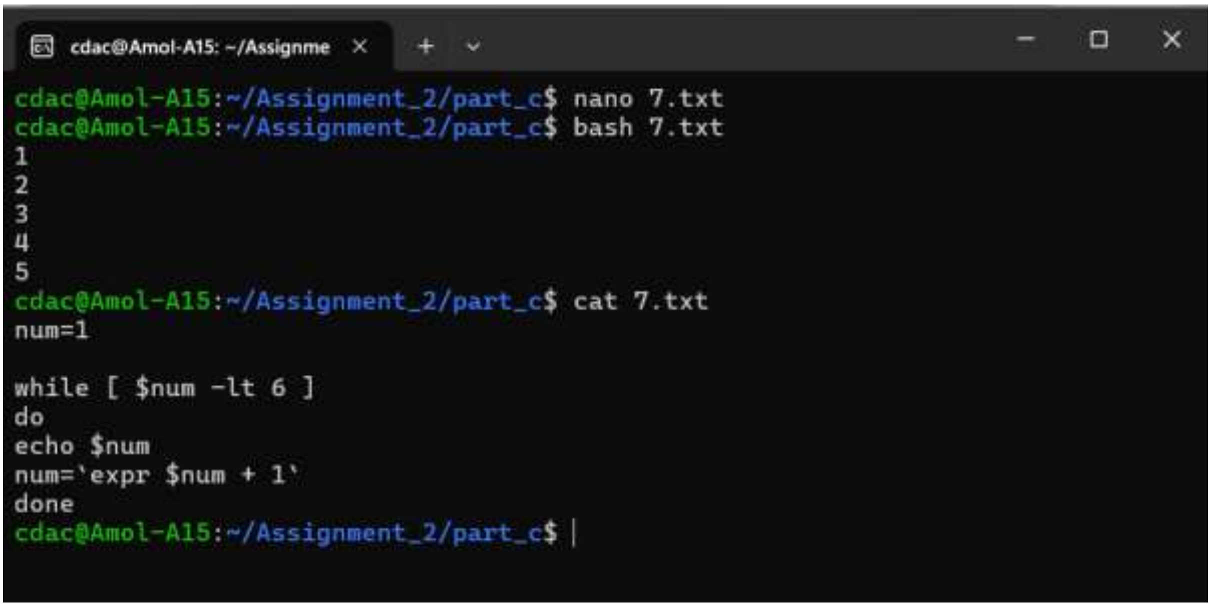
```
cdac@Amol-A15:~/Assignment_2/part_c$ nano 5.txt
cdac@Amol-A15:~/Assignment_2/part_c$ bash 5.txt
Enter the Number : 19
19 is Odd
cdac@Amol-A15:~/Assignment_2/part_c$ bash 5.txt
Enter the Number : 6
6 is Even
cdac@Amol-A15:~/Assignment_2/part_c$ cat 5.txt
read -p "Enter the Number : " num

mod=`expr $num % 2`

if [ $mod == 0 ]
then
echo "$num is Even"
else
echo "$num is Odd"
fi
cdac@Amol-A15:~/Assignment_2/part_c$ |
```

6) Write a shell script that uses a for loop to print numbers from 1 to 5.



