PRACTICAL DEFINITIONS

STREAM : BCA & B.Sc.(IT)SEM 4 SUBJECT : UNIX

➢ Write a shell script Using While Loop.

Create a bash file with the name, 'while_example.sh', **to know the use of** while **loop. In the example,** while **loop will iterate for** 5 **times. The value of** count **variable will increment by** 1 **in each step. When the value of** count **variable will 5 then the** while **loop will terminate.**

```
#!/bin/bash
valid=true
count=1
while [ $valid ]
do
echo $count
if [ $count -eq 5 ];
then
break
fi
((count++))
done
```

Output:

```
$ bash while_example.sh
```

```
ubuntu@ubuntu-VirtualBox:~/code$ bash while_example.sh
1
2
3
4
5
ubuntu@ubuntu-VirtualBox:~/code$
```

➢ Write a shell script Using For Loop.

The basic for **loop declaration is shown in the following example. Create a file named** 'for_example.sh' **and add the following script using** for **loop. Here,** for **loop will iterate for** 10 **times and print all values of the variable,** counter **in single line.**

```
#!/bin/bash
for (( counter=10; counter>0; counter-- ))
do
echo -n "$counter "
done
printf "\n"
```

Output:

```
$ bash for_example.sh
```

```
ubuntu@ubuntu-VirtualBox:~/code$ bash for_example.sh
10 9 8 7 6 5 4 3 2 1
ubuntu@ubuntu-VirtualBox:~/code$
```

➢ Write a shell script to Get User Input.

'read**' command is used to take input from user in bash. Create a file named '**user_input.sh**' and add the following script for taking input from the user. Here, one string value will be taken from the user and display the value by combining other string value.**

```
#!/bin/bash
echo "EnterYourName"
read name
echo "Welcome $name to LinuxHint"
```

Output:

$ bash user_input.sh

```
ubuntu@ubuntu-VirtualBox:~/code$ bash user_input.sh
Enter Your Name
Fahmida
Welcome Fahmida to LinuxHint
ubuntu@ubuntu-VirtualBox:~/code$
```

➢ Write a shell script Using If statement.

You can use if condition with single or multiple conditions. Starting and ending block of this statement is define by 'if' **and** 'fi'**. Create a file named '**simple_if.sh**' with the following script to know the use** if **statement in bash. Here,** 10 **is assigned to the variable,** n**. if the value of** $n **is less than 10 then the output will be "**It is a one digit number**", otherwise the output will be "**It is a two digit number**". For comparison,** '-lt' **is used here. For comparison, you can also use** '-eq' **for** equality**,** '-ne' **for** not equality **and** '-gt' **for** greater than **in bash script.**

```
#!/bin/bash
n=10
if [ $n -lt 10 ];
then
echo "It is a one digit number"
else
echo "It is a two digit number"
fi
```

Output:

```
$ bash simple_if.sh
```

```
ubuntu@ubuntu-VirtualBox:~/code$ bash simple_if.sh
It is a two digit number
ubuntu@ubuntu-VirtualBox:~/code$
```

➢ Write a shell script Using If statement with AND logic.

Different types of logical conditions can be used in if statement with two or more conditions. How you can define multiple conditions in if statement using **AND** logic is shown in the following example. **'&&'** is used to apply **AND** logic of **if** statement. Create a file named **'if_with_AND.sh'** to check the following code. Here, the value of **username** and **password** variables will be taken from the user and compared with '**admin**' and '**secret**'. If both values match then the output will be "**valid user**", otherwise the output will be "**invalid user**".

```
!/bin/bash

echo "Enter username"
read username
echo "Enter password"
```

```
read password

if [[ ( $username == "admin" && $password == "secret" ) ]]; then
echo "valid   user"
else
echo "invalid         user"
fi
```

Output:

```
$ bash if_with_AND.sh
```

```
ubuntu@ubuntu-VirtualBox:~/code$ bash if_with_AND.sh
Enter username
admin
Enter password
1234
invalid user
ubuntu@ubuntu-VirtualBox:~/code$ bash if_with_AND.sh
Enter username
admin
Enter password
secret
valid user
ubuntu@ubuntu-VirtualBox:~/code$
```

➢ Write a shell script Using If statement with OR logic.

