

Name: **Tushar Nankani**

Roll No: **1902112**

Batch: **C23**

Operating System: Assignment 2

Problem Statement: Shell Programming

THEORY:

1. What is a Shell?

A Shell provides you with an interface to the Unix system. It gathers input from you and executes programs based on that input. When a program finishes executing, it displays that program's output.

Shell is an environment in which we can run our commands, programs, and shell scripts. There are different flavors of a shell, just as there are different flavors of operating systems. Each flavor of shell has its own set of recognized commands and functions.

2. What is a Shell Prompt?

The prompt, \$, which is called the command prompt, is issued by the shell. While the prompt is displayed, you can type a command.

Shell reads your input after you press Enter. It determines the command you want executed by looking at the first word of your input. A word is an unbroken set of characters. Spaces and tabs separate words.

3. What is a Shell Script?

Shell Scripts are written using text editors. On your Linux system, open a text editor program, open a new file to begin typing a shell script or shell programming, then give the shell permission to execute your shell script and put your script at the location from where the shell can find it.

How to write a Shell Script?

- Name script file with extension .sh
- Start the script with #! /bin/sh
- Write some code.
- Save the script file as filename.sh
- For executing the script type bash filename.sh

VARIABLES: Variables store data in the form of characters and numbers.

Similarly, Shell variables are used to store information and they can be used by the shell only.

```
variable ="Hello"  
echo $variable
```

The above scripts print **Hello**.

```
echo "what is your name?"  
read name  
echo "I am $name!"
```

This script takes in input as your name and outputs the line.

Operators:

Operator	Description
+ (Addition)	Adds values on either side of the operator
- (Subtraction)	Subtracts right hand operand from left hand operand
* (Multiplication)	Multiplies values on either side of the operator
/ (Division)	Divides left hand operand by right hand operand
% (Modulus)	Divides left hand operand by right hand operand and returns remainder
= (Assignment)	Assigns right operand in left operand
== (Equality)	Compares two numbers, if both are same then returns true.
!= (Not Equality)	Compares two numbers, if both are different then returns true.

RELATIONAL OPERATORS:

Operator	Description
-eq	Checks if the value of two operands are equal or not; if yes, then the condition becomes true.
-ne	Checks if the value of two operands are equal or not; if values are not equal, then the condition becomes true.
-gt	Checks if the value of left operand is greater than the value of right operand; if yes, then the condition becomes true.
-lt	Checks if the value of left operand is less than the value of right operand; if yes, then the condition becomes true.
-ge	Checks if the value of left operand is greater than or equal to the value of right operand; if yes, then the condition becomes true.
-le	Checks if the value of left operand is less than or equal to the value of right operand; if yes, then the condition becomes true.

Shell Decision Making:

If else statements are useful decision-making statements which can be used to select an option from a given set of options.

Unix Shell supports following forms of if...else statement –

- if...fi statement
- if...else...fi statement
- if...elif...else...fi statement

It is very important to understand that all the conditional expressions should be placed inside square braces with spaces around them. For example, [\$a <= \$b] is correct whereas, [\$a <= \$b] is incorrect.

For example, here is a small script to check whether the even or not.

```
read -p "Enter a number: " num
if [ $((num%2)) -eq 1 ]
then
    echo "$num is odd."
else
    echo "$num is even."
fi
```

Loops:

All the loops support nesting concept which means you can put one loop inside another similar one or different loops. This nesting can go up to unlimited number of times based on your requirement.

Their descriptions and syntax are as follows:

while statement

Syntax

```
while command
do
    Statement to be executed;
Done
```

for statement

The for loop operate on lists of items. It repeats a set of commands for every item in a list.

Here var is the name of a variable and word1 to wordN are sequences of characters separated by spaces (words).

Syntax

```
for var in word1 word2 ...wordn
do
    Statement to be executed
Done
```

Q1. Write a shell program to Add 2 numbers

```
read -p "Enter the first number: " a
read -p "Enter the second number: " b
echo "The sum of $a and $b is $((a+b))"
```

```
Tushar Nankani@LAPTOP-SAVOIT3Q MINGW64 ~/Desktop/Git/My Repositories/College/sem
4-assignments/Operating System (OS) (main)
$ bash 01_Add2.sh
Enter the first number: 4
Enter the second number: 5
The sum of 4 and 5 is 9

Tushar Nankani@LAPTOP-SAVOIT3Q MINGW64 ~/Desktop/Git/My Repositories/College/sem
4-assignments/Operating System (OS) (main)
$ bash 01_Add2.sh
Enter the first number: 5
Enter the second number: 8
The sum of 5 and 8 is 13
```

Q2. Write a shell program shell program to check if a number entered is even and odd.

```
read -p "Enter a number: " num
if [ $((num%2)) -eq 1 ]
then
    echo "$num is odd."
else
    echo "$num is even."
fi
```

```
Tushar Nankani@LAPTOP-SAVOIT3Q MINGW64
$ bash 02_EvenOdd.sh
Enter a number: 7
7 is odd.

Tushar Nankani@LAPTOP-SAVOIT3Q MINGW64
$ bash 02_EvenOdd.sh
Enter a number: 9
9 is odd.
```

