FACULTY OF ENGINEERING AND TECHNOLOGY

BACHELOR OF TECHNOLOGY

Operating System Laboratory

(303105252)

4th Semester

Computer Science & Engineering Department

Laboratory Manual

CERTIFICATE

This is to certify that **Ms. PATEL KRISHNA JAYESH**

with enrolment no. ***2303031050428*** *has successfully*

completed his laboratory experiments in

Operating System Laboratory

during the academic year 2025/26.

Date: *Signature of lab teacher:*

Signature of HOD:

INDEX

SR. NO.	TITLE	PAGE NO.	PERFORMANCE DATE	ASSESSMENT DATE	MARKS	SIGN.
1	Study of Basic commands of Linux.					
2	Study the basics of shell programming.					
3	Write a Shell script to print the given number sum of all digits.					
4	Write a shell script to validate the entered date.					
5	Write a shell script to check entered string is palindrome or not.					
6	Write a Shell script to say good morning/Afternoon/Evening as you log in to system.					
7	Write a C program to create a child process.					
8	Finding out the biggest number from given three numbers supplied as command line arguments					
9	Printing the patterns using a for loop.					

10	Shell script to check whether a given file exists or not.					
11	Write a program for process creation using C.					
12	Implementation of FCFS &Round Robin Algorithm.					
13	Implementation of Banker's Algorithm.					

PRACTICAL NO: 1

Aim: Study of Basic commands of Linux.

1. pwd Command :

The pwd command is used to display the location of the current working directory.

2. mkdir Command :

The mkdir command is used to create a new directory under any directory.

3. rmdir Command :

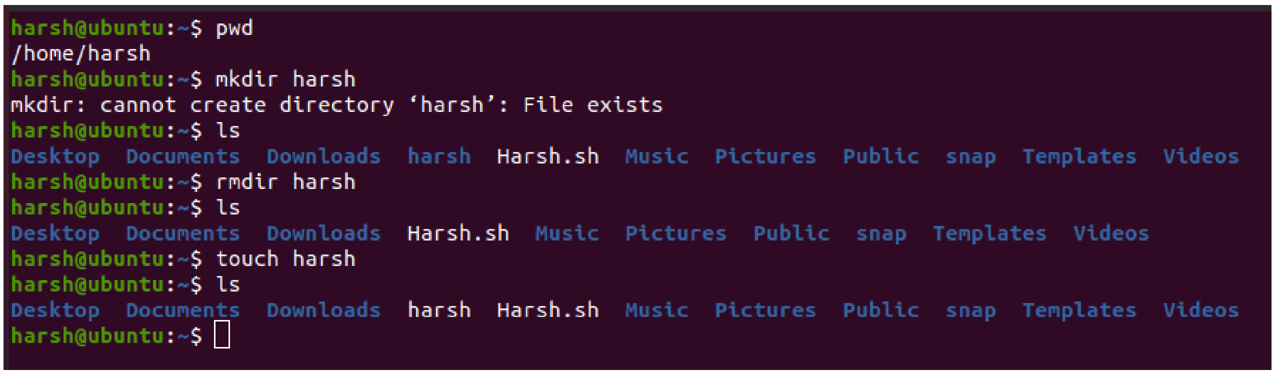
The rmdir command is used to delete a directory.

4. ls Command :

The ls command is used to display a list of content of a directory.

5. touch Command :

The touch command is used to create empty files. We can create multiple empty files by executing it once.



```
harsh@ubuntu:~$ pwd
/home/harsh
harsh@ubuntu:~$ mkdir harsh
mkdir: cannot create directory ‘harsh’: File exists
harsh@ubuntu:~$ ls
Desktop  Documents  Downloads  harsh  Harsh.sh  Music  Pictures  Public  snap  Templates  Videos
harsh@ubuntu:~$ rmdir harsh
harsh@ubuntu:~$ ls
Desktop  Documents  Downloads  Harsh.sh  Music  Pictures  Public  snap  Templates  Videos
harsh@ubuntu:~$ touch harsh
harsh@ubuntu:~$ ls
Desktop  Documents  Downloads  harsh  Harsh.sh  Music  Pictures  Public  snap  Templates  Videos
harsh@ubuntu:~$
```

6. cd Command :

The cd command is used to change the current directory.

7. cat Command :

The cat command is a multi-purpose utility in the Linux system. It can be used to create a file, display content of the file, copy the content of one file to another file, and more.

8. rm Command :

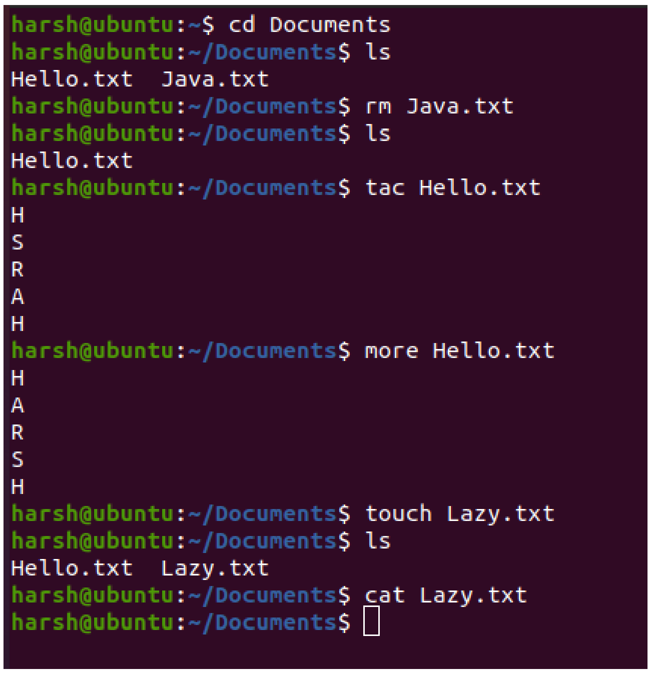
The rm command is used to remove a file.