'||' is used to define OR **logic in** if **condition. Create a file named** 'if_with_OR.sh' **with the following code to check the use of** OR **logic of** if **statement. Here, the value of** n **will be taken from the user. If the value is equal to** 15 **or** 45 **then the output will be "**You won the game**", otherwise the output will be "**You lost the game**".**

```
#!/bin/bash
```

```
echo "Enter any number"
read n

if [[ ( $n -eq 15 || $n -eq 45 ) ]]
then
echo "You won the game"
else
echo "You lost the game"
fi
```

Output:

$ bash if_with_OR.sh

```
ubuntu@ubuntu-VirtualBox:~/code$ bash if_with_OR.sh
Enter any number
40
You lost the game
ubuntu@ubuntu-VirtualBox:~/code$ bash if_with_OR.sh
Enter any number
15
You won the game
ubuntu@ubuntu-VirtualBox:~/code$
```

➢ Write a shell script Using else If statement.

The use of else if **condition is little different in bash than other programming language. 'elif' is used to define** else if **condition in bash. Create a file named, '**elseif_example.sh**' and add the following script to check how** else if **is defined in bash script.**

```
#!/bin/bash

echo "Enter your lucky number"
read n

if [ $n -eq 101 ];
then
echo "You got 1st prize"
elif [ $n -eq 510 ];
then
echo "You got 2nd prize"
elif [ $n -eq 999 ];
```

```
then
echo "You got 3rd prize"

else
echo "Sorry, try for the next time"
fi
```

Output:

$ bash elseif_example.sh

```
ubuntu@ubuntu-VirtualBox:~/code$ bash elseif_example.sh
Enter your lucky number
101
You got 1st prize
ubuntu@ubuntu-VirtualBox:~/code$ bash elseif_example.sh
Enter your lucky number
999
You got 3rd prize
ubuntu@ubuntu-VirtualBox:~/code$ bash elseif_example.sh
Enter your lucky number
100
Sorry, try for the next time
ubuntu@ubuntu-VirtualBox:~/code$
```

➢ Write a shell script Using case statement.

Case **statement is used as the alternative of** if-elseif-else **statement. The starting and ending block of this statement is defined by** ‘case’ **and** ‘esac’**. Create a new file named,** ‘case_example.sh’ **and add the following script. The output of the following script will be same to the previous** else if **example.**

```
#!/bin/bash

echo "Enter your lucky number"
read n
case $n in
101)
```

```
echo echo "You got 1st prize" ;;
510)
echo "You got 2nd prize" ;;
999)
echo "You got 3rd prize" ;;
*)
echo "Sorry, try for the next time" ;;
esac
```

Output:

$ bash case_example.sh

```
ubuntu@ubuntu-VirtualBox:~/code$ bash case_example.sh
Enter your lucky number
101
echo You got 1st prize
ubuntu@ubuntu-VirtualBox:~/code$ bash case_example.sh
Enter your lucky number
510
You got 2nd prize
ubuntu@ubuntu-VirtualBox:~/code$ bash case_example.sh
Enter your lucky number
999
You got 3rd prize
ubuntu@ubuntu-VirtualBox:~/code$ bash case_example.sh
Enter your lucky number
777
Sorry, try for the next time
ubuntu@ubuntu-VirtualBox:~/code$
```

➢ Write a shell script to Get Arguments from Command Line.

