ASSIGNMENT 7

Q.) WAP to implement non-preemptive SJF scheduling algorithm

Consider process name, burst time, waiting time and turn-around time as the important

parameters.

Consider arrival time as same for all processes.

Show the output in in tabular format.

| Process Name | Burst Time | Waiting Time | Turnaround Time |

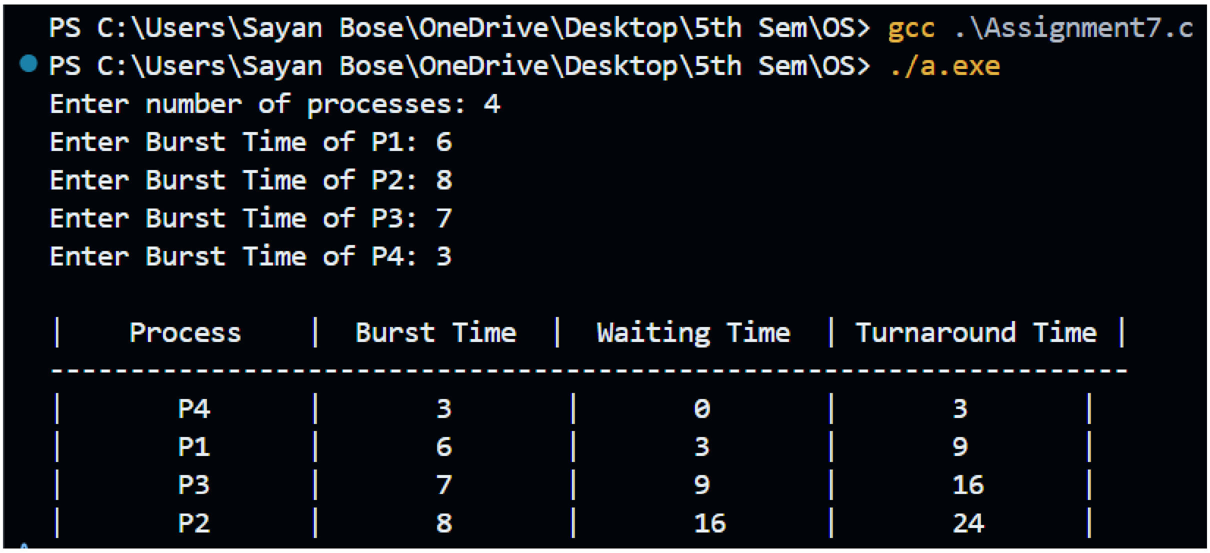
Pseudocode ->

```
Input number of processes n
For each process i:
    Read burst_time[i], assign process name
Sort processes by burst time
waiting_time[0] = 0
For i = 1 to n-1:
    waiting_time[i] = waiting_time[i-1] + burst_time[i-1]
For i = 0 to n-1:
    turnaround_time[i] = waiting_time[i] + burst_time[i]
Print process name, burst time, waiting time, turnaround time
```

Code Implementation in C ->

```
#include <stdio.h>
struct Process {
   int id, bt, wt, tat;
};
int main() {
   int n;
   printf("Enter number of processes: ");
   scanf("%d", &n);
   struct Process p[n];
   for (int i = 0; i < n; i++) {
      p[i].id = i+1;
      printf("Enter Burst Time of P%d: ", i+1);
      scanf("%d", &p[i].bt);
   }
   for (int i = 0; i < n-1; i++) {
      for (int j = i+1; j < n; j++) {
           if (p[i].bt > p[j].bt) {
           struct Process temp = p[i];
           p[i] = p[j];
           p[j] = temp;
           }
      }
   }
   p[0].wt = 0;
   p[0].tat = p[0].bt;
   for (int i = 1; i < n; i++) {
      p[i].wt = p[i-1].wt + p[i-1].bt;
      p[i].tat = p[i].wt + p[i].bt;
   }
   printf("\n|    Process    | Burst Time  | Waiting Time  | Turnaround Time |\n");
   printf("----------------------------------------------------------------\n");
   for (int i = 0; i < n; i++) {
      printf("|\tP%d\t|\t%d\t|\t%d\t|\t%d\t|\n",
      p[i].id, p[i].bt, p[i].wt, p[i].tat);
   }
   return 0;
}
```

Output ->



```
PS C:\Users\Sayan Bose\OneDrive\Desktop\5th Sem\OS> gcc .\Assignment7.c
PS C:\Users\Sayan Bose\OneDrive\Desktop\5th Sem\OS> ./a.exe
Enter number of processes: 4
Enter Burst Time of P1: 6
Enter Burst Time of P2: 8
Enter Burst Time of P3: 7
Enter Burst Time of P4: 3

|    Process    | Burst Time  | Waiting Time  | Turnaround Time |
-------------------------------------------------------------------
|       P4      |       3       |       0       |       3       |
|       P1      |       6       |       3       |       9       |
|       P3      |       7       |       9       |       16      |
|       P2      |       8       |       16      |       24      |
```

ASSIGNMENT 6

Q.) Write a program in C/C++ to implement FCFS CPU scheduling algorithm

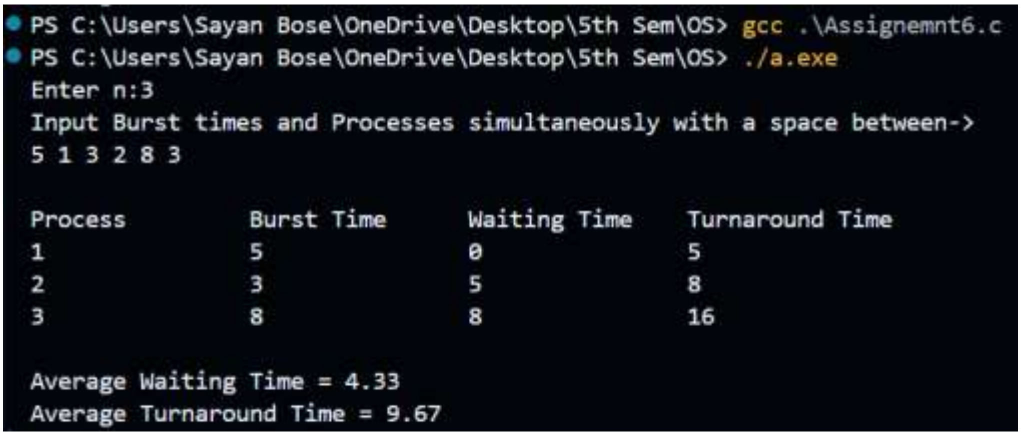
Pseudocode for FCFS Scheduling (where Arrival Time = 0) ->

```
Start

Input n // number of processes

For i = 1 to n do
    Input burst_time[i]
    process[i] = i
EndFor

// Since all arrival times = 0, execution order is same as input order

waiting_time[1] = 0
For i = 2 to n do
    waiting_time[i] = waiting_time[i-1] + burst_time[i-1]
EndFor
For i = 1 to n do
    turnaround_time[i] = waiting_time[i] + burst_time[i]
EndFor

// Calculate average times

total_wt = 0
total_tat = 0
For i = 1 to n do
    total_wt = total_wt + waiting_time[i]
    total_tat = total_tat + turnaround_time[i]
EndFor
avg_wt = total_wt / n
avg_tat = total_tat / n

// Display results
Print "Process Burst Time Waiting Time Turnaround Time"
For i = 1 to n do
    Print process[i], burst_time[i], waiting_time[i], turnaround_time[i]
EndFor
Print "Average Waiting Time = ", avg_wt
Print "Average Turnaround Time = ", avg_tat

End
```

Code Implementation in C ->

```c
#include <stdio.h>
int main(){
    int n;
    printf("Enter n:");
    scanf("%d",&n);
    int burst_time[n],process[n],waiting_time[n],turnaround_time[n];
    printf("Input (Burst times  Processes) ->\n");
    for(int i=0;i<n;i++){
        scanf("%d",&burst_time[i]);
        scanf("%d",&process[i]);
    }
    waiting_time[0]=0;
    for(int i=1;i<n;i++){
        waiting_time[i]=waiting_time[i-1]+burst_time[i-1];
    }
    for(int i=0;i<n;i++){
        turnaround_time[i]=waiting_time[i]+burst_time[i];
    }

    float total_wt = 0.0f, total_tat = 0.0f;
    for (int i = 0; i < n; i++) {
        total_wt += waiting_time[i];
        total_tat += turnaround_time[i];
    }
    float avg_wt = total_wt / n;
    float avg_tat = total_tat / n;
    printf("\nProcess  \tBurst Time\tWaiting Time\tTurnaround Time\n");
    for (int i = 0; i < n; i++) {
        printf("%d\t\t%d\t\t%d\t\t%d\n",
               process[i], burst_time[i], waiting_time[i], turnaround_time[i]);
    }
    printf("\nAverage Waiting Time = %.2f", avg_wt);
    printf("\nAverage Turnaround Time = %.2f\n", avg_tat);
    return 0;
}
```

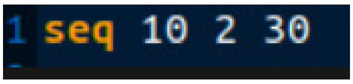
Output ->



```
PS C:\Users\Sayan Bose\OneDrive\Desktop\5th Sem\OS> gcc .\Assignemnt6.c
PS C:\Users\Sayan Bose\OneDrive\Desktop\5th Sem\OS> ./a.exe
Enter n:3
Input Burst times and Processes simultaneously with a space between->
5 1 3 2 8 3

Process         Burst Time      Waiting Time    Turnaround Time
1               5               0               5
2               3               5               8
3               8               8               16

Average Waiting Time = 4.33
Average Turnaround Time = 9.67
```

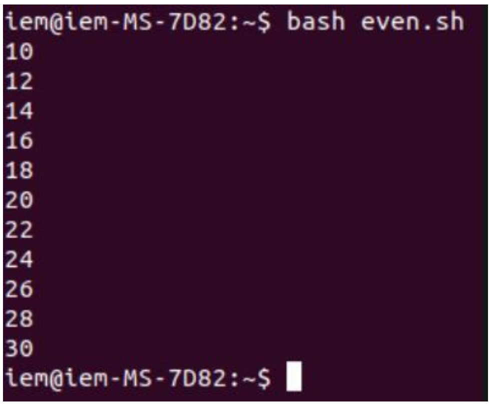
1. **Print even numbers from 10-30.**

<u>CODE:</u>



```
seq 10 2 30
```

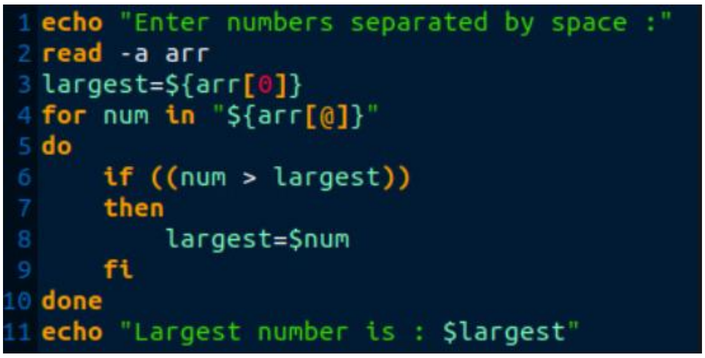
<u>OUTPUT:</u>



```
iem@iem-MS-7D82:~$ bash even.sh
10
12
14
16
18
20
22
24
26
28
30
iem@iem-MS-7D82:~$
```

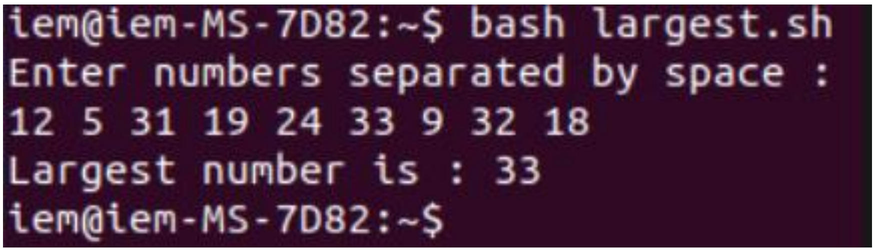
2. **Find the largest number in a list.**