9. tac Command :

The tac command is the reverse of cat command, as its name specified. It displays the file content in reverse order (from the last line).

10. more command :

The more command is quite similar to the cat command, as it is used to display the file content in the same way that the cat command does. The only difference between both commands is that, in case of larger files, the more command displays screenful output at a time.



```
harsh@ubuntu:~$ cd Documents
harsh@ubuntu:~/Documents$ ls
Hello.txt  Java.txt
harsh@ubuntu:~/Documents$ rm Java.txt
harsh@ubuntu:~/Documents$ ls
Hello.txt
harsh@ubuntu:~/Documents$ tac Hello.txt
H
S
R
A
H
harsh@ubuntu:~/Documents$ more Hello.txt
H
A
R
S
H
harsh@ubuntu:~/Documents$ touch Lazy.txt
harsh@ubuntu:~/Documents$ ls
Hello.txt  Lazy.txt
harsh@ubuntu:~/Documents$ cat Lazy.txt
harsh@ubuntu:~/Documents$ 
```

11. head Command :

The head command is used to display the content of a file. It displays the first 10 lines of a file .

12. tail Command :

The tail command is similar to the head command. The difference between both commands is that it displays the last ten lines of the file content. It is useful for reading the error message.

13. passwd Command :

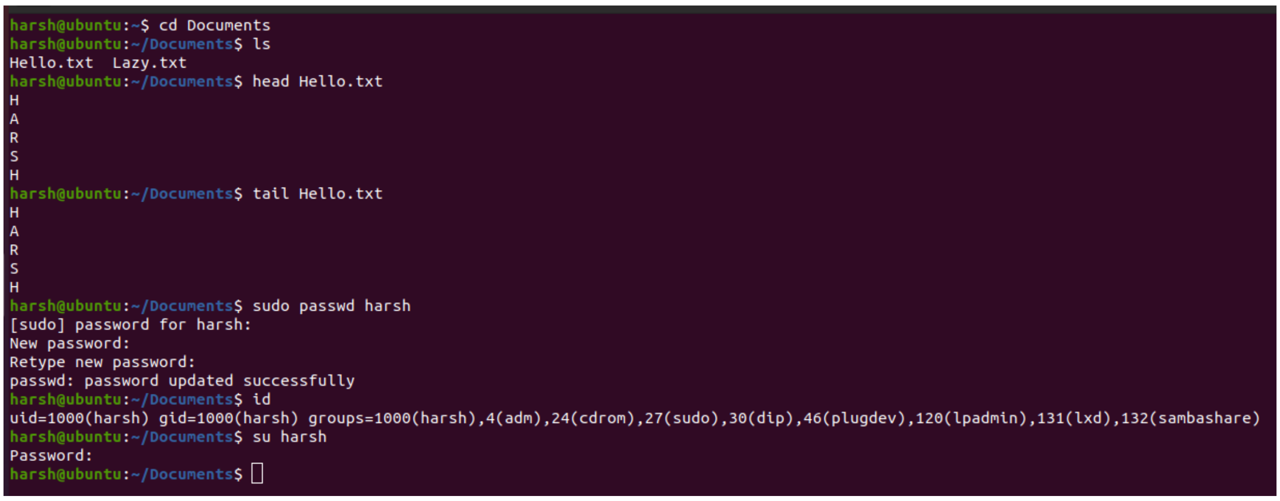
The passwd command is used to create and change the password for a user.

14. id Command :

The id command is used to display the user ID (UID) and group ID (GID).

15. su Command :

The su command provides administrative access to another user. In other words, it allows access of the Linux shell to another user.



```
harsh@ubuntu:~$ cd Documents
harsh@ubuntu:~/Documents$ ls
Hello.txt  Lazy.txt
harsh@ubuntu:~/Documents$ head Hello.txt
H
A
R
S
H
harsh@ubuntu:~/Documents$ tail Hello.txt
H
A
R
S
H
harsh@ubuntu:~/Documents$ sudo passwd harsh
[sudo] password for harsh:
New password:
Retype new password:
passwd: password updated successfully
harsh@ubuntu:~/Documents$ id
uid=1000(harsh) gid=1000(harsh) groups=1000(harsh),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev),120(lpadmin),131(lxd),132(sambashare)
harsh@ubuntu:~/Documents$ su harsh
Password:
harsh@ubuntu:~/Documents$
```

16. less Command :

The less command is similar to the more command. It also includes some extra features such as 'adjustment in width and height of the terminal.' Comparatively, the more command cuts the output in the width of the terminal.

17. cp Command :

The cp command is used to copy a file or directory.