Bash script can read input from command line argument like other programming language. For example, $1 **and** $2 **variable are used to read first and second command line arguments. Create a file named "**command_line.sh**" and add the following script. Two argument values read by the following script and prints the total number of arguments and the argument values as output.**

```
#!/bin/bash
echo "Total arguments :      $#"
echo "1st     Argument  =      $1"
echo "2nd argument = $2"
```

Output:

$ bash command_line.sh Linux Hint

```
ubuntu@ubuntu-VirtualBox:~/code$ bash command_line.sh Linux Hint
Total arguments : 2
1st Argument = Linux
2nd argument = Hint
ubuntu@ubuntu-VirtualBox:~/code$
```

➢ Write a shell script to Get Arguments from Command Line with names.

How you can read command line arguments with names is shown in the following script. Create a file named, 'command_line_names.sh' and add the following code. Here, two arguments, X and Y are read by this script and print the sum of X and Y.

```
#!/bin/bash
for arg in "$@"
do
index=$(echo $arg | cut -f1 -d=)
val=$(echo $arg | cut -f2 -d=)
case $index in
X) x=$val;;

Y) y=$val;;

*)
```

```
esac
done
((result=x+y))
echo "X+Y=$result"
```

Output:

$ bash command_line_names **X**=45 **Y**=30

```
ubuntu@ubuntu-VirtualBox:~/code$ bash command_line.sh Linux Hint
Total arguments : 2
1st Argument = Linux
2nd argument = Hint
ubuntu@ubuntu-VirtualBox:~/code$
```

➢ Write a shell script to combine string variables.

You can easily combine string variables in bash. Create a file named "string_combine.sh**" and add the following script to check how you can combine string variables in bash by placing variables together or using '+' operator.**

```
#!/bin/bash

string1="Linux"
string2="Hint"
echo "$string1$string2"
string3=$string1+$string2
string3+=" is a good tutorial blog site"
echo $string3
```

Output:

$ bash string_combine.sh

```
ubuntu@ubuntu-VirtualBox:~/code$ bash string_combine.sh
LinuxHint
Linux+Hint is a good tutorial blog site
ubuntu@ubuntu-VirtualBox:~/code$
```

➢ Write a shell script to Add two numbers.

You can do the arithmetical operations in bash in different ways. How you can add two integer numbers in bash using double brackets is shown in the following script. Create a file named 'add_numbers.sh' with the following code. Two integer values will be taken from the user and printed the result of addition.

```
#!/bin/bash
echo "Enter first    number"
read x
echo "Enter second        number"
read y
$(( sum=x+y ))
echo "The result of addition=$sum"
```

Output:

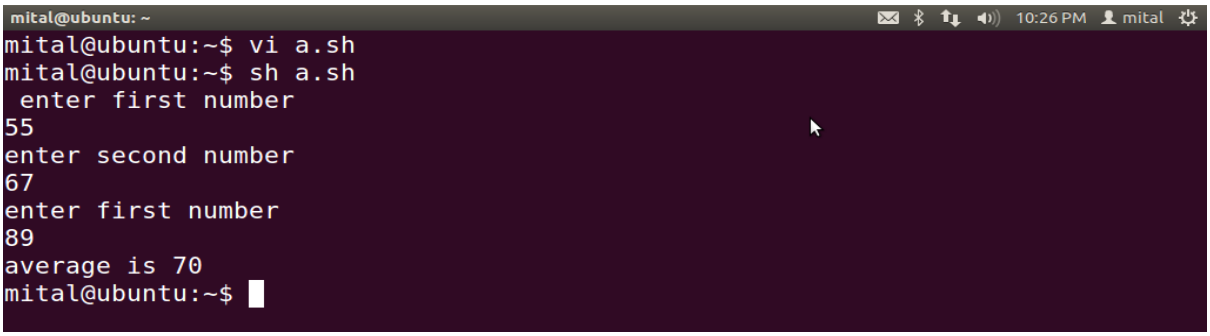
$ bash add_numbers.sh

```
ubuntu@ubuntu-VirtualBox:~/code$ bash add_numbers.sh
Enter first number
25
Enter second number
56
The result of addition=81
ubuntu@ubuntu-VirtualBox:~/code$
```