<u>CODE:</u>



```
echo "Enter numbers separated by space :"
read -a arr
largest=${arr[0]}
for num in "${arr[@]}"
do
    if ((num > largest))
    then
        largest=$num
    fi
done
echo "Largest number is : $largest"
```

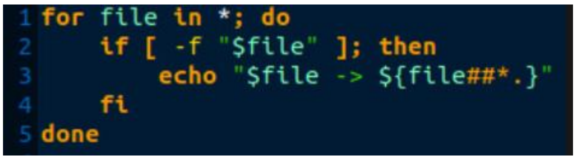
<u>OUTPUT:</u>



```
iem@iem-MS-7D82:~$ bash largest.sh
Enter numbers separated by space :
12 5 31 19 24 33 9 32 18
Largest number is : 33
iem@iem-MS-7D82:~$
```

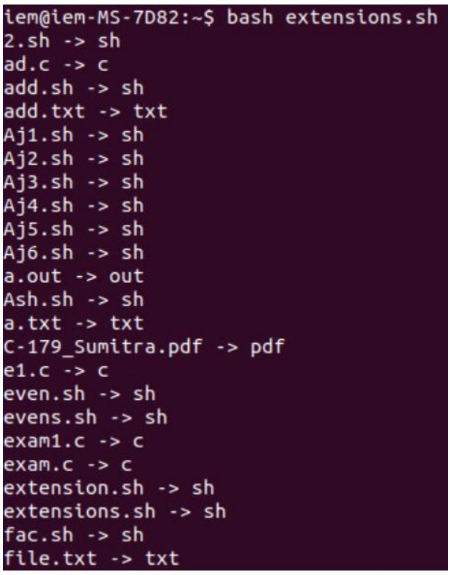
3. **Identify extension types of files in a current working directory.**

CODE:



```
for file in *; do
    if [ -f "$file" ]; then
        echo "$file -> ${file##*.}"
    fi
done
```

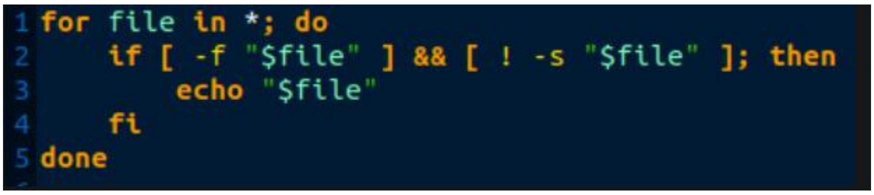
OUTPUT:



```
iem@iem-MS-7D82:~$ bash extensions.sh
2.sh -> sh
ad.c -> c
add.sh -> sh
add.txt -> txt
Aj1.sh -> sh
Aj2.sh -> sh
Aj3.sh -> sh
Aj4.sh -> sh
Aj5.sh -> sh
Aj6.sh -> sh
a.out -> out
Ash.sh -> sh
a.txt -> txt
C-179_Sumitra.pdf -> pdf
e1.c -> c
even.sh -> sh
evens.sh -> sh
exam1.c -> c
exam.c -> c
extension.sh -> sh
extensions.sh -> sh
fac.sh -> sh
file.txt -> txt
```

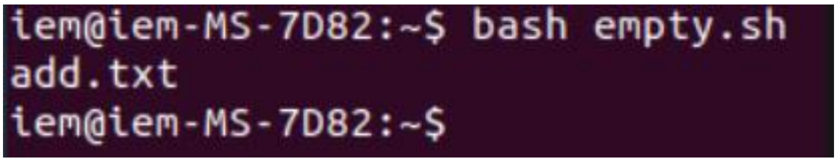
4. **Find all empty files in current directory.**

CODE:



```
for file in *; do
    if [ -f "$file" ] && [ ! -s "$file" ]; then
        echo "$file"
    fi
done
```

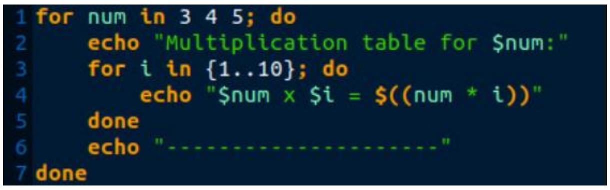
OUTPUT:



```
iem@iem-MS-7D82:~$ bash empty.sh
add.txt
iem@iem-MS-7D82:~$
```

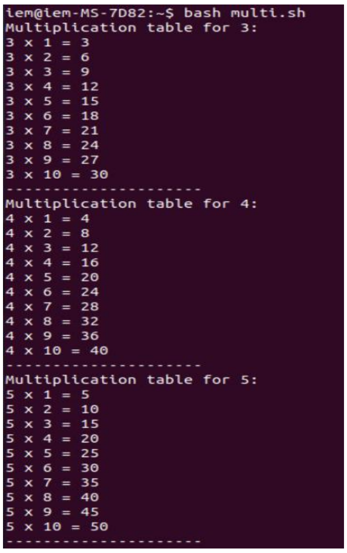
5. **Print the multiplication table for numbers 3 to 5.**

CODE:



```
for num in 3 4 5; do
    echo "Multiplication table for $num:"
    for i in {1..10}; do
        echo "$num x $i = $((num * i))"
    done
    echo "--------------------"
done
```

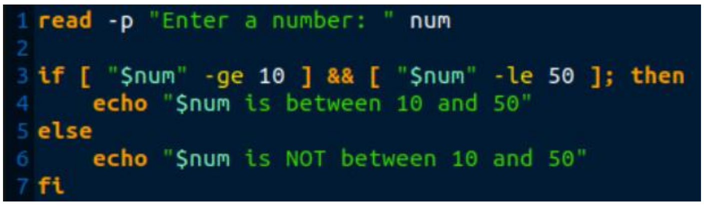
OUTPUT:



```
iem@iem-MS-7D82:~$ bash multi.sh
Multiplication table for 3:
3 x 1 = 3
3 x 2 = 6
3 x 3 = 9
3 x 4 = 12
3 x 5 = 15
3 x 6 = 18
3 x 7 = 21
3 x 8 = 24
3 x 9 = 27
3 x 10 = 30
--------------------
Multiplication table for 4:
4 x 1 = 4
4 x 2 = 8
4 x 3 = 12
4 x 4 = 16
4 x 5 = 20
4 x 6 = 24
4 x 7 = 28
4 x 8 = 32
4 x 9 = 36
4 x 10 = 40
--------------------
Multiplication table for 5:
5 x 1 = 5
5 x 2 = 10
5 x 3 = 15
5 x 4 = 20
5 x 5 = 25
5 x 6 = 30
5 x 7 = 35
5 x 8 = 40
5 x 9 = 45
5 x 10 = 50
--------------------
```

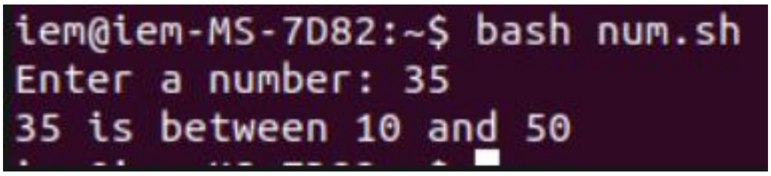
6. **Check if the given number is between 10 and 50.**

CODE:



```
read -p "Enter a number: " num

if [ "$num" -ge 10 ] && [ "$num" -le 50 ]; then
    echo "$num is between 10 and 50"
else
    echo "$num is NOT between 10 and 50"
fi
```

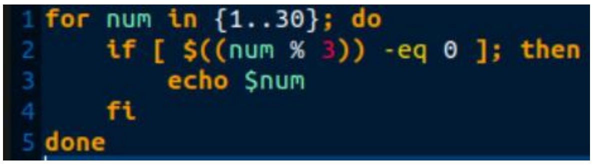
OUTPUT:



```
iem@iem-MS-7D82:~$ bash num.sh
Enter a number: 35
35 is between 10 and 50
```

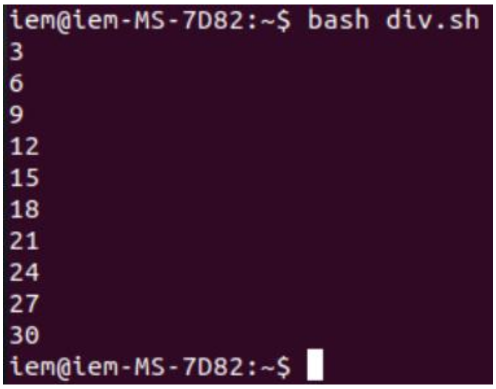
7. **Print the number divisible by 3 in a given range. (Suppose 1-30)**

CODE:



```
for num in {1..30}; do
    if [ $((num % 3)) -eq 0 ]; then
        echo $num
    fi
done
```

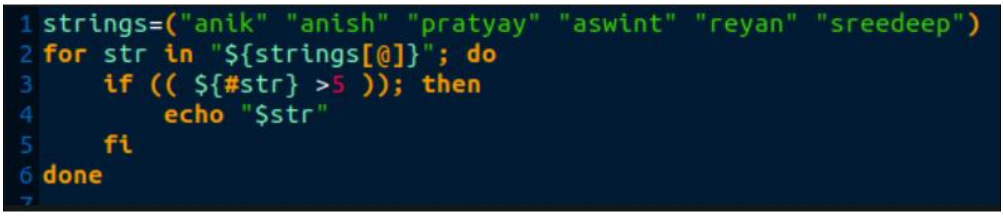
OUTPUT:



```
iem@iem-MS-7D82:~$ bash div.sh
3
6
9
12
15
18
21
24
27
30
iem@iem-MS-7D82:~$ 
```

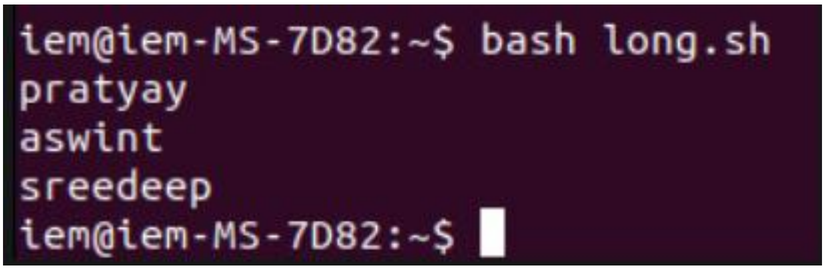
8. **Print the strings longer than 5 characters.**

CODE:



```
strings=("anik" "anish" "pratyay" "aswint" "reyan" "sreedeep")
for str in "${strings[@]}"; do
    if (( ${#str} >5 )); then
        echo "$str"
    fi
done
```

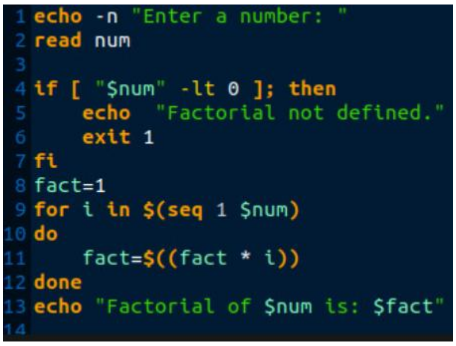
OUTPUT:



```
iem@iem-MS-7D82:~$ bash long.sh
pratyay
aswint
sreedeep
iem@iem-MS-7D82:~$ 
```

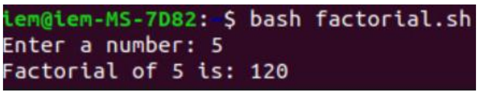
1. **Write a shell program to calculate the factorial of a number.**

   CODE:



```bash
echo -n "Enter a number: "
read num

if [ "$num" -lt 0 ]; then
    echo  "Factorial not defined."
    exit 1
fi
fact=1
for i in $(seq 1 $num)
do
    fact=$((fact * i))
done
echo "Factorial of $num is: $fact"
```