18. wc Command :

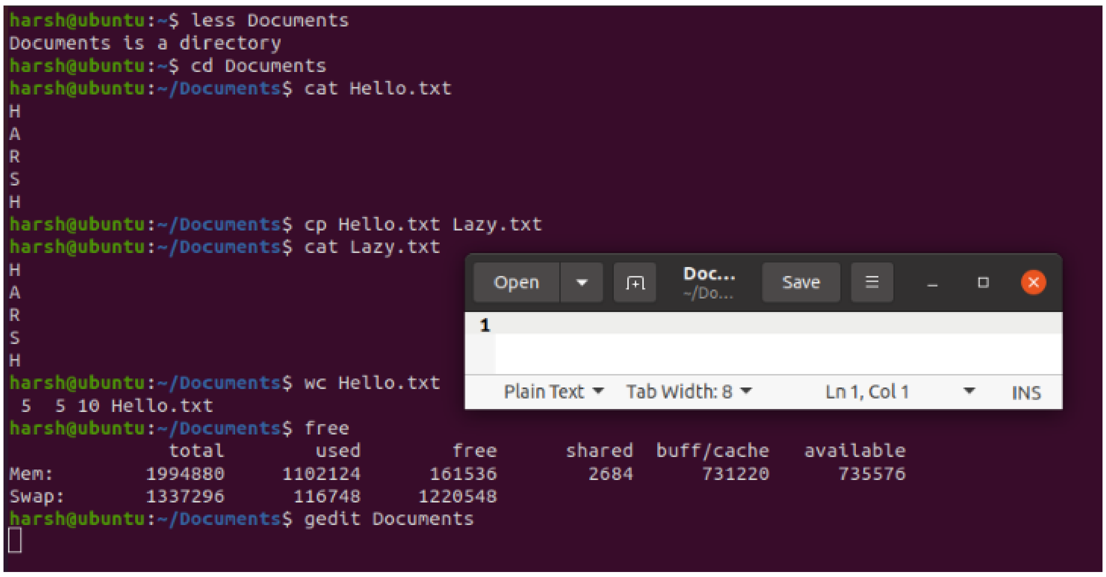
The wc (word count) command, can return the number of lines, words, and characters in a file.

19. free Command :

The free command Display RAM details in Linux machine.

20. gedit Command :

The gedit command is used to create and open a file.



```
harsh@ubuntu:~$ less Documents
Documents is a directory
harsh@ubuntu:~$ cd Documents
harsh@ubuntu:~/Documents$ cat Hello.txt
H
A
R
S
H
harsh@ubuntu:~/Documents$ cp Hello.txt Lazy.txt
harsh@ubuntu:~/Documents$ cat Lazy.txt
H
A
R
S
H
harsh@ubuntu:~/Documents$ wc Hello.txt
 5  5 10 Hello.txt
harsh@ubuntu:~/Documents$ free
              total        used        free      shared  buff/cache   available
Mem:        1994880     1102124      161536        2684      731220      735576
Swap:       1337296      116748     1220548
harsh@ubuntu:~/Documents$ gedit Documents
```

PRACTICAL NO: 2

Aim: Study the basics of shell programming.

- **WHAT IS A SHELL?**

➢ An Operating is made of many components, but its two prime components.

1. Kernel
2. Shell

➢ A Kernel is at the nucleus of a computer. It makes the communication between the hardware and software possible. While the Kernel is the innermost part of an operating system, a shell is the outermost one.

➢ A shell in a Linux operating system takes input from you in the form of commands, processes it, and then gives an output. It is the interface through which a user works on the programs, commands, and scripts. A shell is accessed by a terminal which runs it.

➢ When you run the terminal, the Shell issues a command prompt (usually $), where you can type your input, which is then executed when you hit the Enter key. The output or the result is thereafter displayed on the terminal.

- The Shell wraps around the delicate interior of an Operating system protecting it from accidental damage. Hence the name Shell.

- **TYPES OF SHELL:**
  1. Bourne Shell
  2. C shell
  3. Korn Shell
  4. GNU Bourne-Again Shell

- **WHAT IS SHELL SCRIPTING?**
- Shell scripting is writing a series of command for the shell to execute. It can combine lengthy and repetitive sequences of commands into a single and simple script, which can be stored and executed anytime. This reduces the effort required by the end user.

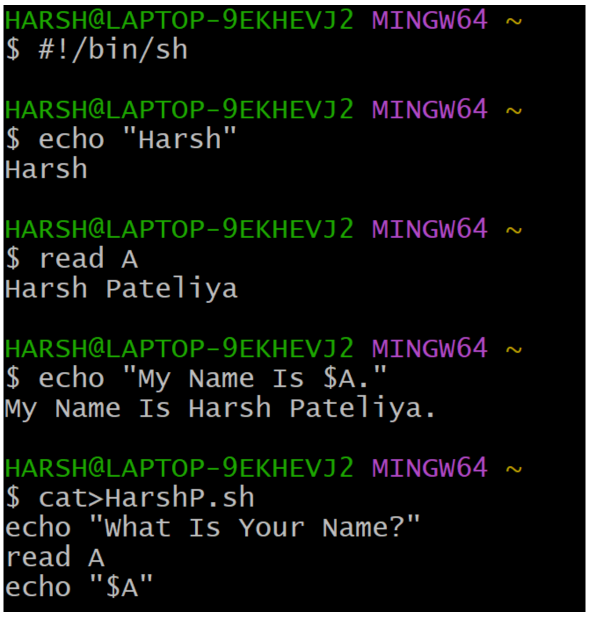
- **LET US UNDERSTAND THE STEPS IN CREATING A SHELL SCRIPT:**
  1. Create a file using a vi editor (or any other editor). Name script file with extension .sh
  2. Start the script with #! /bin/sh
  3. Write some code.
  4. Save the script file as filename.sh
  5. For executing the script type bash filename.sh

- "#!" is an operator called shebang which directs the script to the interpreter location. So, if we use"#!/bin/sh" the script gets directed to the bourne-shell.
- Echo function is use for output the statement.
- Read function use for input the data.