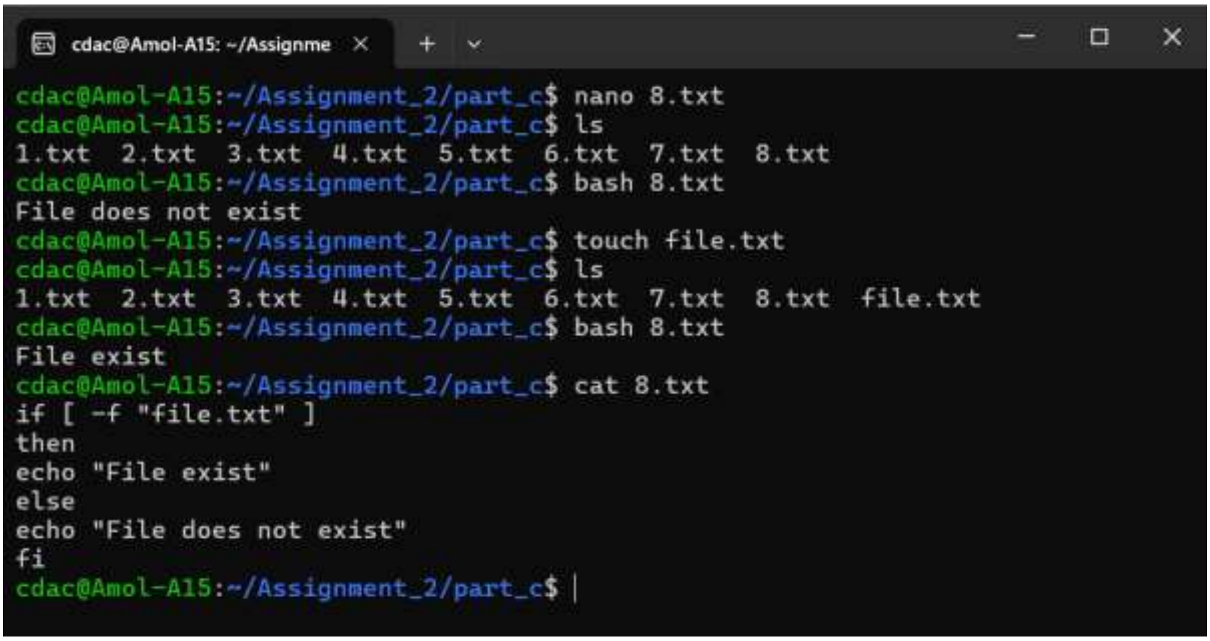
```
cdac@Amol-A15:~/Assignment_2/part_c$ nano 6.txt
cdac@Amol-A15:~/Assignment_2/part_c$ bash 6.txt
1
2
3
4
5
cdac@Amol-A15:~/Assignment_2/part_c$ cat 6.txt
for i in {1..5}
do
echo $i
done
cdac@Amol-A15:~/Assignment_2/part_c$ |
```

7) Write a shell script that uses a while loop to print numbers from 1 to 5.



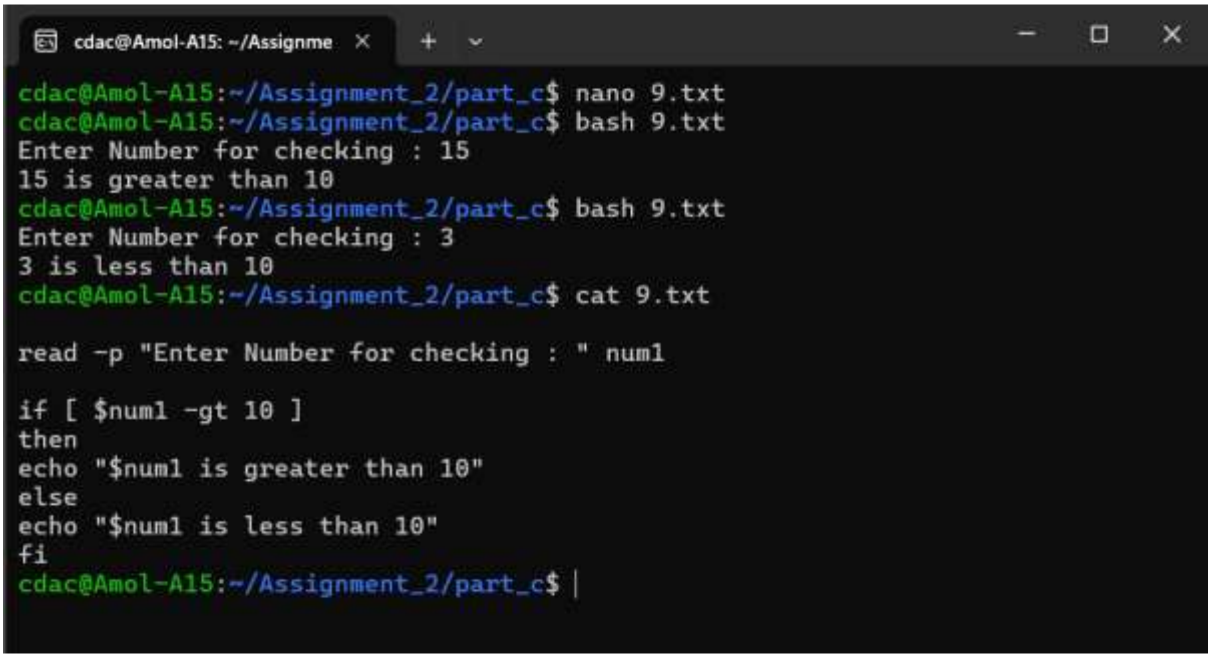
```
cdac@Amol-A15:~/Assignment_2/part_c$ nano 7.txt
cdac@Amol-A15:~/Assignment_2/part_c$ bash 7.txt
1
2
3
4
5
cdac@Amol-A15:~/Assignment_2/part_c$ cat 7.txt
num=1

while [ $num -lt 6 ]
do
echo $num
num=`expr $num + 1`
done
cdac@Amol-A15:~/Assignment_2/part_c$ |
```