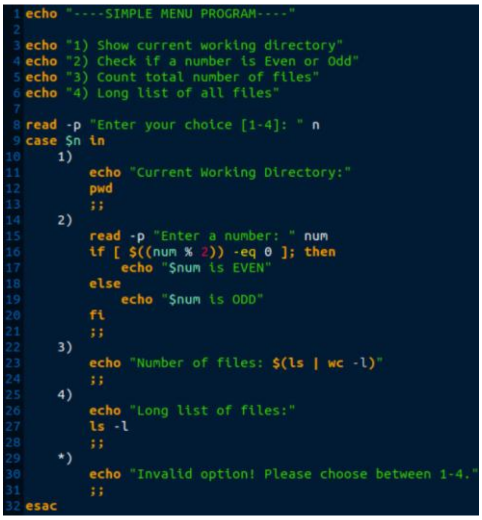
   OUTPUT:



```
iem@iem-MS-7D82:~$ bash factorial.sh
Enter a number: 5
Factorial of 5 is: 120
```

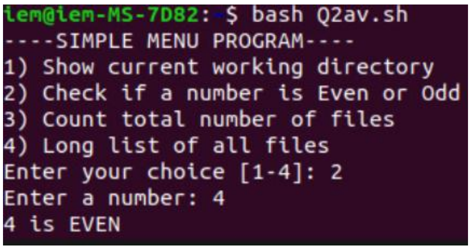
2. **Write a shell menu driven program to do the following: a. Display the current working directory. b. Check whether an input number is even or odd. c. Display the number of counts of all the files in the directory. d. Print the long listing of all the files.**

   CODE:



```bash
echo "----SIMPLE MENU PROGRAM----"

echo "1) Show current working directory"
echo "2) Check if a number is Even or Odd"
echo "3) Count total number of files"
echo "4) Long list of all files"

read -p "Enter your choice [1-4]: " n
case $n in
    1)
        echo "Current Working Directory:"
        pwd
        ;;
    2)
        read -p "Enter a number: " num
        if [ $((num % 2)) -eq 0 ]; then
            echo "$num is EVEN"
        else
            echo "$num is ODD"
        fi
        ;;
    3)
        echo "Number of files: $(ls | wc -l)"
        ;;
    4)
        echo "Long list of files:"
        ls -l
        ;;
    *)
        echo "Invalid option! Please choose between 1-4."
        ;;
esac
```

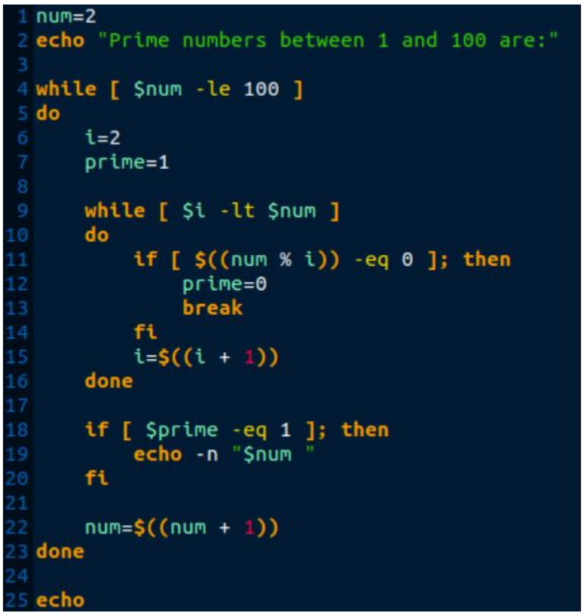
   OUTPUT:



```
iem@iem-MS-7D82:~$ bash Q2av.sh
----SIMPLE MENU PROGRAM----
1) Show current working directory
2) Check if a number is Even or Odd
3) Count total number of files
4) Long list of all files
Enter your choice [1-4]: 2
Enter a number: 4
4 is EVEN
```

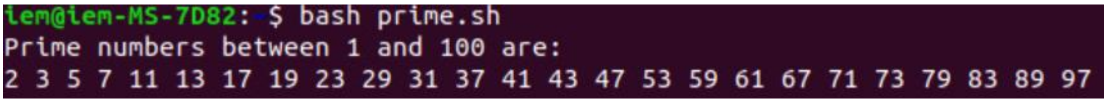
3. **Write a shell program to display all the prime numbers between 1 to 100 using while loop.**

CODE:



```
num=2
echo "Prime numbers between 1 and 100 are:"

while [ $num -le 100 ]
do
    i=2
    prime=1

    while [ $i -lt $num ]
    do
        if [ $((num % i)) -eq 0 ]; then
            prime=0
            break
        fi
        i=$((i + 1))
    done

    if [ $prime -eq 1 ]; then
        echo -n "$num "
    fi

    num=$((num + 1))
done

echo
```

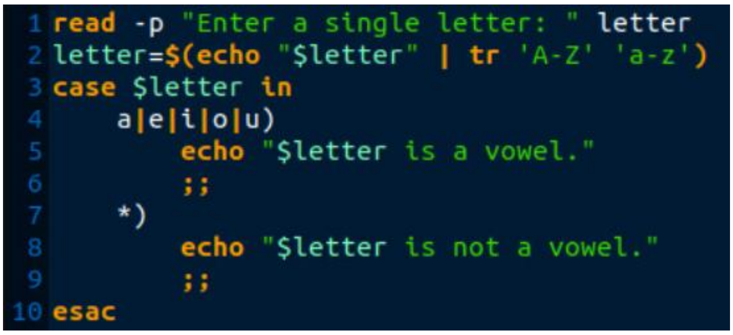
OUTPUT:



```
iem@iem-MS-7D82:~$ bash prime.sh
Prime numbers between 1 and 100 are:
2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71 73 79 83 89 97
```

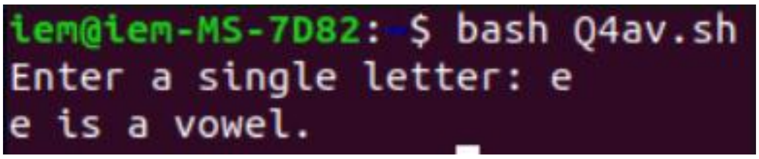
4. **Write a menu program to find out whether a given letter is vowel or not.**

CODE:



```
read -p "Enter a single letter: " letter
letter=$(echo "$letter" | tr 'A-Z' 'a-z')
case $letter in
    a|e|i|o|u)
        echo "$letter is a vowel."
        ;;
    *)
        echo "$letter is not a vowel."
        ;;
esac
```

OUTPUT:

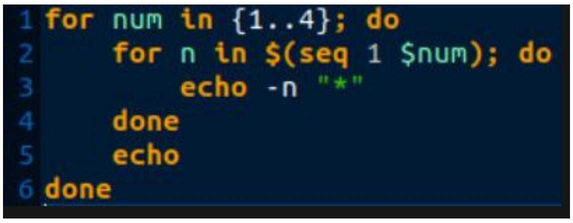


```
iem@iem-MS-7D82:~$ bash Q4av.sh
Enter a single letter: e
e is a vowel.
```

5. **Write a shell script which will generate the output as follows:**

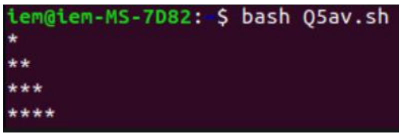
```
*
* *
* * *
* * * *
```

CODE:



```
for num in {1..4}; do
    for n in $(seq 1 $num); do
        echo -n "*"
    done
    echo
done
```

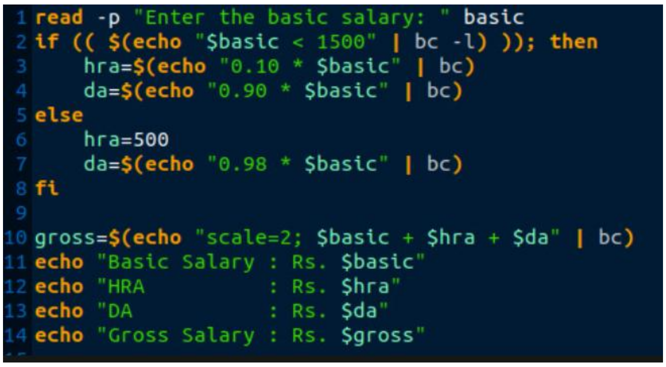
OUTPUT:



```
iem@iem-MS-7D82:~$ bash Q5av.sh
*
**
***
****
```

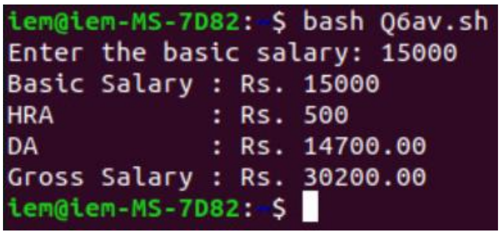
6. **Write a shell script that computes the gross salary of a employee according to the following rules: i)If basic salary is < 1500 then HRA =10% of the basic and DA =90% of the basic. ii)If basic salary is >=1500 then HRA =Rs500 and DA=98% of the basic. The basic salary is entered interactively through the key board.**

CODE:



```
read -p "Enter the basic salary: " basic
if (( $(echo "$basic < 1500" | bc -l) )); then
    hra=$(echo "0.10 * $basic" | bc)
    da=$(echo "0.90 * $basic" | bc)
else
    hra=500
    da=$(echo "0.98 * $basic" | bc)
fi

gross=$(echo "scale=2; $basic + $hra + $da" | bc)
echo "Basic Salary : Rs. $basic"
echo "HRA          : Rs. $hra"
echo "DA           : Rs. $da"
echo "Gross Salary : Rs. $gross"
```

OUTPUT:

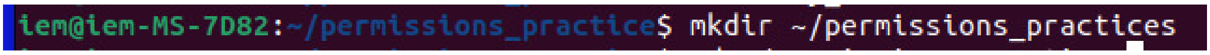


```
iem@iem-MS-7D82:~$ bash Q6av.sh
Enter the basic salary: 15000
Basic Salary : Rs. 15000
HRA          : Rs. 500
DA           : Rs. 14700.00
Gross Salary : Rs. 30200.00
iem@iem-MS-7D82:~$ 
```

ASSIGNMENT 3

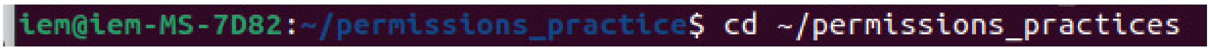
A. Exploring Default Permissions and Creating Files :

1. Create a working directory :



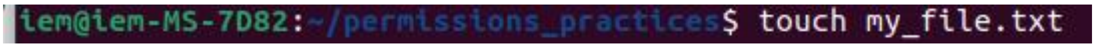
```
iem@iem-MS-7D82:~/permissions_practice$ mkdir ~/permissions_practices
```

2. Navigate into the directory :



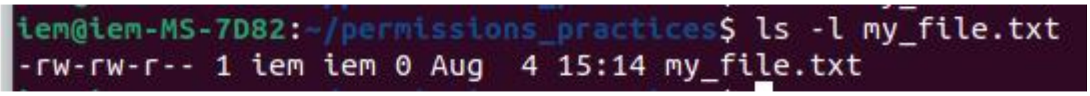
```
iem@iem-MS-7D82:~/permissions_practice$ cd ~/permissions_practices
```

3. Create a file :



```
iem@iem-MS-7D82:~/permissions_practices$ touch my_file.txt
```

4. View file permissions :



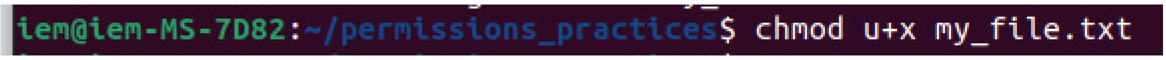
```
iem@iem-MS-7D82:~/permissions_practices$ ls -l my_file.txt
-rw-rw-r-- 1 iem iem 0 Aug  4 15:14 my_file.txt
```

Observation :

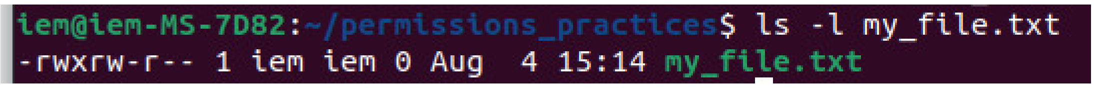
- **Owner** has : **read (r)** and **write (w)** permissions.
- **Group** has : **read (r)** and **write (w)** permissions.
- **Others** have : **read (r)** permission only.