- There is 3 types to create file.
  1. Touch command
  2. Cat Command
  3. Text editor

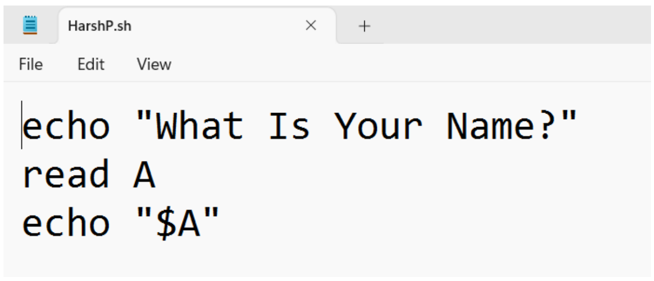
SHELL SCRIPT EXAMPLE:

- **GIT BASH SCRIPT:**



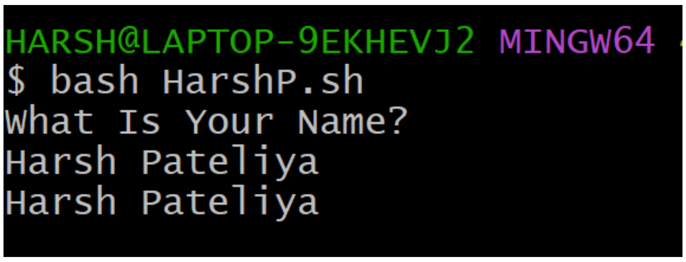
```
HARSH@LAPTOP-9EKHEVJ2 MINGW64 ~
$ #!/bin/sh

HARSH@LAPTOP-9EKHEVJ2 MINGW64 ~
$ echo "Harsh"
Harsh

HARSH@LAPTOP-9EKHEVJ2 MINGW64 ~
$ read A
Harsh Pateliya

HARSH@LAPTOP-9EKHEVJ2 MINGW64 ~
$ echo "My Name Is $A."
My Name Is Harsh Pateliya.

HARSH@LAPTOP-9EKHEVJ2 MINGW64 ~
$ cat>HarshP.sh
echo "What Is Your Name?"
read A
echo "$A"
```

- **TEXT EDITOR:**



```
echo "What Is Your Name?"
read A
echo "$A"
```

- **OUTPUT:**



```
HARSH@LAPTOP-9EKHEVJ2 MINGW64
$ bash HarshP.sh
What Is Your Name?
Harsh Pateliya
Harsh Pateliya
```

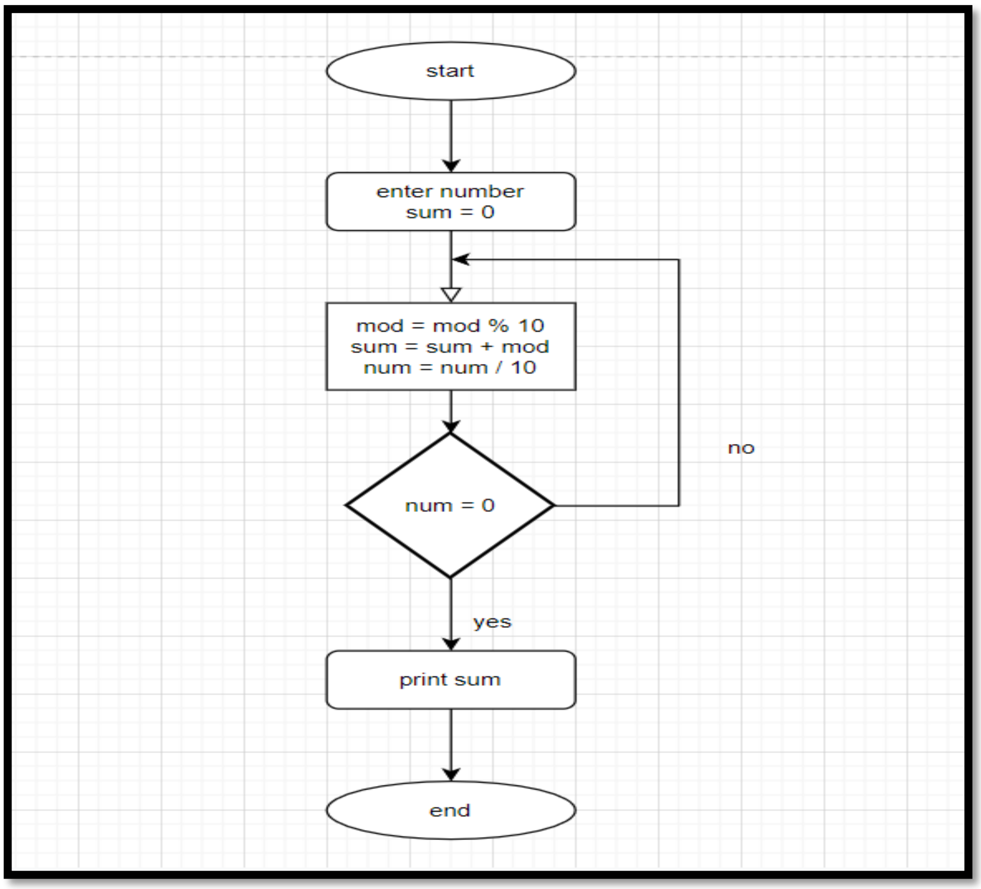
PRACTICAL NO: 3

Aim: Write a Shell script to print the given number sum of all digits.