Q3. Write a shell program to find sum of n numbers.

```
read -p "Enter a number: " num
sum=0
echo "Enter $num numbers: "
for((i=0; i<$num; i++))
do
    read -p "Enter number $((i+1)): " a
    sum=$((sum+a))
done
echo "The sum of the entered numbers is $sum"
```

```
Tushar Nankani@LAPTOP-SAVOIT3Q MINGW64 ~/Desktop
4-assignments/Operating System (OS) (main)
$ bash 03_Sum.sh
Enter a number: 5
Enter 5 numbers:
Enter number 1: 8
Enter number 2: 8
Enter number 3: 9
Enter number 4: 4
Enter number 5: 2
The sum of the entered numbers is 31
```

Q4. Write a shell program to determine if a person is eligible to vote or not.

```
read -p "Enter the age of the candidate: " age
if (($age >= 18));
then
    echo "The candidate is eligible for voting."
else
    echo "The candidate is not eligible for voting."
fi
```

```
Tushar Nankani@LAPTOP-SAVOIT3Q MINGW64 ~/Desktop,
4-assignments/Operating System (OS) (main)
$ bash 04_EligiblityForVoting.sh
Enter the age of the candidate: 18
The candidate is eligible for voting.

Tushar Nankani@LAPTOP-SAVOIT3Q MINGW64 ~/Desktop,
4-assignments/Operating System (OS) (main)
$ bash 04_EligiblityForVoting.sh
Enter the age of the candidate: 15
The candidate is not eligible for voting.
```

Q5. Write a shell program to display all filenames beginning with character 'a' and displays its contents.

```
for file in a*
do
    echo -e "\nIn $file:"
    cat $file
done
```

```
Tushar Nankani@LAPTOP-SAVOIT3Q MINGW64 ~/Desktop,
$ ls a*
a.txt  ab.txt  apple.txt

Tushar Nankani@LAPTOP-SAVOIT3Q MINGW64 ~/Desktop,
$ bash 05_DisplayingFiles.sh

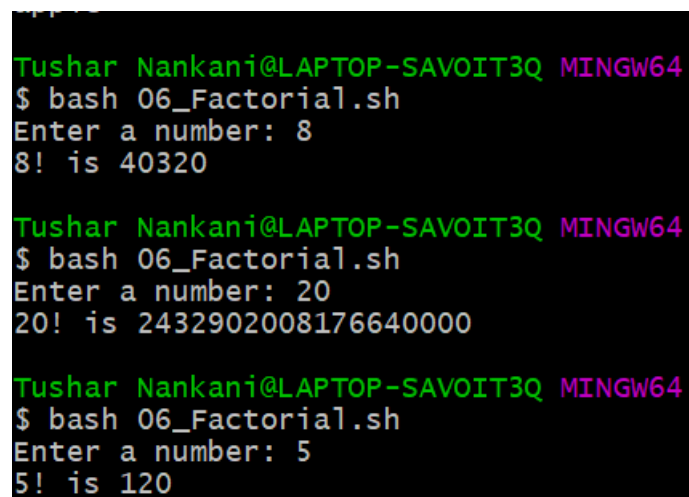
In a.txt:
1
2
3

In ab.txt:
ascmd
dk1c1v sdmv
dvjn;DS

In apple.txt:
apple
```


Q6. Write a shell program to find factorial of a number.

```
read -p "Enter a number: " num
fac=1
for ((i=1;i<=$num;i++))
do
    fac=$((fac*i))
done
echo "$num! is $fac"
```



The screenshot shows a terminal window with the following text:

```
Tushar Nankani@LAPTOP-SAVOIT3Q MINGW64
$ bash 06_Factorial.sh
Enter a number: 8
8! is 40320

Tushar Nankani@LAPTOP-SAVOIT3Q MINGW64
$ bash 06_Factorial.sh
Enter a number: 20
20! is 2432902008176640000

Tushar Nankani@LAPTOP-SAVOIT3Q MINGW64
$ bash 06_Factorial.sh
Enter a number: 5
5! is 120
```

Q7. Write a shell program to check validity of a username and password with a function defined in the code.

```
function check
{
    if [[ $username == "ADMIN"
        && $password == "123456" ]]
    then
        bool=1
    else
        bool=0
    fi
}

read -p "Enter the username: " username
read -p "Enter the password: " password
check
if (($bool==1))
then
    echo "Username and Password verified."
else
    echo "Username and Password not verified."
fi
```

```
Tushar Nankani@LAPTOP-SAVOIT3Q MINGW64 ~/Desktop
$ bash 07_LoginCheck.sh
Enter the username: admin
Enter the password: 12345
Username and Password not verified.

Tushar Nankani@LAPTOP-SAVOIT3Q MINGW64 ~/Desktop
$ bash 07_LoginCheck.sh
Enter the username: ADMIN
Enter the password: 123456
Username and Password verified.

Tushar Nankani@LAPTOP-SAVOIT3Q MINGW64 ~/Desktop
$ bash 07_LoginCheck.sh
Enter the username: ADMIN
Enter the password: 12345
Username and Password not verified.

Tushar Nankani@LAPTOP-SAVOIT3Q MINGW64 ~/Desktop
$ bash 07_LoginCheck.sh
Enter the username: ADMIN
Enter the password: 123456
Username and Password verified.
```