B. Changing Permissions (Symbolic Mode) :

1. Add execute permission for the owner :



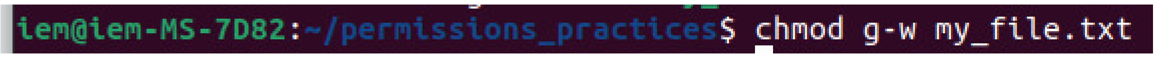
```
iem@iem-MS-7D82:~/permissions_practices$ chmod u+x my_file.txt
```



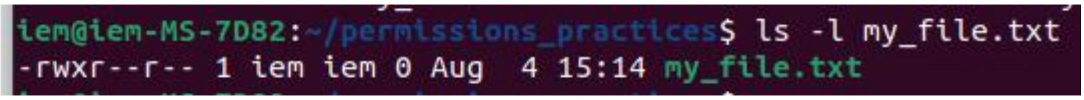
```
iem@iem-MS-7D82:~/permissions_practices$ ls -l my_file.txt
-rwxrw-r-- 1 iem iem 0 Aug  4 15:14 my_file.txt
```

Observation : The permission string changed by adding execute (x) permission for the owner.

2. Remove write permission for the group :



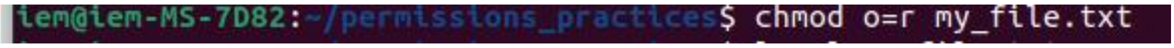
```
iem@iem-MS-7D82:~/permissions_practices$ chmod g-w my_file.txt
```



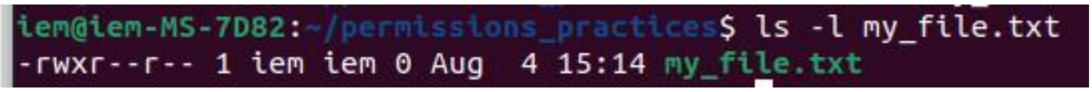
```
iem@iem-MS-7D82:~/permissions_practices$ ls -l my_file.txt
-rwxr--r-- 1 iem iem 0 Aug  4 15:14 my_file.txt
```

<u>Observation :</u> The write permission (w) for the group was successfully removed. Now, users in the group can read the file but cannot modify it.

3. Set read – only permission for others :



```
iem@iem-MS-7D82:~/permissions_practices$ chmod o=r my_file.txt
```

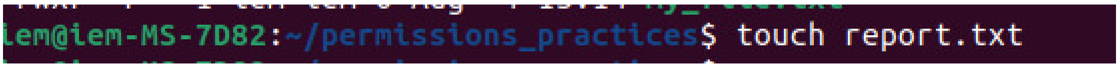


```
iem@iem-MS-7D82:~/permissions_practices$ ls -l my_file.txt
-rwxr--r-- 1 iem iem 0 Aug  4 15:14 my_file.txt
```

<u>Observation :</u> The permissions for others changed to read-only (r--). Now, other users can only view the file but cannot write to or execute it.

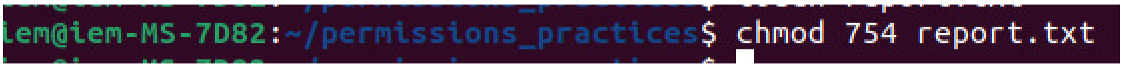
C. <u>Changing Permissions (Numeric Mode) :</u>

1. Set permissions for a new file :

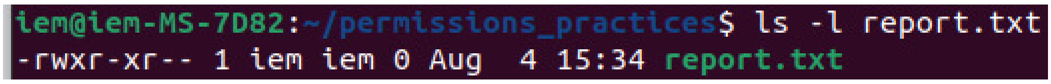


```
iem@iem-MS-7D82:~/permissions_practices$ touch report.txt
```

2. Set the permissions of report.txt such that :



```
iem@iem-MS-7D82:~/permissions_practices$ chmod 754 report.txt
```



```
iem@iem-MS-7D82:~/permissions_practices$ ls -l report.txt
-rwxr-xr-- 1 iem iem 0 Aug  4 15:34 report.txt
```

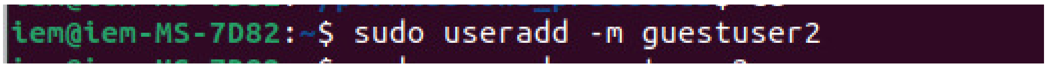
<u>Observation :</u>

The file report.txt is a regular file.

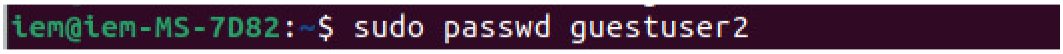
- The owner can read, write, and execute it.
- The group can read and execute, but not write.
- Others can only read the file.

D. Changing Ownership :

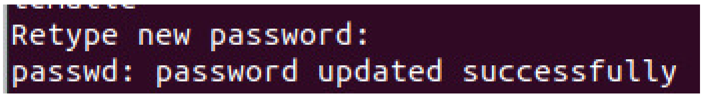
1. Create a user (Requires sudo or root access) :



```
iem@iem-MS-7D82:~$ sudo useradd -m guestuser2
```

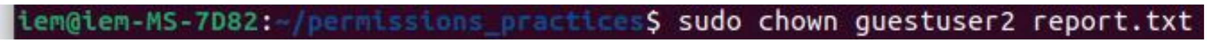


```
iem@iem-MS-7D82:~$ sudo passwd guestuser2
```

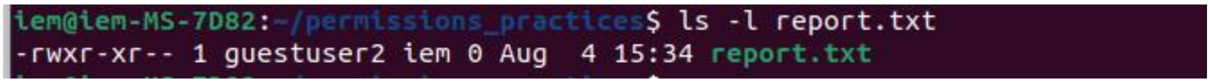


```
Retype new password:
passwd: password updated successfully
```

2. Change ownership of report.txt :



```
iem@iem-MS-7D82:~/permissions_practices$ sudo chown guestuser2 report.txt
```

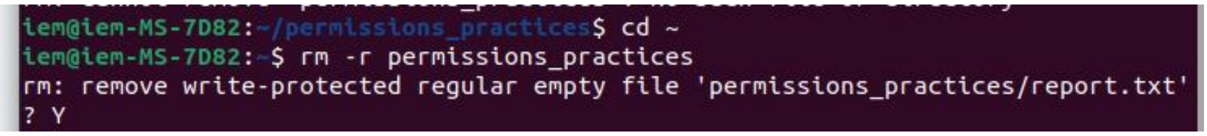


```
iem@iem-MS-7D82:~/permissions_practices$ ls -l report.txt
-rwxr-xr-- 1 guestuser2 iem 0 Aug  4 15:34 report.txt
```

Observation : After running the chown command, the file's ownership successfully changed. Now, guestuser has the permissions assigned to the owner, as per the permission string (rwx).

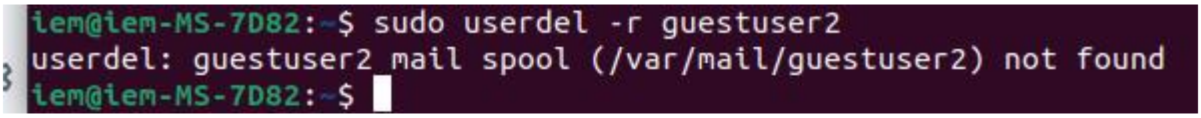
E. Clean up :

1. Navigate back to your home directory :



```
iem@iem-MS-7D82:~/permissions_practices$ cd ~
iem@iem-MS-7D82:~$ rm -r permissions_practices
rm: remove write-protected regular empty file 'permissions_practices/report.txt'
? Y
```

2. Remove the working directory and its contents :

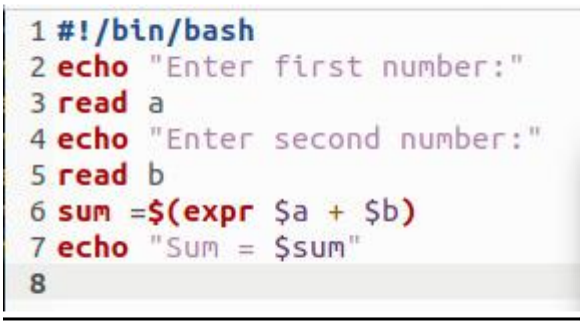


```
iem@iem-MS-7D82:~$ sudo userdel -r guestuser2
userdel: guestuser2 mail spool (/var/mail/guestuser2) not found
iem@iem-MS-7D82:~$
```

ASSIGNMENT 4

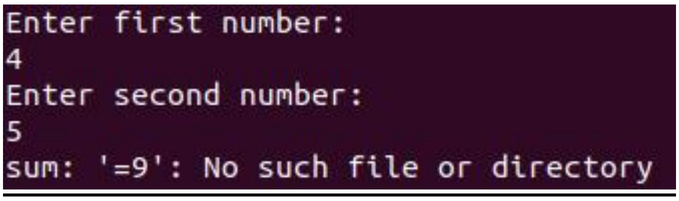
1. **Write a shell program to add two numbers given by user.**

Code :



```
#!/bin/bash
echo "Enter first number:"
read a
echo "Enter second number:"
read b
sum =$(expr $a + $b)
echo "Sum = $sum"

```

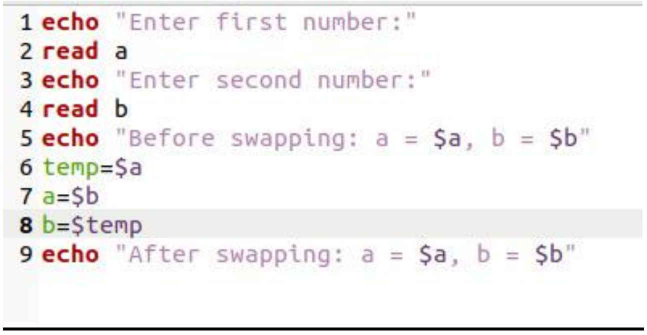
Output :



```
Enter first number:
4
Enter second number:
5
sum: '=9': No such file or directory
```

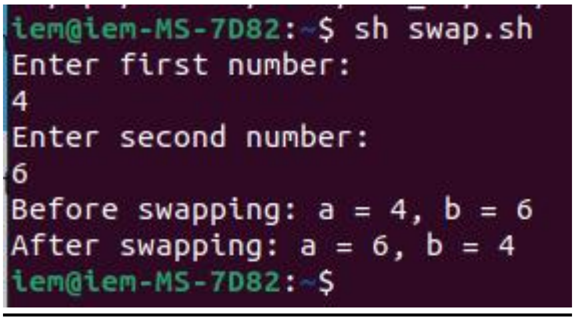
2. **Write a shell program to perform the swapping between two numbers taken from user during run time.**

Code :



```
echo "Enter first number:"
read a
echo "Enter second number:"
read b
echo "Before swapping: a = $a, b = $b"
temp=$a
a=$b
b=$temp
echo "After swapping: a = $a, b = $b"
```

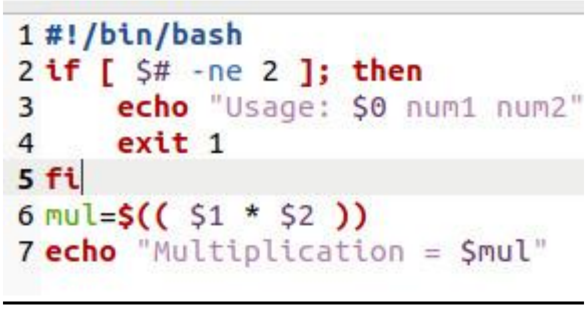
Output :



```
iem@iem-MS-7D82:~$ sh swap.sh
Enter first number:
4
Enter second number:
6
Before swapping: a = 4, b = 6
After swapping: a = 6, b = 4
iem@iem-MS-7D82:~$
```

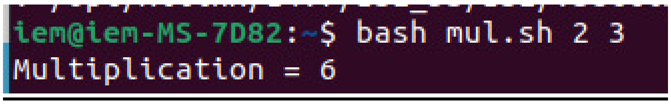
3. **Write a shell program to perform the multiply two numbers given as command line arguments.**

Code :



```
#!/bin/bash
if [ $# -ne 2 ]; then
    echo "Usage: $0 num1 num2"
    exit 1
fi
mul=$(( $1 * $2 ))
echo "Multiplication = $mul"
```