ALGORITHM:

1. Get a number.
2. Split each digit from the number using the modulo operator.
3. Calculate the sum.
4. Print the result.

FLOWCHART:

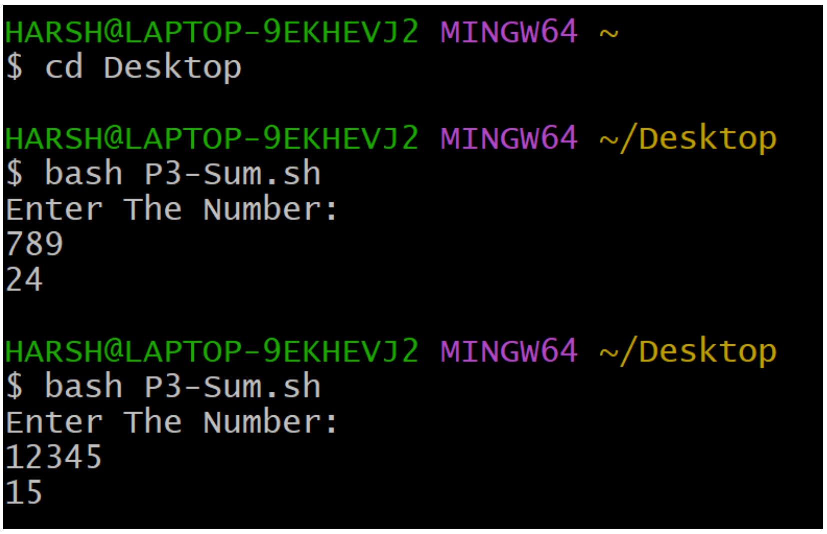


INPUT:

```
#sum of all digits - shell script
echo "Enter The Number:"
read num
sum=0
while [ $num -gt 0 ]
do
   mod=`expr $num % 10`
   sum=`expr $sum + $mod`
   num=`expr $num / 10`
```

```
done

echo $sum
```

OUTPUT :



```
HARSH@LAPTOP-9EKHEVJ2 MINGW64 ~
$ cd Desktop

HARSH@LAPTOP-9EKHEVJ2 MINGW64 ~/Desktop
$ bash P3-Sum.sh
Enter The Number:
789
24

HARSH@LAPTOP-9EKHEVJ2 MINGW64 ~/Desktop
$ bash P3-Sum.sh
Enter The Number:
12345
15
```

PRACTICAL NO: 4

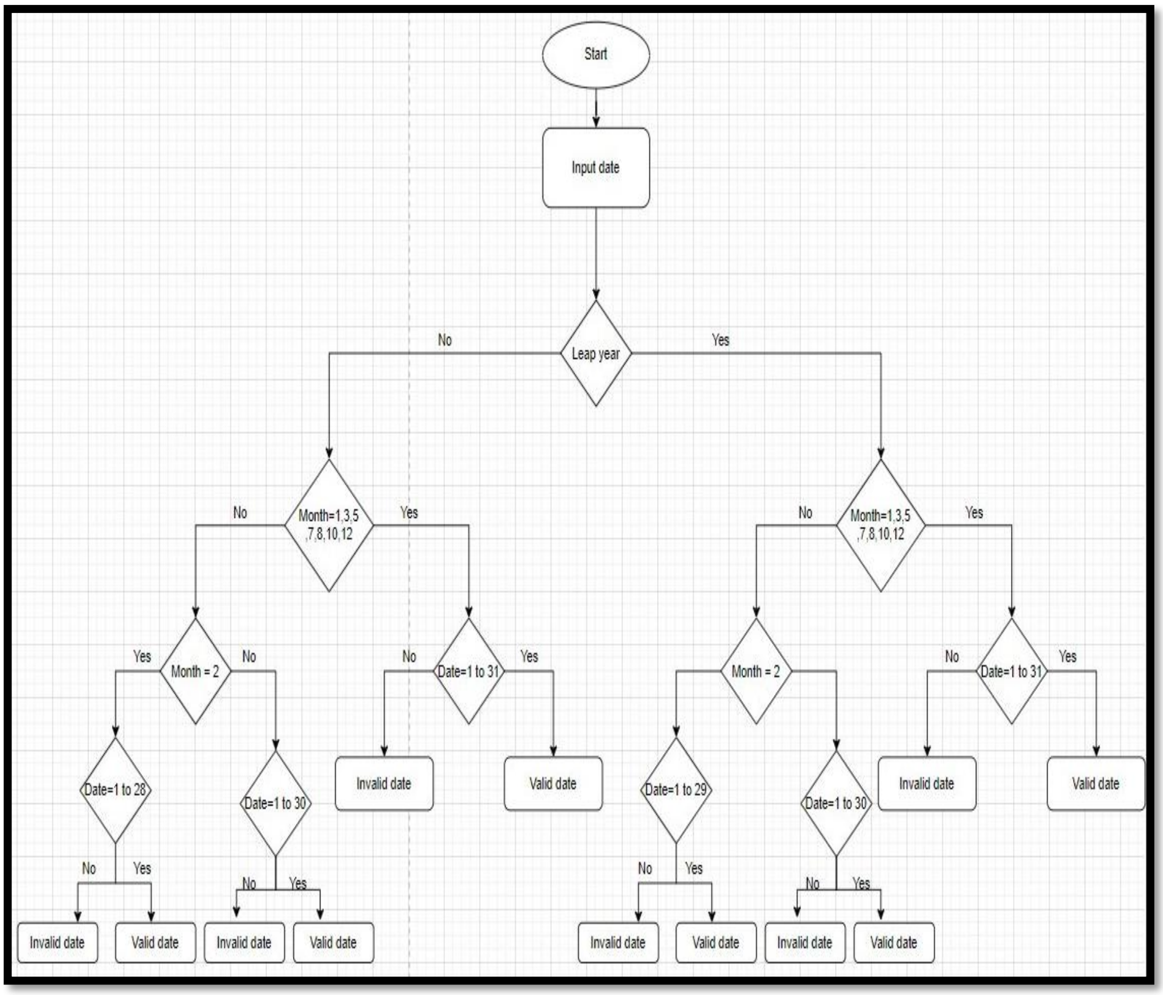
Aim: Write a shell script to validate the entered date.