➢ Write a shell script to Average Of 3 Numbers.

```
#!/bin/bash
Echo " enter first number"
read a
echo "enter second number"
read b
echo "enter first number"
 read c
 total=`expr $a + $b + $c`
avg=`expr $total / 3`
 echo "average is $avg"
```

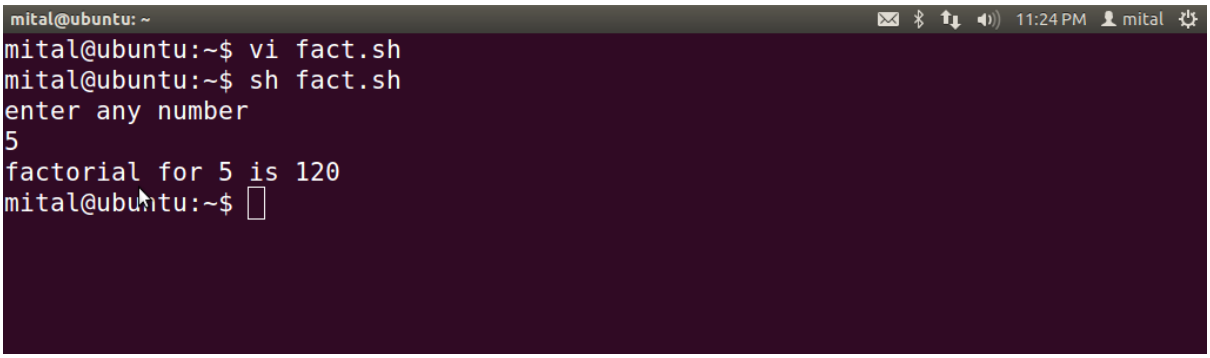
Output:



➢ Write a shell script to Calculate Factorial Of Given Number

```
#!/bin/bash
n=0
 on=0
 fact=1
 echo "enter any number "
 read n
on=$n while [ $n -ge 1 ]
do
 fact=`expr $fact \* $n` n=`expr $n - 1`
 done
 echo "factorial for $on is $fact"
```

Output:



- **Shell Script To Determine Whether Given File Exist or Not.**

```
#!/bin/bash
echo "enter file name"
read fname
if [ -f $fname ] then
echo "$fname file exist"
else
echo "sorry, $fname file does not exist"
fi
```

Output:

➢ Shell Program to find Largest of Three Numbers.

```
#!/bin/bash
echo "Enter Num1"
read num1
echo "Enter Num2"
read num2
echo "Enter Num3"
read num3

if [ $num1 -gt $num2 ] && [ $num1 -gt $num3 ]
then
   echo $num1
elif [ $num2 -gt $num1 ] && [ $num2 -gt $num3 ]
then
   echo $num2
else
   echo $num3
fi
```

Output:

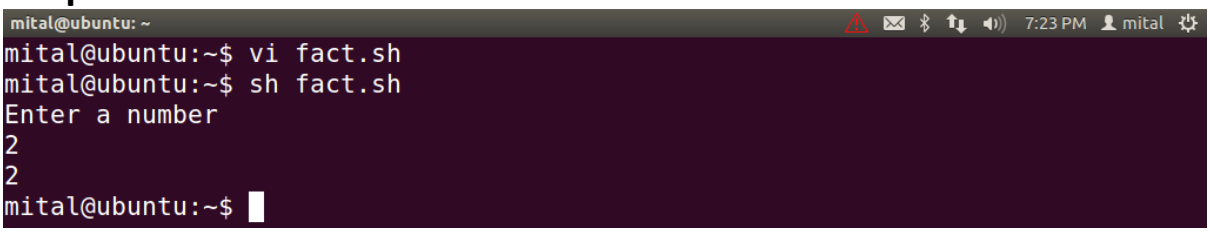
```
mital@ubuntu:~$ vi max.sh
mital@ubuntu:~$ sh max.sh
Enter Num1
1
Enter Num2
2
Enter Num3
3
3
mital@ubuntu:~$
```