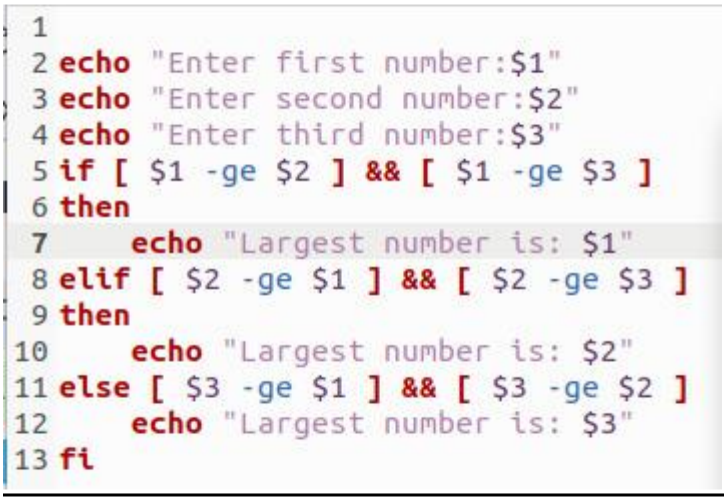
Output :



```
iem@iem-MS-7D82:~$ bash mul.sh 2 3
Multiplication = 6
```

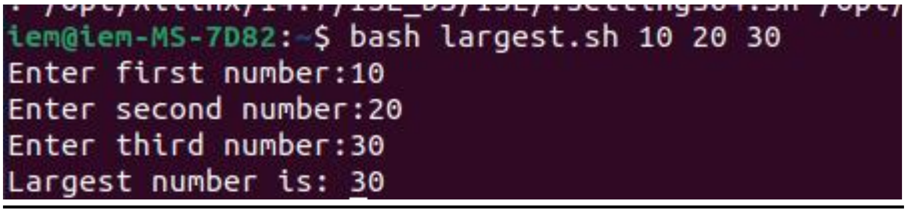
4. **Write a shell program to print the largest among three numbers by passing the numbers through command line arguments.**

Code :



```

echo "Enter first number:$1"
echo "Enter second number:$2"
echo "Enter third number:$3"
if [ $1 -ge $2 ] && [ $1 -ge $3 ]
then
    echo "Largest number is: $1"
elif [ $2 -ge $1 ] && [ $2 -ge $3 ]
then
    echo "Largest number is: $2"
else [ $3 -ge $1 ] && [ $3 -ge $2 ]
    echo "Largest number is: $3"
fi
```

Output :

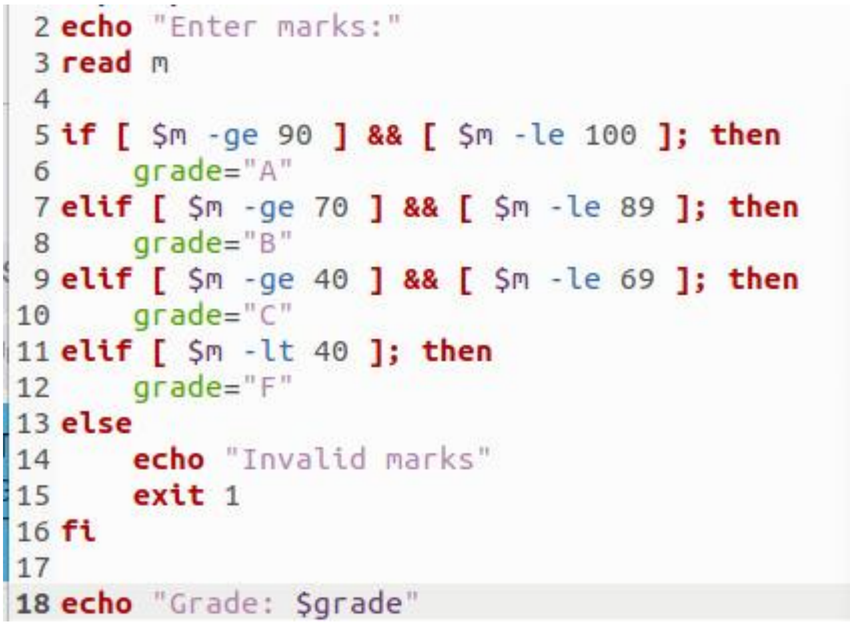


```
iem@iem-MS-7D82:~$ bash largest.sh 10 20 30
Enter first number:10
Enter second number:20
Enter third number:30
Largest number is: 30
```

5. **Write a shell program to display the following mark sheets of students by taking the input marks of student through the terminal.**

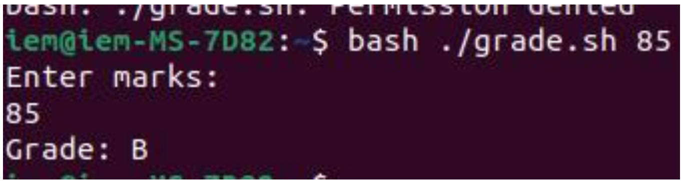
Marks range	Grade
90 >= M <= 100	A
70 >= M <= 89	B
40 >= M <= 69	C
M < 40	F

Code :



```
echo "Enter marks:"
read m

if [ $m -ge 90 ] && [ $m -le 100 ]; then
    grade="A"
elif [ $m -ge 70 ] && [ $m -le 89 ]; then
    grade="B"
elif [ $m -ge 40 ] && [ $m -le 69 ]; then
    grade="C"
elif [ $m -lt 40 ]; then
    grade="F"
else
    echo "Invalid marks"
    exit 1
fi

echo "Grade: $grade"
```

Output :



```
bash: ./grade.sh: Permission denied
iem@iem-MS-7D82:~$ bash ./grade.sh 85
Enter marks:
85
Grade: B
iem@iem-MS-7D82:~$
```

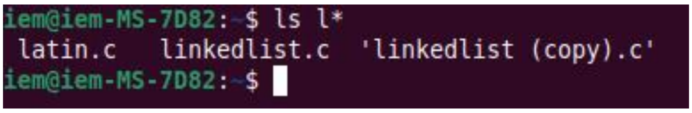
ASSIGNMENT 2

1. **For each command, give a brief description of what it does and how it can be used.**

A. Meta characters :

i. * wildcard :

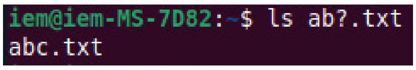
- **Description :** Wildcard to match any number of characters in filenames.
- **Syntax :** $ ls l*
- **Output :**



```
iem@iem-MS-7D82:~$ ls l*
 latin.c    linkedlist.c   'linkedlist (copy).c'
iem@iem-MS-7D82:~$ 
```

ii. ? question mark :

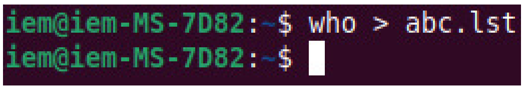
- **Description :** Wildcard to match exactly one character.
- **Syntax :** $ ls ab?.txt
- **Output :**



```
iem@iem-MS-7D82:~$ ls ab?.txt
abc.txt
```

iii. > redirection :

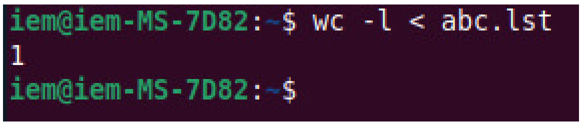
- **Description :** Redirects (overwrites) output to a file.
- **Syntax :** $ who > abc.lst
- **Output :**



```
iem@iem-MS-7D82:~$ who > abc.lst
iem@iem-MS-7D82:~$ 
```

iv. < redirection :

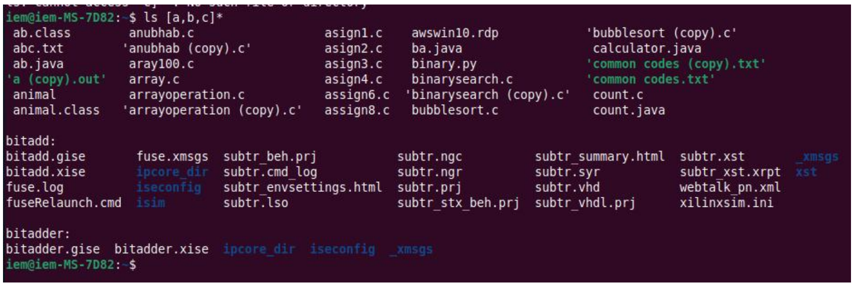
- **Description :** Takes input for a command from a file.
- **Syntax :** $ wc -l < abc.lst
- **Output :**



```
iem@iem-MS-7D82:~$ wc -l < abc.lst
1
iem@iem-MS-7D82:~$
```

v. [ ] brackets :

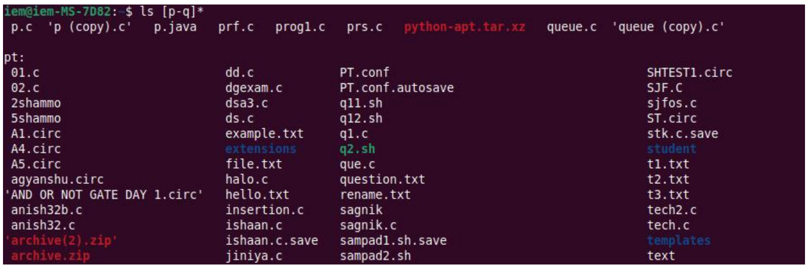
- **Description :** Matches any one of the characters inside brackets.
- **Syntax :** $ ls [a,b,c]*
- **Output :**



```
iem@iem-MS-7D82:~$ ls [a,b,c]*
 ab.class        anubhab.c                  asign1.c    awswin10.rdp              'bubblesort (copy).c'
 abc.txt        'anubhab (copy).c'          asign2.c    ba.java                    calculator.java
 ab.java         aray100.c                  asign3.c    binary.py                 'common codes (copy).txt'
'a (copy).out'   array.c                    asign4.c    binarysearch.c            'common codes.txt'
 animal          arrayoperation.c           assign6.c  'binarysearch (copy).c'     count.c
 animal.class   'arrayoperation (copy).c'   assign8.c   bubblesort.c               count.java

bitadd:
bitadd.gise       fuse.xmsgs  subtr_beh.prj          subtr.ngc          subtr_summary.html  subtr.xst       _xmsgs
bitadd.xise       ipcore_dir  subtr.cmd_log          subtr.ngr          subtr.syr           subtr_xst.xrpt  xst
fuse.log          iseconfig   subtr_envsettings.html subtr.prj          subtr.vhd           webtalk_pn.xml
fuseRelaunch.cmd  isim        subtr.lso              subtr_stx_beh.prj  subtr_vhdl.prj      xilinxsim.ini

bitadder:
bitadder.gise  bitadder.xise  ipcore_dir  iseconfig  _xmsgs
iem@iem-MS-7D82:~$
```

vi. – hyphen :

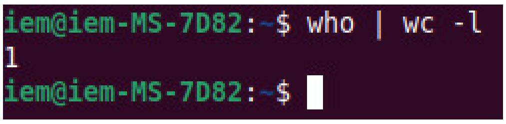
- **Description :** Indicates a range inside brackets.
- **Syntax :** $ ls [p-q]*
- **Output :**



```
iem@iem-MS-7D82:~$ ls [p-q]*
 p.c  'p (copy).c'   p.java   prf.c   prog1.c   prs.c   python-apt.tar.xz   queue.c  'queue (copy).c'

pt:
 01.c                           dd.c            PT.conf                                   SHTEST1.circ
 02.c                           dgexam.c        PT.conf.autosave                          SJF.C
 2shammo                        dsa3.c          q11.sh                                    sjfos.c
 5shammo                        ds.c            q12.sh                                    ST.circ
 A1.circ                        example.txt     q1.c                                      stk.c.save
 A4.circ                        extensions      q2.sh                                     student
 A5.circ                        file.txt        que.c                                     t1.txt
 agyanshu.circ                  halo.c          question.txt                              t2.txt
'AND OR NOT GATE DAY 1.circ'    hello.txt       rename.txt                                t3.txt
 anish32b.c                     insertion.c     sagnik                                    tech2.c
 anish32.c                      ishaan.c        sagnik.c                                  tech.c
'archive(2).zip'                ishaan.c.save   sampad1.sh.save                           templates
 archive.zip                    jiniya.c        sampad2.sh                                text
```

vii. | pipe :