ALGORITHM:

1. Start.
2. Input a date.
3. Check year is leap or not.
4. Check month is (m=1,3,5,7,8,10,12) then date is valid from 1 to 31.
5. Check month is (m=4,6,9,11) then date is valid from 1 to 30.
6. Check month is (m=2) and year is leap year then date is valid from 1 to 29.

7. Check month is (m=2) and not leap year then date is valid from 1 to 28.
8. Else date is invalid.
9. End.

FLOWCHART:



INPUT :

dd=0

mm=0

yy=0

days=0

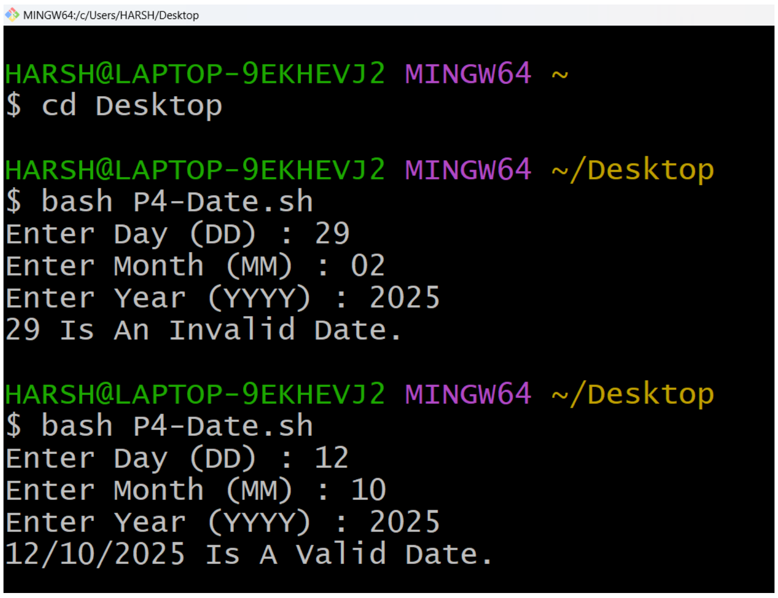
echo -n "Enter Day (DD) : "

read dd

```
echo -n "Enter Month (MM) : "
read mm
echo -n "Enter Year (YYYY) : "
read yy
if [ $mm -le 0 -o $mm -gt 12 ];
then
 echo "$mm is invalid month."
  exit 1
  fih
case $mm in
    1) days=31;;
    2) days=28;;
    3) days=31;;
    4) days=30;;
    5) days=31;;
    6) days=30;;
    7) days=31;;
    8) days=31;;
    9) days=30;;
    10) days=31;;
    11) days=30;;
    12) days=31;;
    *) days=-1;;
esac
if [ $mm -eq 2 ]; # if it is feb month then only check of leap year
```

```
then
if [ $((yy % 4)) -ne 0 ] ; then
 :
elif [ $((yy % 400)) -eq 0 ] ; then
days=29
elif [ $((yy % 100)) -eq 0 ] ; then
  :
else
 days=29
fi
fi
 if [ $dd -le 0 -o $dd -gt $days ];
then
 echo "$dd Is An Invalid Date."
  exit 3
fi
echo "$dd/$mm/$yy Is A Valid Date."
```

OUTPUT:



```
HARSH@LAPTOP-9EKHEVJ2 MINGW64 ~
$ cd Desktop

HARSH@LAPTOP-9EKHEVJ2 MINGW64 ~/Desktop
$ bash P4-Date.sh
Enter Day (DD) : 29
Enter Month (MM) : 02
Enter Year (YYYY) : 2025
29 Is An Invalid Date.

HARSH@LAPTOP-9EKHEVJ2 MINGW64 ~/Desktop
$ bash P4-Date.sh
Enter Day (DD) : 12
Enter Month (MM) : 10
Enter Year (YYYY) : 2025
12/10/2025 Is A Valid Date.
```