➢ Shell Program to find Largest of Three Numbers.

```
#!/bin/bash
echo "Enter a number"
read num
```

```
fact=1
while [ $num -gt 1 ]
do
 fact=$((fact * num))  #fact = fact * num
 num=$((num - 1))      #num = num - 1
done
echo $fact
```

Output:



➢ Write a Shell Script Basic Math Calculator (Case Statement)

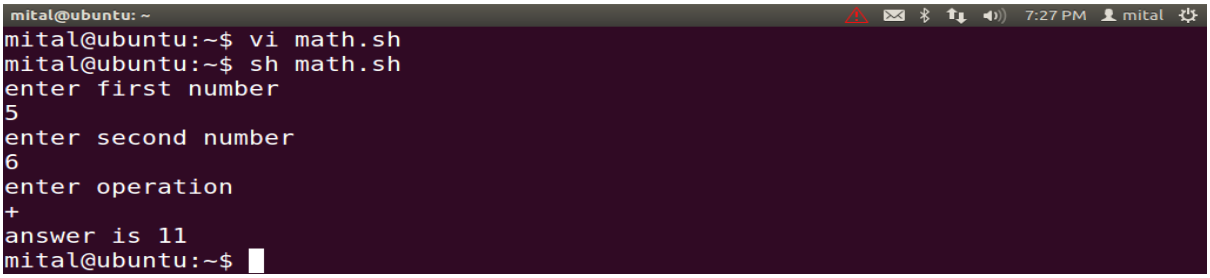
```
#!/bin/bash
echo enter first number
 read a
echo enter second number
 read b
 echo enter operation
 read c
case $c in
+) z=`expr $a + $b`;;
 -) z=`expr $a - $b`;;
/) z=`expr $a / $b`;;
 x) z=`expr $a \* $b`;;
 esac
echo answer is $z
```

Output:

➢ Write a Shell Program to Print an Array.

```
#!/bin/bash
echo "Enter number of elements: "
 read n
echo "Enter array elements: "
 for ((i=0; i<n; i++))
do
 read a[$i]
done
echo "The elements of an array are: "
for ((i=0; i<n; i++))
do
echo a[$i];
 echo ${a[$i]}
done
```

Output:



➢ Write a Shell Program to Find no is prime or not.

```
echo enter a number
read n
```

```
na=`expr $n - 1`
while [ $na -gt 1 ]
do
        t=`expr $n % $na`
        if test $t = 0
        then
                echo this number is not prime
                exit
        else
                na=`expr $na - 1`
        fi
done
if [ $na -eq 1 ]
then
        echo this is a prime number
fi
```

Output:

➢ Write a Shell Program to create a Simple Function.

```
#!/bin/bash
function F1()
{
echo 'I like bash programming'
}

F1
```

Output:

```
mital@ubuntu: ~                                    10:00 PM mital
mital@ubuntu:~$ vi f.sh
mital@ubuntu:~$ bash f.sh
I like bash programming
mital@ubuntu:~$
```

➢ Write a Shell Program to create a directory and check exist or not.

```
#!/bin/bash
echo "Enter directory name"
read ndir
if [ -d "$ndir" ]
then
echo "Directory exist"
else
`mkdir $ndir`
echo "Directory created"
fi
```

Output:

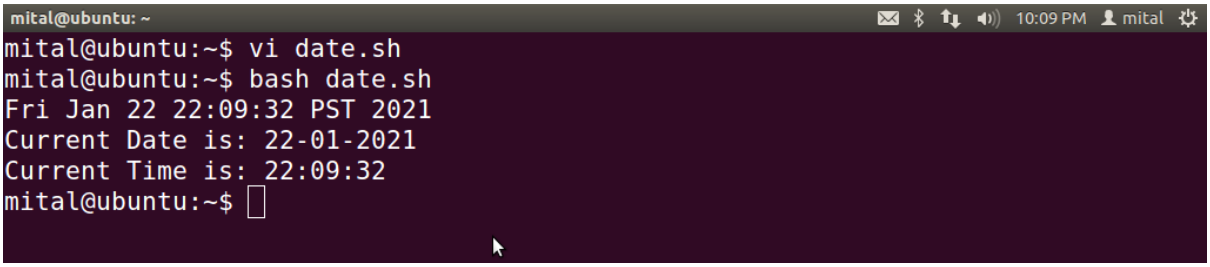
```
mital@ubuntu: ~                                    10:06 PM mital
mital@ubuntu:~$ vi d.sh
mital@ubuntu:~$ bash d.sh
Enter directory name
mital
Directory created
mital@ubuntu:~$ bash d.sh
Enter directory name
mital
Directory exist
mital@ubuntu:~$
```

➢ Write a Shell Program to display current date and time.

```
#!/bin/bash
Year=`date +%Y`
```

```
Month=`date +%m`
Day=`date +%d`
Hour=`date +%H`
Minute=`date +%M`
Second=`date +%S`
echo `date`
echo "Current Date is: $Day-$Month-$Year"
echo "Current Time is: $Hour:$Minute:$Second"
```

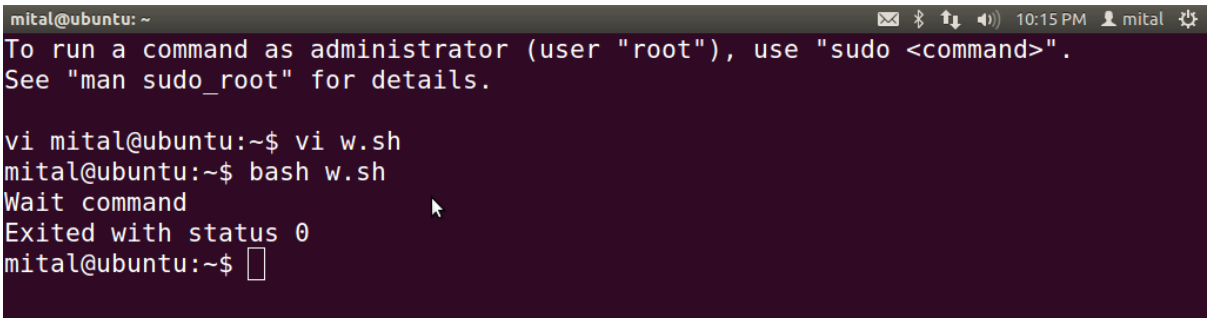
Output:



➢ Write a Shell script to wait command.

```
#!/bin/bash
echo "Wait command" &
process_id=$!
wait $process_id
echo "Exited with status $?"
```

Output:



➢ Write a Shell script to sleep command.

```
#!/bin/bash

echo “Wait for 5 seconds”
sleep 5
echo “Completed”
```

Output:

➢ Write a Shell script to if else statement.

```
#!/bin/bash
echo "Enter your lucky number"
read n

if [ $n -eq 101 ];
then
echo "You got 1st prize"
elif [ $n -eq 510 ];
then
echo "You got 2nd prize"
elif [ $n -eq 999 ];
then
echo "You got 3rd prize"
else
echo "Sorry, try for the next time"
fi
```

Output:

```
ubuntu@ubuntu-VirtualBox:~/code$ bash elseif_example.sh
Enter your lucky number
101
You got 1st prize
ubuntu@ubuntu-VirtualBox:~/code$ bash elseif_example.sh
Enter your lucky number
999
You got 3rd prize
ubuntu@ubuntu-VirtualBox:~/code$ bash elseif_example.sh
Enter your lucky number
100
Sorry, try for the next time
ubuntu@ubuntu-VirtualBox:~/code$
```

THANK YOU