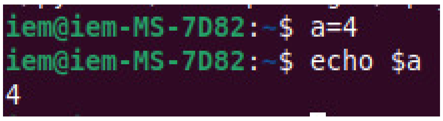
- **Description :** Passes the output of one command as input to another.
- **Syntax :** $ who | wc -l
- **Output :**



```
iem@iem-MS-7D82:~$ who | wc -l
1
iem@iem-MS-7D82:~$ 
```

viii. $ (system) variable :

- **Description :** Used to access shell variable values.
- **Syntax :** $ a = 4
- **Output :**

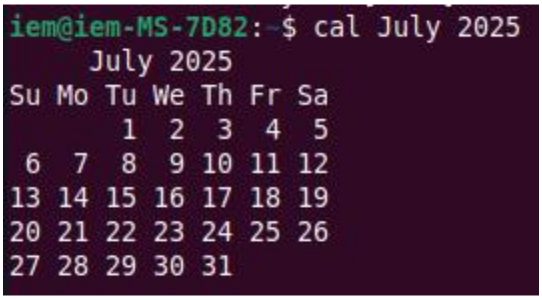


```
iem@iem-MS-7D82:~$ a=4
iem@iem-MS-7D82:~$ echo $a
4
```

B. UNIX commands :

ix. cal :

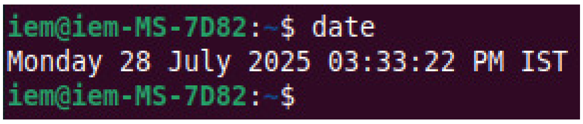
- **Description :** Displays calendar.
- **Syntax :** $ cal July 2025
- **Output :**



```
iem@iem-MS-7D82:~$ cal July 2025
     July 2025
Su Mo Tu We Th Fr Sa
       1  2  3  4  5
 6  7  8  9 10 11 12
13 14 15 16 17 18 19
20 21 22 23 24 25 26
27 28 29 30 31
```

x. date :

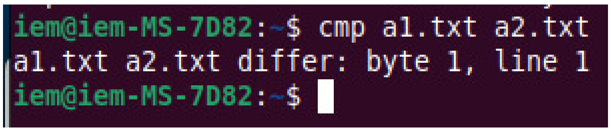
- **Description :** Displays current date and time.
- **Syntax :** $ date
- **Output :**



```
iem@iem-MS-7D82:~$ date
Monday 28 July 2025 03:33:22 PM IST
iem@iem-MS-7D82:~$
```

xi. cmp :

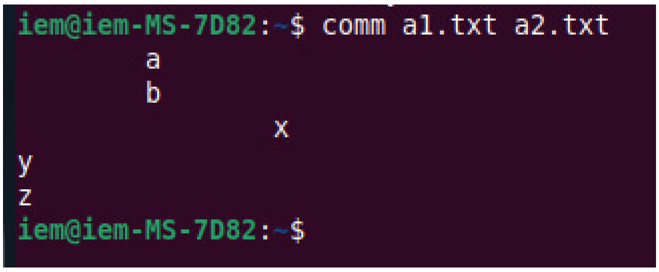
- **Description :** Compares two files byte by byte.
- **Syntax :** $ cmp a1.txt a2.txt
- **Output :**



```
iem@iem-MS-7D82:~$ cmp a1.txt a2.txt
a1.txt a2.txt differ: byte 1, line 1
iem@iem-MS-7D82:~$
```

xii. comm :

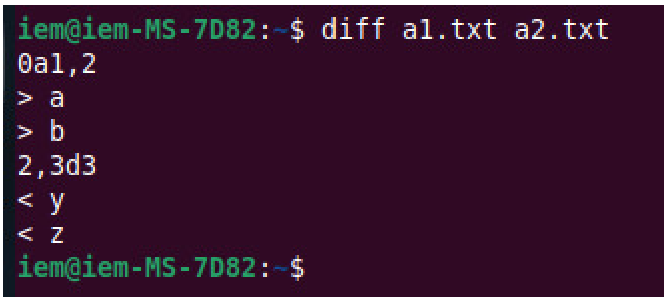
- **Description :** Compares two sorted files line by line.
- **Syntax :** $ comm a1.txt a2.txt
- **Output :**



```
iem@iem-MS-7D82:~$ comm a1.txt a2.txt
        a
        b
                x
y
z
iem@iem-MS-7D82:~$
```

xiii. **diff :**

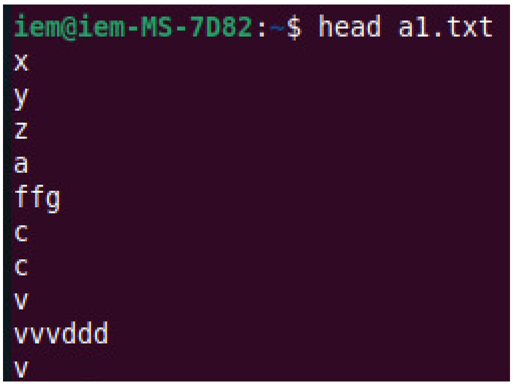
- **Description : Shows differences between two text files.**
- **Syntax :**
- **Output :**



```
iem@iem-MS-7D82:~$ diff a1.txt a2.txt
0a1,2
> a
> b
2,3d3
< y
< z
iem@iem-MS-7D82:~$
```

xiv. **head :**

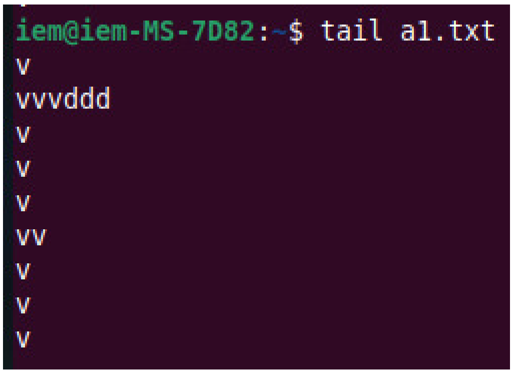
- **Description :** Shows the first 10 lines of a file by default.
- **Syntax :** $ head a1.txt
- **Output :**



```
iem@iem-MS-7D82:~$ head a1.txt
x
y
z
a
ffg
c
c
v
vvvddd
v
```

xv. **tail :**

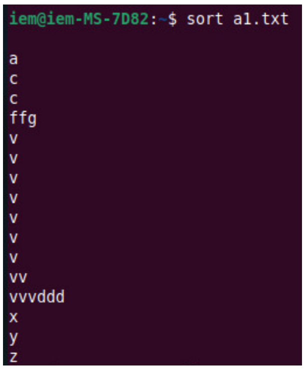
- **Description :** Shows the last 10 lines of a file by default.
- **Syntax :** $ tail a1.txt
- **Output :**



```
iem@iem-MS-7D82:~$ tail a1.txt
v
vvvddd
v
v
v
vv
v
v
v
```

xvi. **sort :**

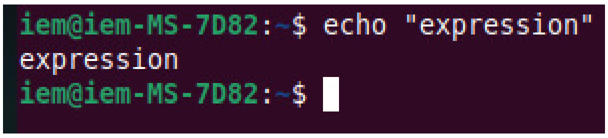
- **Description :** Sorts lines alphabetically or numerically.
- **Syntax :** $ sort a1.txt
- **Output :**



```
iem@iem-MS-7D82:~$ sort a1.txt

a
c
c
ffg
v
v
v
v
v
v
v
vv
vvvddd
x
y
z
```

xvii. **bc :**

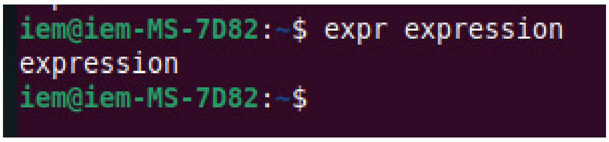
- **Description :** Command – line calculator.
- **Syntax :** $ echo “expression”
- **Output :**



```
iem@iem-MS-7D82:~$ echo "expression"
expression
iem@iem-MS-7D82:~$
```

xviii. **expr :**

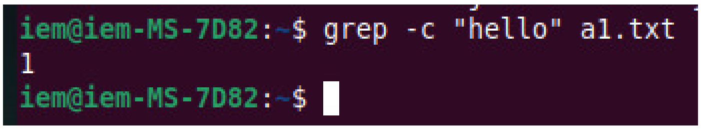
- **Description :** Evaluates expressions in shell.
- **Syntax :** $ expr expression
- **Output :**



```
iem@iem-MS-7D82:~$ expr expression
expression
iem@iem-MS-7D82:~$
```

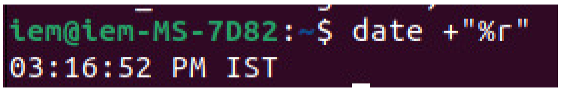
xix. **grep :**

- **Description :** Searches for patterns in files.
- **Syntax :** $ grep -c “hello” a1.txt
- **Output :**



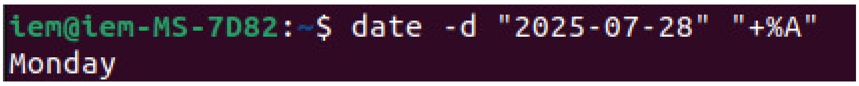
```
iem@iem-MS-7D82:~$ grep -c "hello" a1.txt
1
iem@iem-MS-7D82:~$
```

2. **Display the current time in 12 – hour format.**



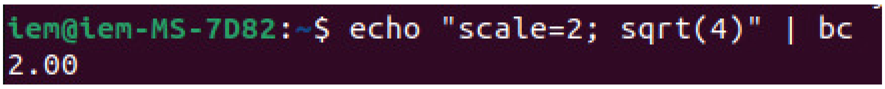
```
iem@iem-MS-7D82:~$ date +"%r"
03:16:52 PM IST
```

3. **With a user – specified date, display only the day of the week (e.g. Tuesday).**



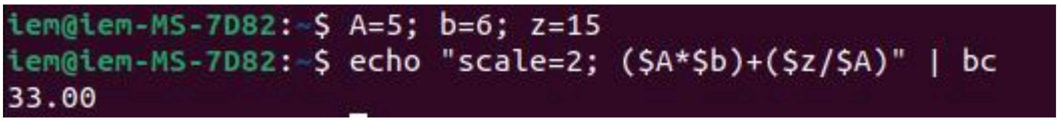
```
iem@iem-MS-7D82:~$ date -d "2025-07-28" "+%A"
Monday
```

4. **Write the command to find the square root of 4.**



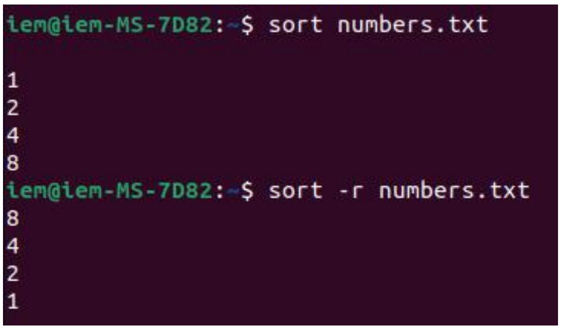
```
iem@iem-MS-7D82:~$ echo "scale=2; sqrt(4)" | bc
2.00
```

5. **Show how we can calculate the following expression in the terminal of UNIX where, A = 5, b = 6, z = 15 : Total = (A * b) + (z / A). Display the Total.**



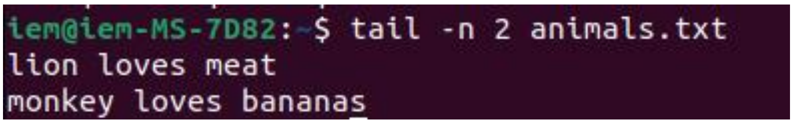
```
iem@iem-MS-7D82:~$ A=5; b=6; z=15
iem@iem-MS-7D82:~$ echo "scale=2; ($A*$b)+($z/$A)" | bc
33.00
```

6. **How can we sort a list of numbers in a file (both ascending and descending order)?**