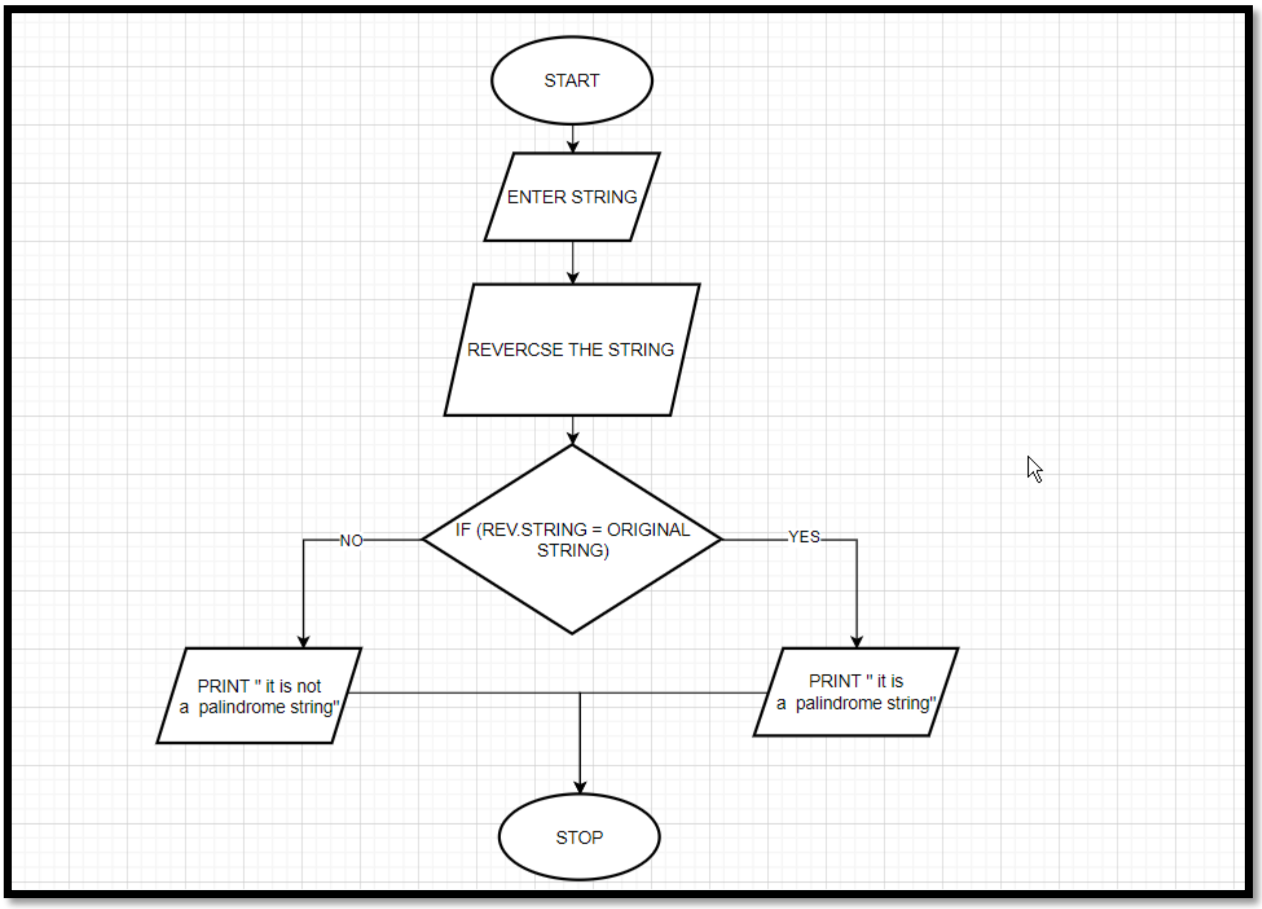
PRACTICAL NO: 5

Aim: Write a shell script to check entered string is palindrome or not.

ALGORITHM:

1. Start.
2. Get the string from user.
3. Hold the string in temporary variable.
4. Reverse the string.
5. Compare the temporary string with reversed string.
6. Both strings are same, then print "The string is a palindrome".
7. Else print "The string is not a palindrome".
8. End.

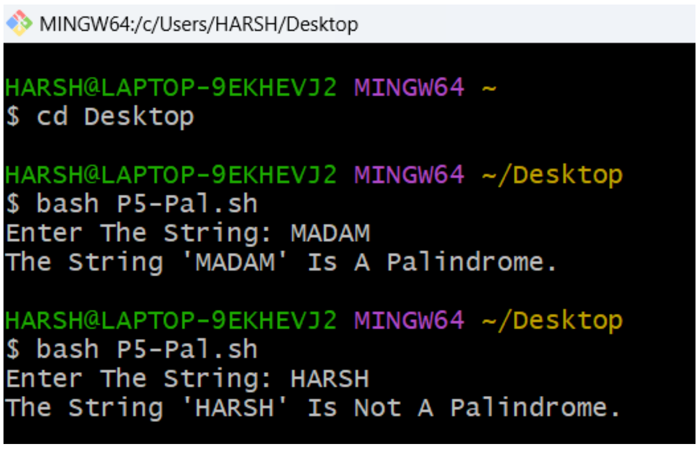
FLOWCHART:


START
ENTER STRING
REVERCSE THE STRING
IF (REV.STRING = ORIGINAL STRING)
NO
YES
PRINT " it is not a palindrome string"
PRINT " it is a palindrome string"
STOP

INPUT :

```
check_palindrome()
{
    local str="$1"
    local len=${#str}
    local reversed=""
    for (( i=$len-1; i>=0; i-- )); do
        reversed="$reversed${str:$i:1}"
    done
    if [ "$str" == "$reversed" ]; then
        echo "The String '$str' Is A Palindrome."
    else
        echo "The String '$str' Is Not A Palindrome."
    fi
}
echo -n "Enter The String: "
read input_string
check_palindrome "$input_string"
```