8) Write a shell script that checks if a file named "file.txt" exists in the current directory. If it does, print "File exists", otherwise, print "File does not exist".



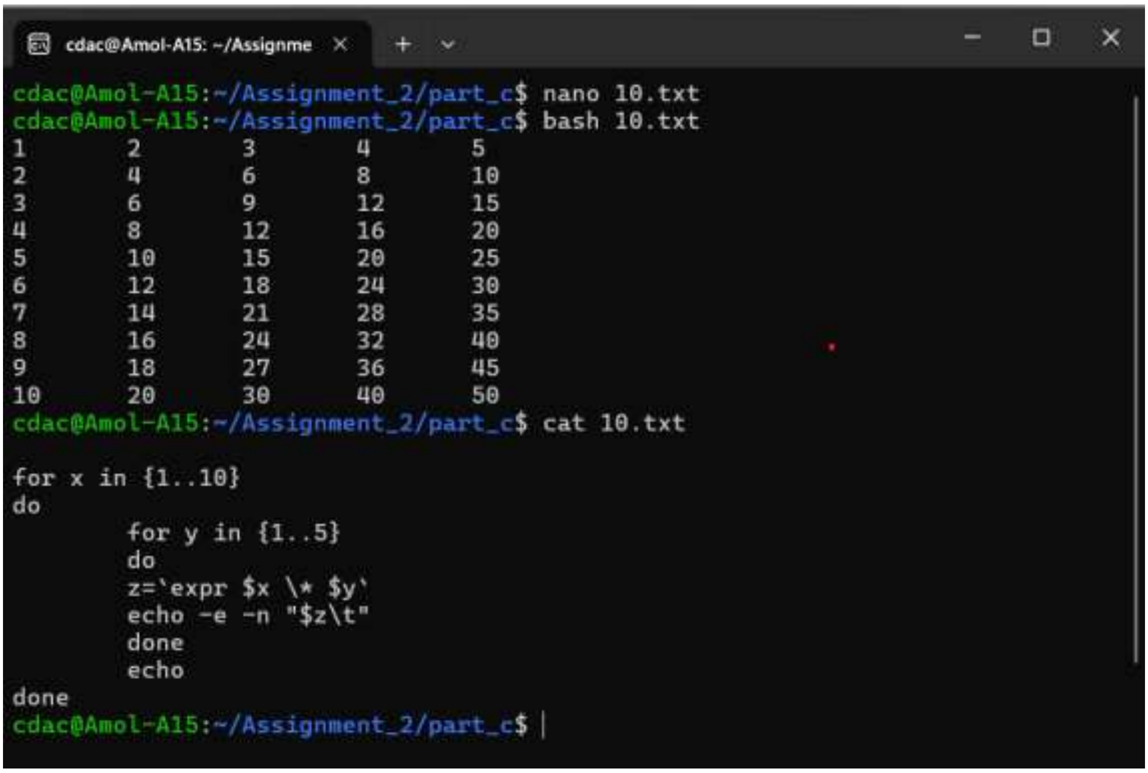
```
cdac@Amol-A15:~/Assignment_2/part_c$ nano 8.txt
cdac@Amol-A15:~/Assignment_2/part_c$ ls
1.txt  2.txt  3.txt  4.txt  5.txt  6.txt  7.txt  8.txt
cdac@Amol-A15:~/Assignment_2/part_c$ bash 8.txt
File does not exist
cdac@Amol-A15:~/Assignment_2/part_c$ touch file.txt
cdac@Amol-A15:~/Assignment_2/part_c$ ls
1.txt  2.txt  3.txt  4.txt  5.txt  6.txt  7.txt  8.txt  file.txt
cdac@Amol-A15:~/Assignment_2/part_c$ bash 8.txt
File exist
cdac@Amol-A15:~/Assignment_2/part_c$ cat 8.txt
if [ -f "file.txt" ]
then
echo "File exist"
else
echo "File does not exist"
fi
cdac@Amol-A15:~/Assignment_2/part_c$ |
```