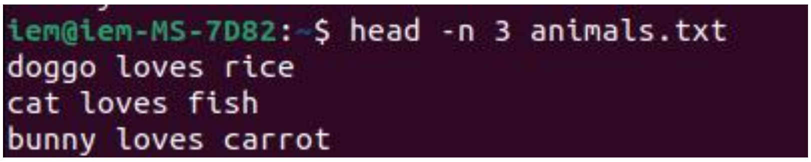
```
iem@iem-MS-7D82:~$ sort numbers.txt

1
2
4
8
iem@iem-MS-7D82:~$ sort -r numbers.txt
8
4
2
1
```

7. **Show the last 2 lines of the file animals.txt.**



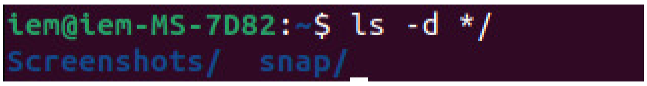
```
iem@iem-MS-7D82:~$ tail -n 2 animals.txt
lion loves meat
monkey loves bananas
```

8. **Show the first 3 lines of the file animals.txt.**



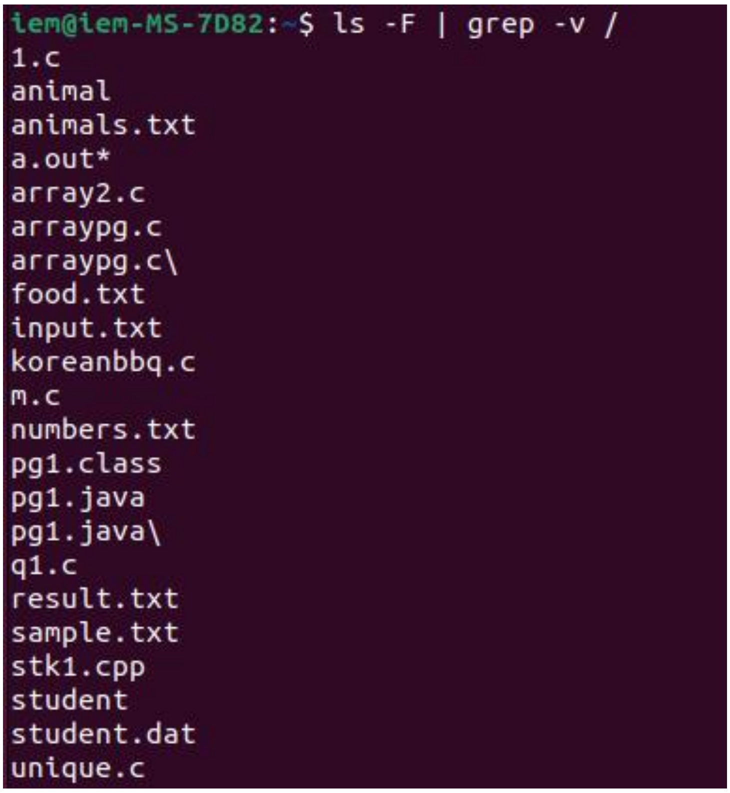
```
iem@iem-MS-7D82:~$ head -n 3 animals.txt
doggo loves rice
cat loves fish
bunny loves carrot
```

9. (Revisit) List only the directory files in your current directory.



```
iem@iem-MS-7D82:~$ ls -d */
Screenshots/  snap/
```

10. Count the number of directories in your current directory.



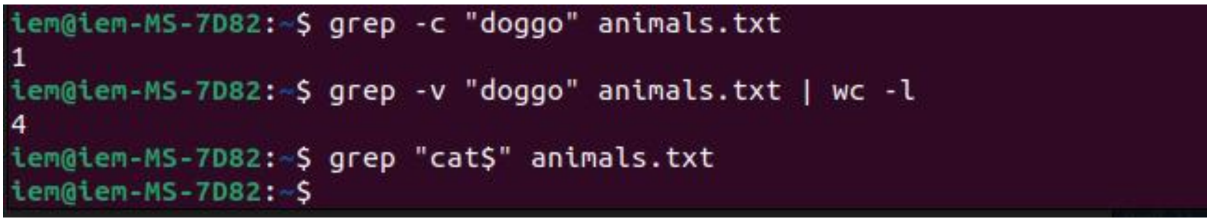
```
iem@iem-MS-7D82:~$ ls -F | grep -v /
1.c
animal
animals.txt
a.out*
array2.c
arraypg.c
arraypg.c\
food.txt
input.txt
koreanbbq.c
m.c
numbers.txt
pg1.class
pg1.java
pg1.java\
q1.c
result.txt
sample.txt
stk1.cpp
student
student.dat
unique.c
```

11. Dog is a domestic animal.
    Dog hates cat.
    Cat drinks milk.
    Dog is bigger than Cat.
    Cat is also a domestic animal.

a) Find the total number of lines contains the word ‘Dog’ in animals.txt.

b) Also find the total number of lines does not contain the word ‘Dog’ in animals.txt.

c) Display the lines in animals.txt that end with the word 'cat'.

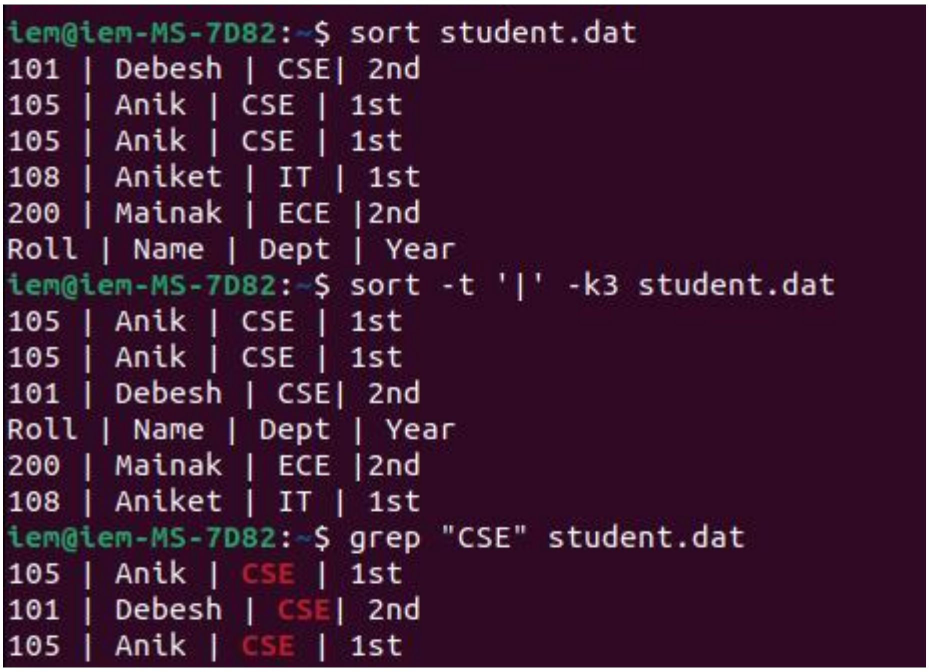


```
iem@iem-MS-7D82:~$ grep -c "doggo" animals.txt
1
iem@iem-MS-7D82:~$ grep -v "doggo" animals.txt | wc -l
4
iem@iem-MS-7D82:~$ grep "cat$" animals.txt
iem@iem-MS-7D82:~$
```

12. Create the file student.dat as follows :

Roll	Name	Dept.	Year
105	Anik	CSE	1st
101	Debesh	CSE	2nd
108	Aniket	IT	1st
200	Mainak	ECE	2nd
105	Anik	CSE	1st

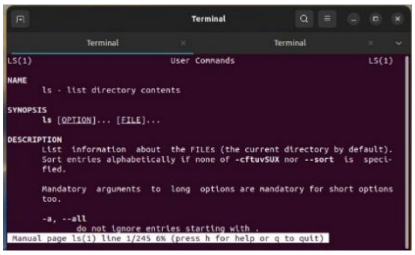
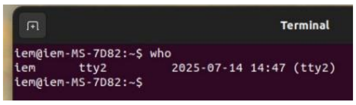
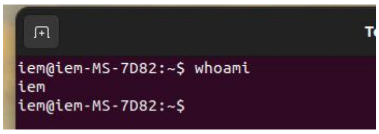
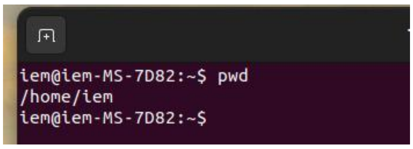
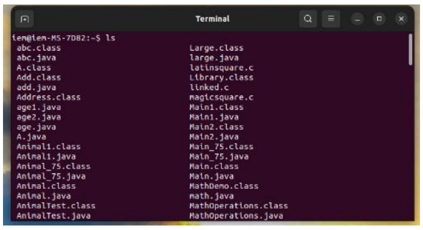
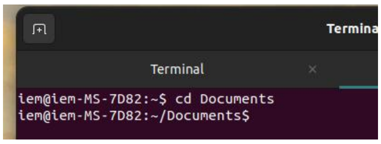
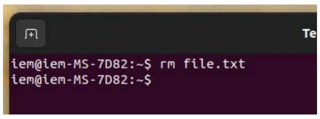
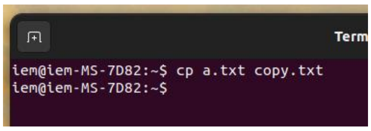
a) Sort the data according to Roll.
b) Sort the data according to Dept.
c) Show only the records of students from the CSE Dept.

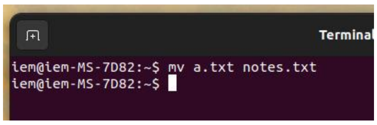
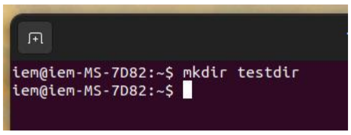
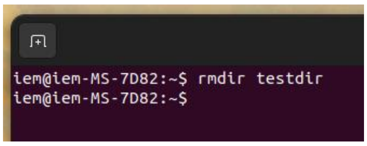
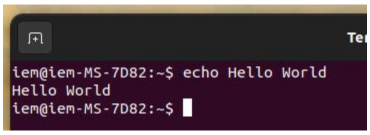
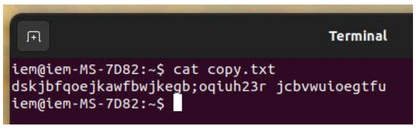
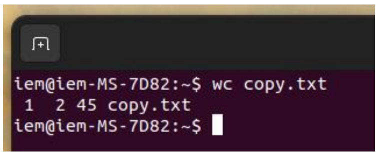


```
iem@iem-MS-7D82:~$ sort student.dat
101 | Debesh | CSE| 2nd
105 | Anik | CSE | 1st
105 | Anik | CSE | 1st
108 | Aniket | IT | 1st
200 | Mainak | ECE |2nd
Roll | Name | Dept | Year
iem@iem-MS-7D82:~$ sort -t '|' -k3 student.dat
105 | Anik | CSE | 1st
105 | Anik | CSE | 1st
101 | Debesh | CSE| 2nd
Roll | Name | Dept | Year
200 | Mainak | ECE |2nd
108 | Aniket | IT | 1st
iem@iem-MS-7D82:~$ grep "CSE" student.dat
105 | Anik | CSE | 1st
101 | Debesh | CSE| 2nd
105 | Anik | CSE | 1st
```

ASSIGNMENT 1

1. **For each basic UNIX command, give a brief description of what it does and how it can be used.**