OUTPUT:


MINGW64:/c/Users/HARSH/Desktop

```
HARSH@LAPTOP-9EKHEVJ2 MINGW64 ~
$ cd Desktop

HARSH@LAPTOP-9EKHEVJ2 MINGW64 ~/Desktop
$ bash P5-Pal.sh
Enter The String: MADAM
The String 'MADAM' Is A Palindrome.

HARSH@LAPTOP-9EKHEVJ2 MINGW64 ~/Desktop
$ bash P5-Pal.sh
Enter The String: HARSH
The String 'HARSH' Is Not A Palindrome.
```

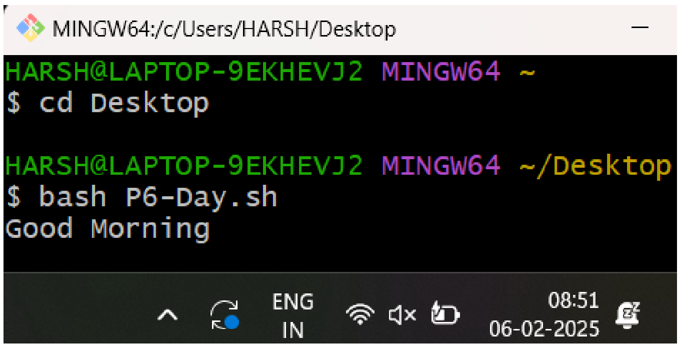
PRACTICAL NO: 6

Aim: Write a Shell script to say Good Morning/ Afternoon /Evening / Night as you log in to system.

INPUT :

```
h=$(date +"%H")
if [ $h -gt 6 -a $h -le 12 ]
then
echo Good Morning
elif [ $h -gt 12 -a $h -le 16 ]
then
echo Good Afternoon
elif [ $h -gt 16 -a $h -le 20 ]
then
echo Good Evening
else
echo Good Night
fi
```

OUTPUT:



```
MINGW64:/c/Users/HARSH/Desktop
HARSH@LAPTOP-9EKHEVJ2 MINGW64 ~
$ cd Desktop

HARSH@LAPTOP-9EKHEVJ2 MINGW64 ~/Desktop
$ bash P6-Day.sh
Good Morning
```

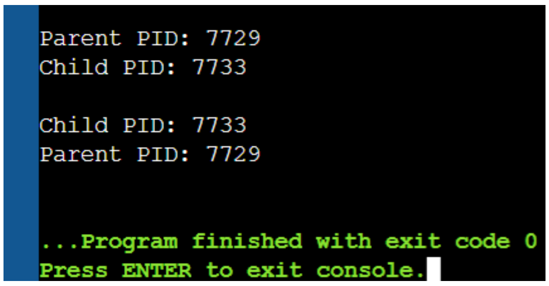
PRACTICAL NO: 7

Aim: Write a C program to create a child process.

INPUT :

```
#include <sys/types.h>
#include <stdio.h>
int main()
{
    pid_t pid = fork();  // Create a child process
    if (pid < 0)
    {
        // Fork failed
        perror("Fork failed");
        return 1;
    }
    if (pid == 0)
    {
        // This is the child process
        printf("\nChild PID: %d\n", getpid());
        printf("Parent PID: %d\n", getppid());
    }
    else
    {
        // This is the parent process
        printf("\nParent PID: %d\n", getpid());
        printf("Child PID: %d\n", pid);
    }
    return 0;
}
```

OUTPUT:



```
Parent PID: 7729
Child PID: 7733

Child PID: 7733
Parent PID: 7729


...Program finished with exit code 0
Press ENTER to exit console.
```

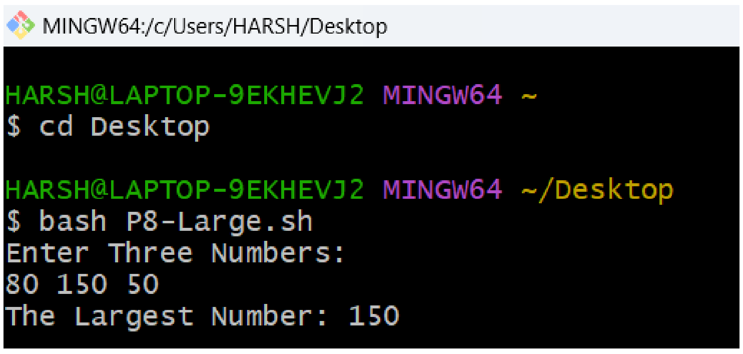
PRACTICAL NO: 8

Aim: Find out the biggest number from the given three numbers supplied as command line arguments.

INPUT :

```
echo "Enter Three Numbers:"
read num1 num2 num3

if [[ $num1 -ge $num2 && $num1 -ge $num3 ]];
then
   echo "The Largest Number: $num1"
elif [[ $num2 -ge $num1 && $num2 -ge $num3 ]];
then
   echo "The Largest Number: $num2"
else
   echo "The Largest Number: $num3"
fi
```

OUTPUT:



MINGW64:/c/Users/HARSH/Desktop

```
HARSH@LAPTOP-9EKHEVJ2 MINGW64 ~
$ cd Desktop

HARSH@LAPTOP-9EKHEVJ2 MINGW64 ~/Desktop
$ bash P8-Large.sh
Enter Three Numbers:
80 150 50
The Largest Number: 150
```