9) Write a shell script that uses the if statement to check if a number is greater than 10 and prints a message accordingly.



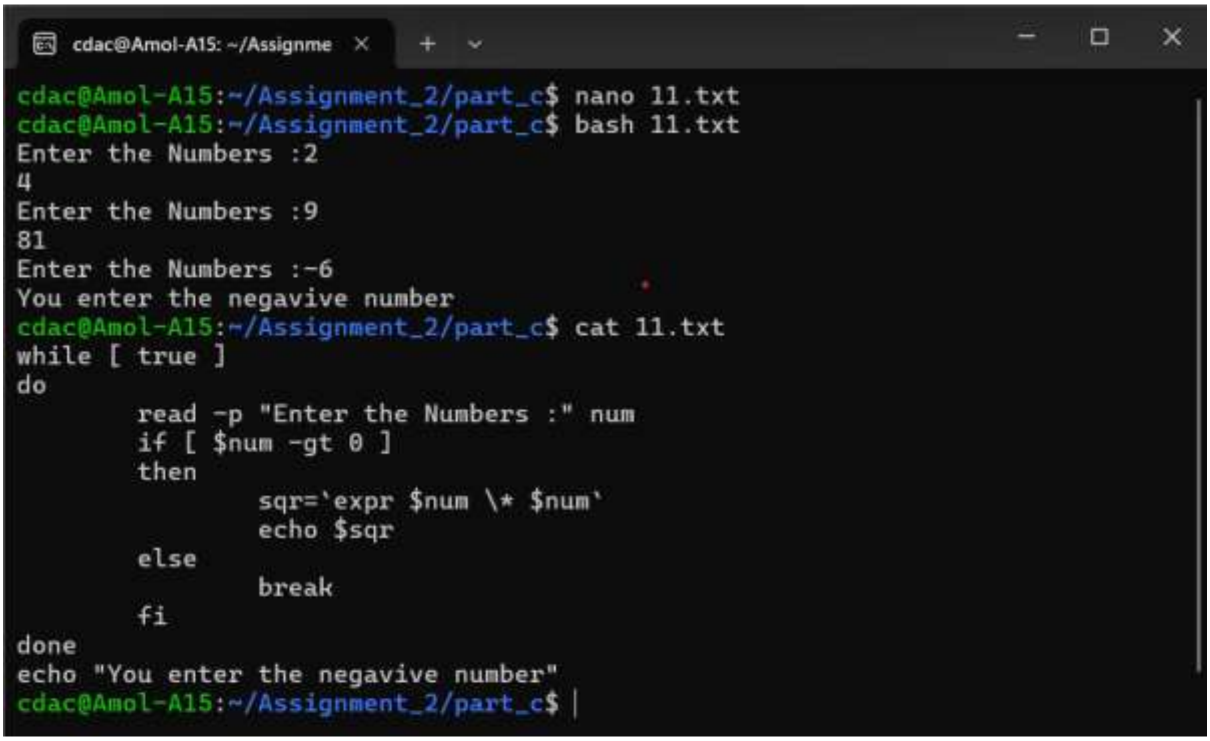
```
cdac@Amol-A15:~/Assignment_2/part_c$ nano 9.txt
cdac@Amol-A15:~/Assignment_2/part_c$ bash 9.txt
Enter Number for checking : 15
15 is greater than 10
cdac@Amol-A15:~/Assignment_2/part_c$ bash 9.txt
Enter Number for checking : 3
3 is less than 10
cdac@Amol-A15:~/Assignment_2/part_c$ cat 9.txt

read -p "Enter Number for checking : " num1

if [ $num1 -gt 10 ]
then
echo "$num1 is greater than 10"
else
echo "$num1 is less than 10"
fi
cdac@Amol-A15:~/Assignment_2/part_c$ |
```