Command	Description	Syntax	Sample Output
man	Displays the manual / help page for a given command	$ man 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 default). Sort entries alphabetically if none of -cftuvSUX nor --sort is specified. Mandatory arguments to long options are mandatory for short options too. -a, --all do not ignore entries starting with . Manual page ls(1) line 1/245 6% (press h for help or q to quit)
who	Shows all users currently using the system	$ who	 iem@iem-MS-7D82:~$ who iem tty2 2025-07-14 14:47 (tty2) iem@iem-MS-7D82:~$
whoami	Displays the current user's name	$ whoami	 iem@iem-MS-7D82:~$ whoami iem iem@iem-MS-7D82:~$
pwd	Prints the current working directory	$ pwd	 iem@iem-MS-7D82:~$ pwd /home/iem iem@iem-MS-7D82:~$
ls	Lists all files and directories in the current directory	$ ls	 iem@iem-MS-7D82:~$ ls abc.class Large.class abc.java large.java A.class latinsquare.c Add.class Library.class add.java linked.c Address.class magicsquare.c age1.java Main1.class age2.java Main1.java age.java Main2.class A.java Main2.java Animal1.class Main_75.class Animal1.java Main_75.java Animal_75.class Main.class Animal_75.java Main.java Animal.class MathDemo.class Animal.java math.java AnimalTest.class MathOperations.class AnimalTest.java MathOperations.java
cd	Changes the current directory	$ cd Documents	 iem@iem-MS-7D82:~$ cd Documents iem@iem-MS-7D82:~/Documents$
rm	Removes a file	$ rm.file1.txt	 iem@iem-MS-7D82:~$ rm file.txt iem@iem-MS-7D82:~$
cp	Copies files or directories	$ cp file1.txt copy.txt	 iem@iem-MS-7D82:~$ cp a.txt copy.txt iem@iem-MS-7D82:~$

mv	Moves or renames files	$ mv file1.txt notes.txt	 iem@iem-MS-7D82:~$ mv a.txt notes.txt iem@iem-MS-7D82:~$
mkdir	Creates a new directory	$ mkdir testdir	iem@iem-MS-7D82:~$ mkdir testdir iem@iem-MS-7D82:~$
rmdir	Removes an empty directory	$ rmdir testdir	iem@iem-MS-7D82:~$ rmdir testdir iem@iem-MS-7D82:~$
echo	Displays a line of text on the screen	$ echo Hello World	iem@iem-MS-7D82:~$ echo Hello World Hello World iem@iem-MS-7D82:~$
cat	Displays contents of a file	$ cat note1	 iem@iem-MS-7D82:~$ cat copy.txt dskjbfqoejkawfbwjkegb;oqiuh23r jcbvwuioegtfu iem@iem-MS-7D82:~$
wc	Counts lines, words, and characters in a file	$ wc note1	iem@iem-MS-7D82:~$ wc copy.txt 1 2 45 copy.txt iem@iem-MS-7D82:~$

2. Provide a short write – up (1 or 2 paragraphs) on the following :

A. History of Unix and Linux

Ken Thompson and Dennis Ritchie created UNIX at AT&T's Bell Labs in the late 1960s. It was a ground – breaking operating system with multiple users and multitasking capabilities that was made with efficiency and flexibility in mind. Linus Torvalds, however, presented Linux as a free and open – source substitute for UNIX in 1991. Since then, Linux has grown to be one of the most popular operating systems, powering devices ranging from servers to smartphones.

B. Kernel of an Operating System

An operating system's kernel is its central component. It controls memory, hardware communication, system calls, processes, and system resources. The kernel makes sure that software and hardware work together seamlessly. Different kernel types include hybrid, microkernel, and monolithic kernels.

C. Multi – Tasking OS

Multiple processes can operate concurrently on an operating system that supports multitasking. It makes it possible for the CPU to switch between tasks so fast that it looks like they are all running simultaneously. This is typical of contemporary operating systems like Linux, macOS, and Windows.

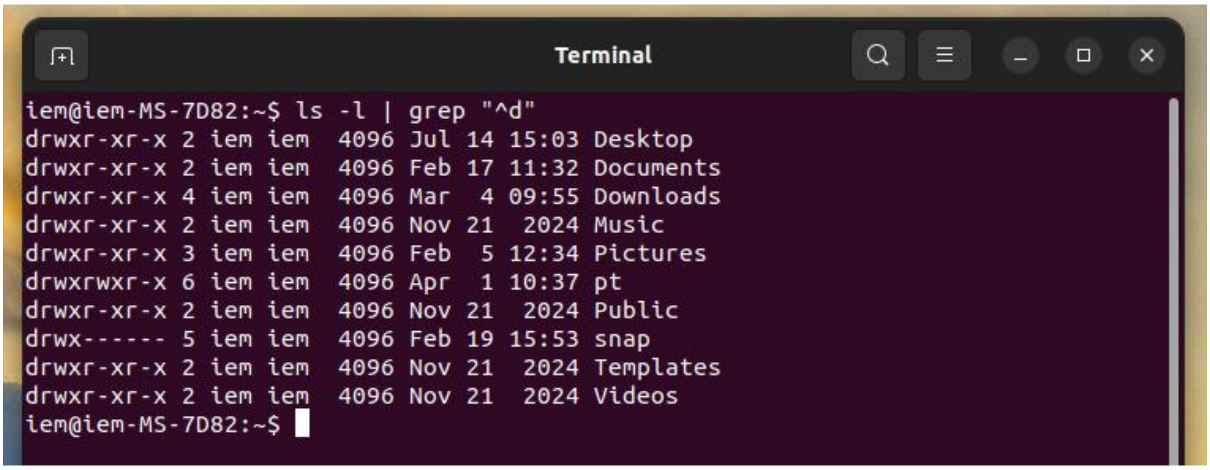
D. Multi – User OS

Multiple users can access a multi – user operating system at the same time. It guards against interference and guarantees that every user has a unique session. Two well – known multi – user operating systems that are utilized in servers and large systems are UNIX and Linux.

3. List only the directory files in your current directory.

Command : ls -l | grep "^d"

Output :

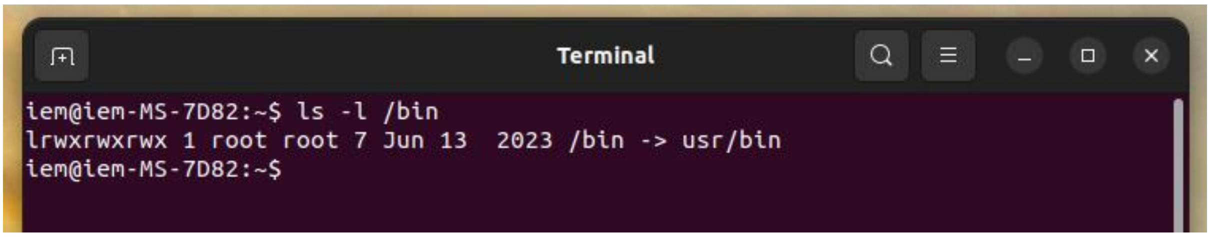


```
iem@iem-MS-7D82:~$ ls -l | grep "^d"
drwxr-xr-x 2 iem iem  4096 Jul 14 15:03 Desktop
drwxr-xr-x 2 iem iem  4096 Feb 17 11:32 Documents
drwxr-xr-x 4 iem iem  4096 Mar  4 09:55 Downloads
drwxr-xr-x 2 iem iem  4096 Nov 21  2024 Music
drwxr-xr-x 3 iem iem  4096 Feb  5 12:34 Pictures
drwxrwxr-x 6 iem iem  4096 Apr  1 10:37 pt
drwxr-xr-x 2 iem iem  4096 Nov 21  2024 Public
drwx------ 5 iem iem  4096 Feb 19 15:53 snap
drwxr-xr-x 2 iem iem  4096 Nov 21  2024 Templates
drwxr-xr-x 2 iem iem  4096 Nov 21  2024 Videos
iem@iem-MS-7D82:~$ 
```

4. **List all the files and directories of '/bin' with detail information from your current directory.**

Command : ls -l /bin

Output :

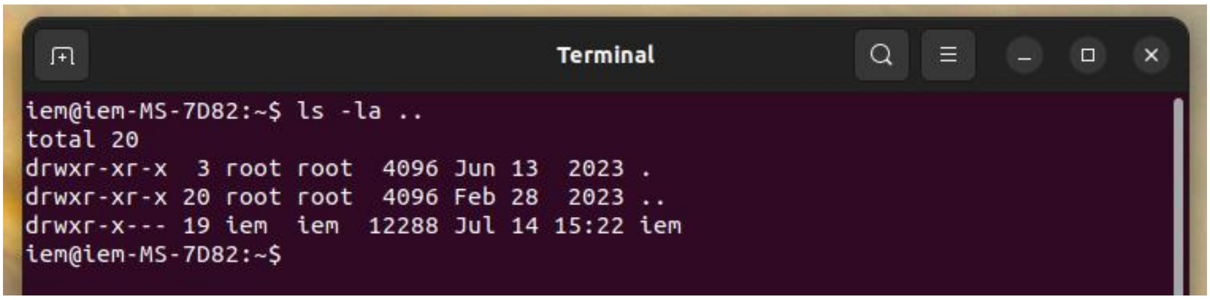


```
iem@iem-MS-7D82:~$ ls -l /bin
lrwxrwxrwx 1 root root 7 Jun 13  2023 /bin -> usr/bin
iem@iem-MS-7D82:~$
```

5. **List all the files including hidden files in your parent directory.**

Command : ls -la ..

Output :

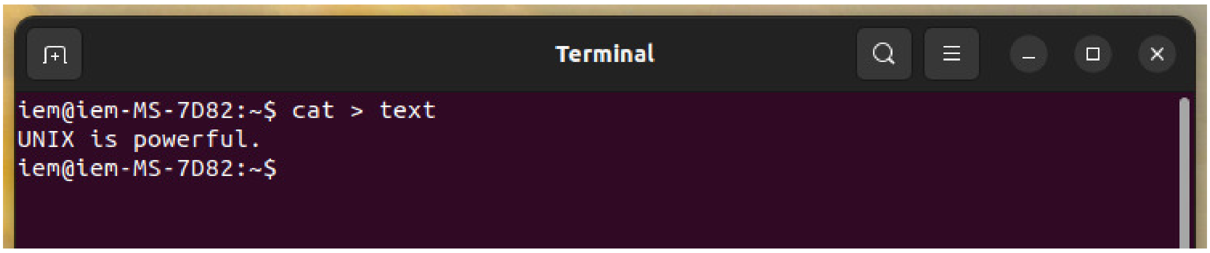


```
iem@iem-MS-7D82:~$ ls -la ..
total 20
drwxr-xr-x  3 root root  4096 Jun 13  2023 .
drwxr-xr-x 20 root root  4096 Feb 28  2023 ..
drwxr-x--- 19 iem  iem  12288 Jul 14 15:22 iem
iem@iem-MS-7D82:~$
```

6. **Create a file 'text 1' by taking input from the keyboard.**

Command : cat > text1

Output :

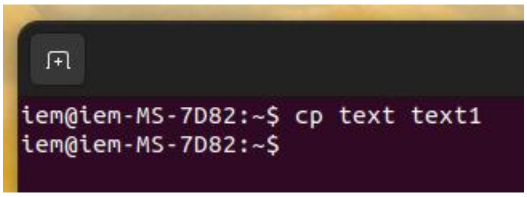


```
iem@iem-MS-7D82:~$ cat > text
UNIX is powerful.
iem@iem-MS-7D82:~$
```

7. Copy the contents of file' text1' to another file 'text2'.

Command : cp text1 text2

Output :

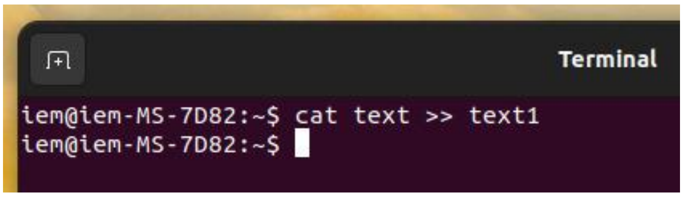


```
iem@iem-MS-7D82:~$ cp text text1
iem@iem-MS-7D82:~$
```

8. Append the contents of file 'text2 'to file 'text1'.

Command : cat text2 >> text1

Output :

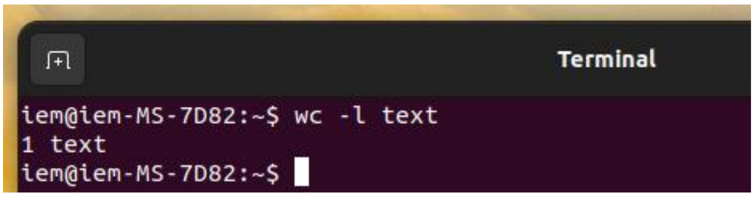


```
iem@iem-MS-7D82:~$ cat text >> text1
iem@iem-MS-7D82:~$
```

9. Count the number of lines in the file 'text1'.

Command : wc -l text1

Output :



```
iem@iem-MS-7D82:~$ wc -l text
1 text
iem@iem-MS-7D82:~$
```