10) Write a shell script that uses nested for loops to print a multiplication table for numbers from 1 to 5. The output should be formatted nicely, with each row representing a number and each column representing the multiplication result for that number.



```
cdac@Amol-A15:~/Assignment_2/part_c$ nano 10.txt
cdac@Amol-A15:~/Assignment_2/part_c$ bash 10.txt
1       2       3       4       5
2       4       6       8       10
3       6       9       12      15
4       8       12      16      20
5       10      15      20      25
6       12      18      24      30
7       14      21      28      35
8       16      24      32      40
9       18      27      36      45
10      20      30      40      50
cdac@Amol-A15:~/Assignment_2/part_c$ cat 10.txt

for x in {1..10}
do
        for y in {1..5}
        do
        z=`expr $x \* $y`
        echo -e -n "$z\t"
        done
        echo
done
cdac@Amol-A15:~/Assignment_2/part_c$ |
```

11) Write a shell script that uses a while loop to read numbers from the user until the user enters a negative number. For each positive number entered, print its square. Use the break statement to exit the loop when a negative number is entered.



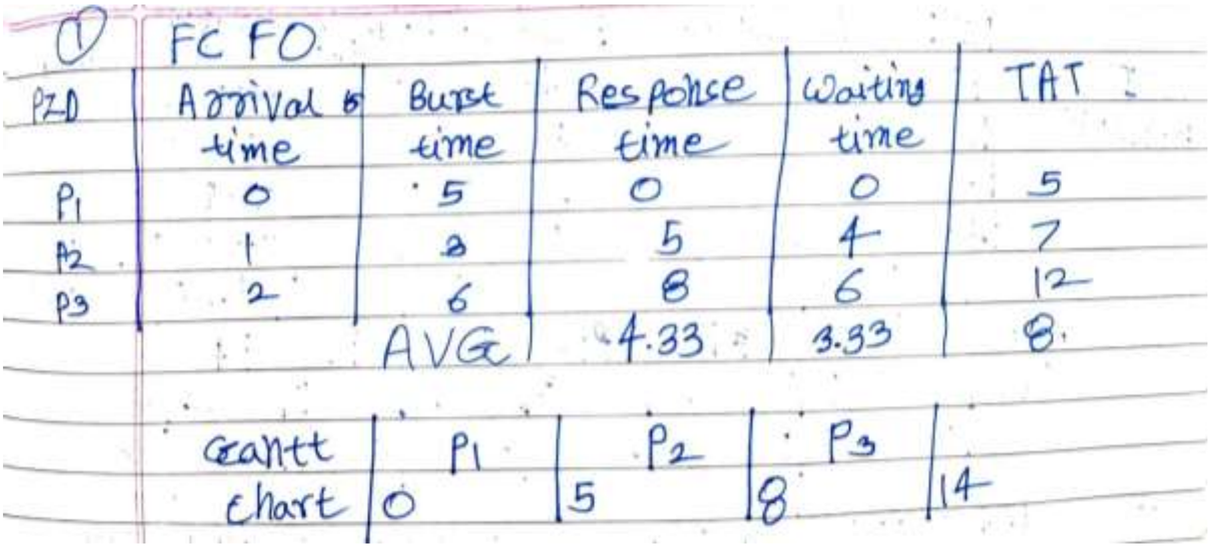
```
cdac@Amol-A15:~/Assignment_2/part_c$ nano 11.txt
cdac@Amol-A15:~/Assignment_2/part_c$ bash 11.txt
Enter the Numbers :2
4
Enter the Numbers :9
81
Enter the Numbers :-6
You enter the negavive number
cdac@Amol-A15:~/Assignment_2/part_c$ cat 11.txt
while [ true ]
do
        read -p "Enter the Numbers :" num
        if [ $num -gt 0 ]
        then
                sqr=`expr $num \* $num`
                echo $sqr
        else
                break
        fi
done
echo "You enter the negavive number"
cdac@Amol-A15:~/Assignment_2/part_c$ |
```

Part E

1. Consider the following processes with arrival times and burst times: |

Process	Arrival Time	Burst Time
P1	0	5
P2	1	3
P3	2	6

   Calculate the average waiting time using First-Come, First-Served (FCFS) scheduling.



① FC FO

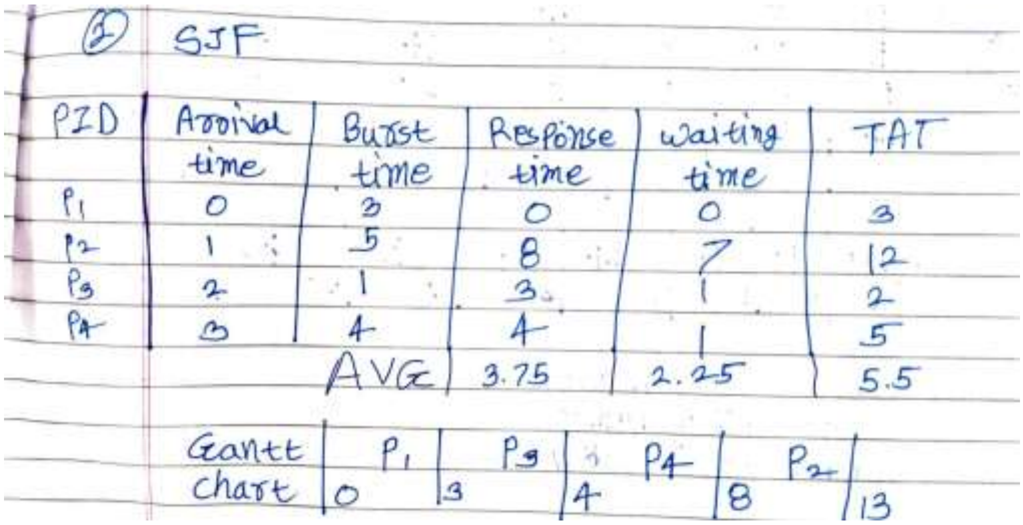
PID	Arrival time	Burst time	Response time	Waiting time	TAT
P1	0	5	0	0	5
P2	1	3	5	4	7
P3	2	6	8	6	12
		AVG	4.33	3.33	8.

Gantt chart	P1	P2	P3	
	0	5	8	14

2. Consider the following processes with arrival times and burst times:

Process	Arrival Time	Burst Time
P1	0	3
P2	1	5
P3	2	1
P4	3	4

   Calculate the average turnaround time using Shortest Job First (SJF) scheduling.



② SJF

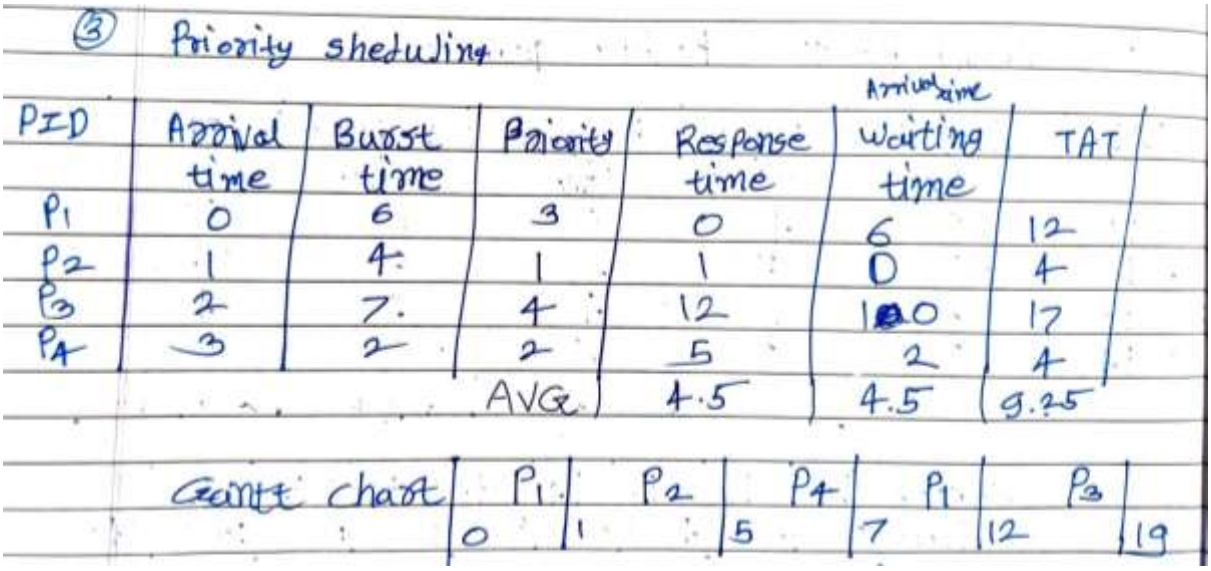
PID	Arrival time	Burst time	Response time	waiting time	TAT
P1	0	3	0	0	3
P2	1	5	8	7	12
P3	2	1	3	1	2
P4	3	4	4	1	5
		AVG	3.75	2.25	5.5

Gantt chart	P1	P3	P4	P2	
	0	3	4	8	13

3. Consider the following processes with arrival times, burst times, and priorities (lower number indicates higher priority):

Process	Arrival Time	Burst Time	Priority
P1	0	6	3
P2	1	4	1
P3	2	7	4
P4	3	2	2

Calculate the average waiting time using Priority Scheduling.



③ Priority sheduling

PID	Arrival time	Burst time	Priority	Response time	Arrival time waiting time	TAT
P1	0	6	3	0	6	12
P2	1	4	1	1	0	4
P3	2	7	4	12	10	17
P4	3	2	2	5	2	4
			AVG	4.5	4.5	9.25

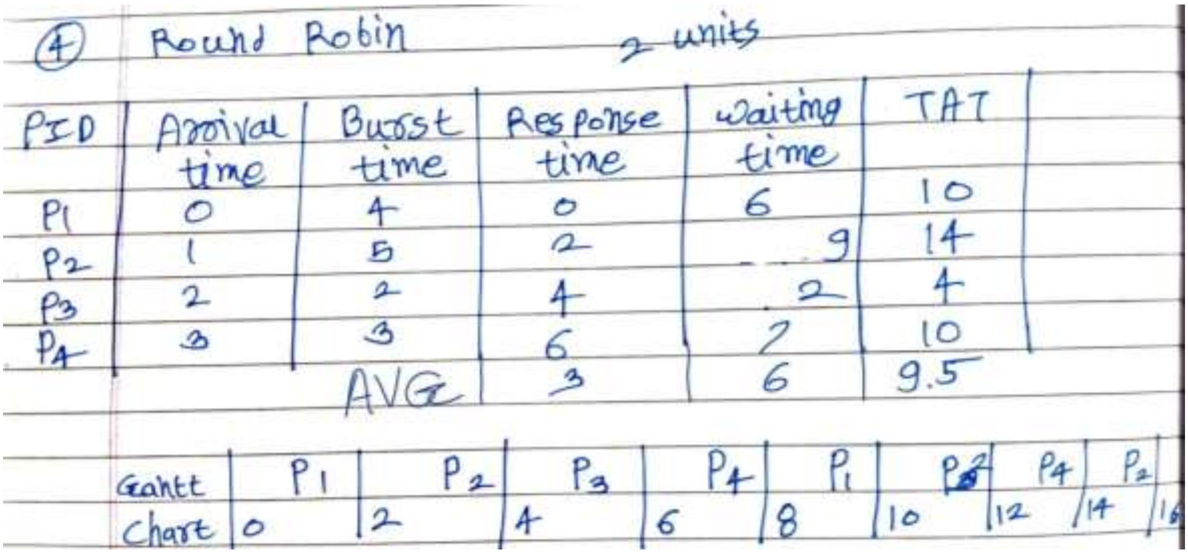
Gantt chart

P1	P2	P4	P1	P3	
0	1	5	7	12	19

4. Consider the following processes with arrival times and burst times, and the time quantum for Round Robin scheduling is 2 units:

Process	Arrival Time	Burst Time
P1	0	4
P2	1	5
P3	2	2
P4	3	3

Calculate the average turnaround time using Round Robin scheduling.

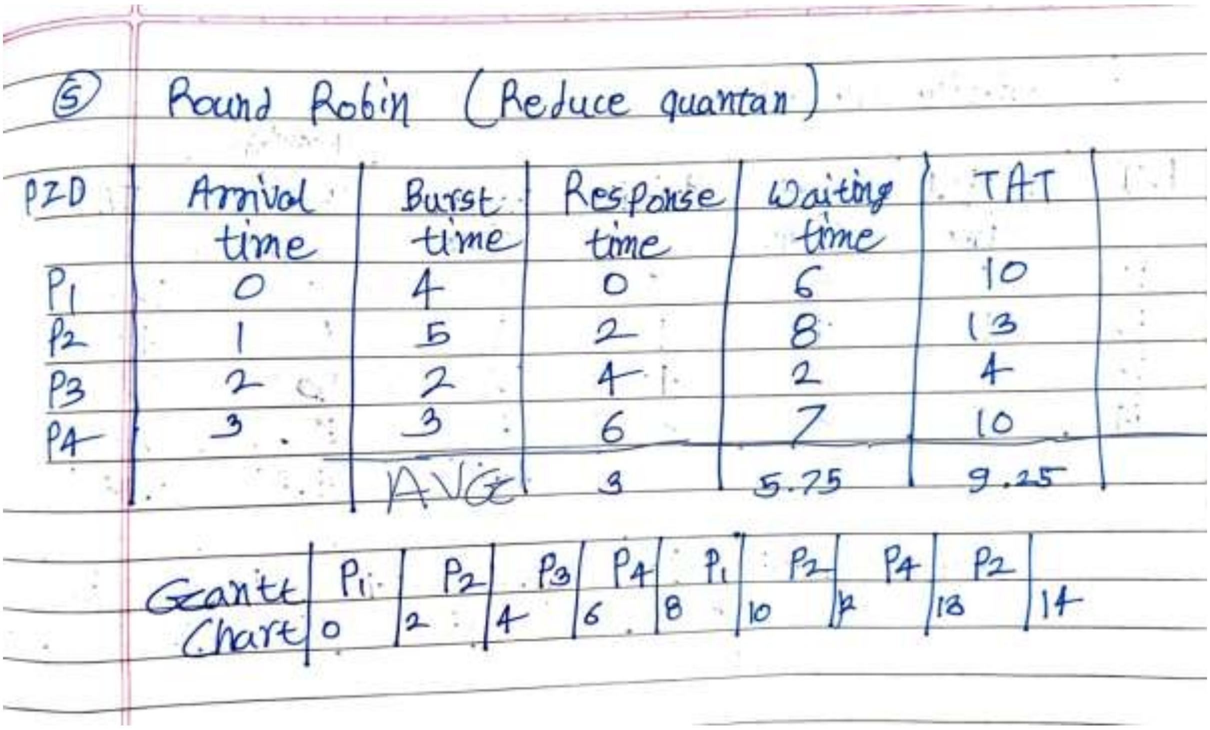


④ Round Robin 2 units

PID	Arrival time	Burst time	Response time	Waiting time	TAT
P1	0	4	0	6	10
P2	1	5	2	9	14
P3	2	2	4	2	4
P4	3	3	6	7	10
		AVG	3	6	9.5

Gantt Chart

P1	P2	P3	P4	P1	P2	P4	P2	
0	2	4	6	8	10	12	14	16



⑤ Round Robin (Reduce quantan)

PID	Arrival time	Burst time	Response time	Waiting time	TAT
P1	0	4	0	6	10
P2	1	5	2	8	13
P3	2	2	4	2	4
P4	3	3	6	7	10
		AVG	3	5.75	9.25

Gantt Chart	P1	P2	P3	P4	P1	P2	P4	P2	
	0	2	4	6	8	10	12	13	14

5. Consider a program that uses the fork() system call to create a child process. Initially, the parent process has a variable x with a value of 5. After forking, both the parent and child processes increment the value of x by 1. What will be the final values of x in the parent and child processes after the fork() call?

- When the fork() method call is used, it creates a child process that has its own copy of the parent's memory.
- Before forking, the parent has a variable x = 5. After the fork, both the parent and child have separate copies of x, but still same equal to 5.
- Each process then increase by 1, so both the parent and child have x = 6.