Lab manual

Description:

This manual aims to familiarize the user with basic loops and command line arguments of linux bourne shell.

Part 1: Decisions

Description and Syntax:

Decisions are taken based on some particular value or input.

`Syntax of if statements:`

`if(some test)`

`then do something`

`Syntax of if statements in linux:`

`if [ some test ];`

`then do something`

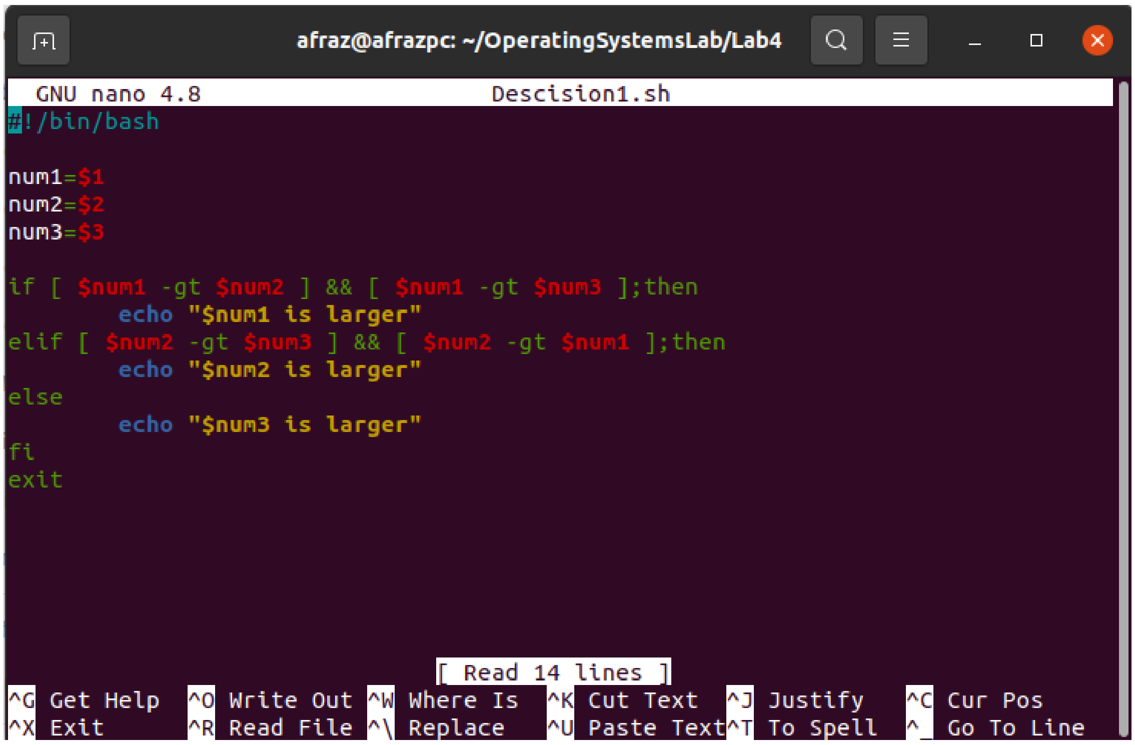
Now, linux uses the basic if-else block for its decision taking. Further examples will be given in the sample tasks below.

Tasks:

1. Create a script that takes 3 command line arguments and prints the larger of the three numbers.

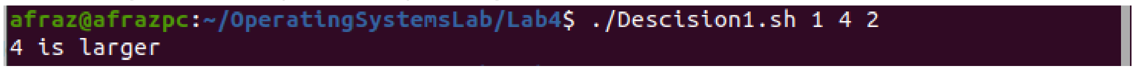
   This uses the simple concept of two if and one else statements. First we get the three command line arguments by the `$` directive. Then, as the user can see in the attached screenshot, the linux syntax of flag based if-else is given below. One can also utilize c++ based if else, they come in double brackets, example syntax `if (( some test in c++`

`syntax))`.



```
afraz@afrazpc: ~/OperatingSystemsLab/Lab4
  GNU nano 4.8                         Descision1.sh
#!/bin/bash

num1=$1
num2=$2
num3=$3

if [ $num1 -gt $num2 ] && [ $num1 -gt $num3 ];then
        echo "$num1 is larger"
elif [ $num2 -gt $num3 ] && [ $num2 -gt $num1 ];then
        echo "$num2 is larger"
else
        echo "$num3 is larger"
fi
exit

                              [ Read 14 lines ]
^G Get Help  ^O Write Out ^W Where Is  ^K Cut Text  ^J Justify   ^C Cur Pos
^X Exit      ^R Read File ^\ Replace   ^U Paste Text^T To Spell  ^_ Go To Line
```

This code gives the output corresponding to :



```
afraz@afrazpc:~/OperatingSystemsLab/Lab4$ ./Descision1.sh 1 4 2
4 is larger
```

2. Create a bash script that prints different output corresponding to day of week.

   `switch case syntax in linux`

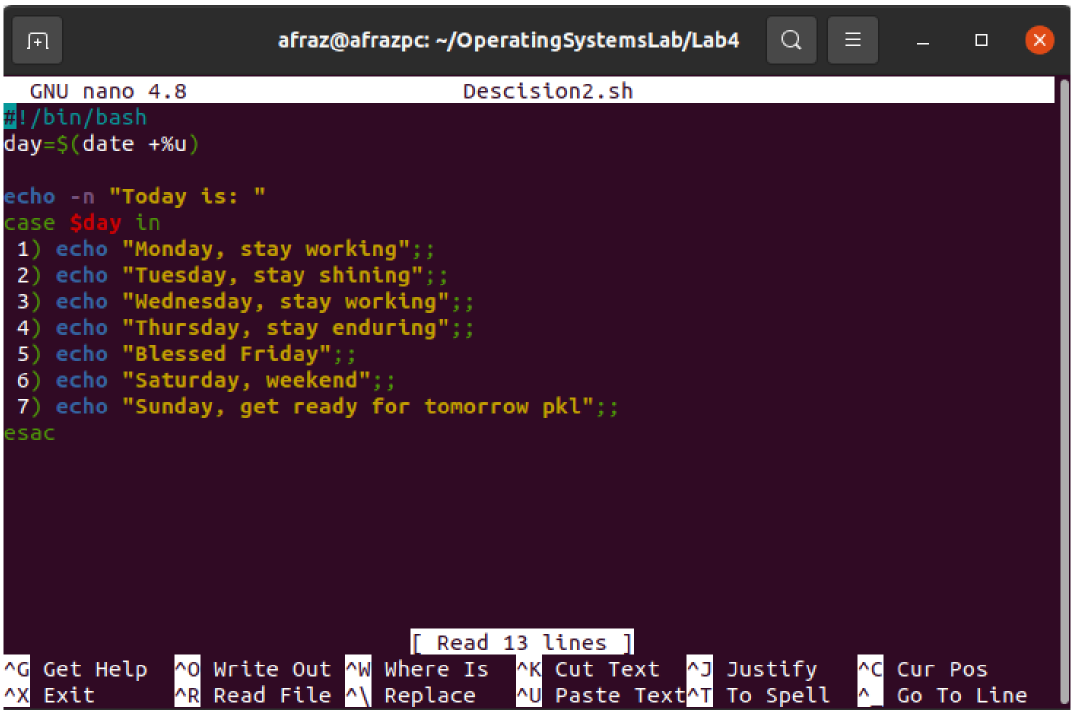
   `case $VALUE in`

   `case 1) ....`

   `case 2) ....`

   `esac    #ends the switch block`

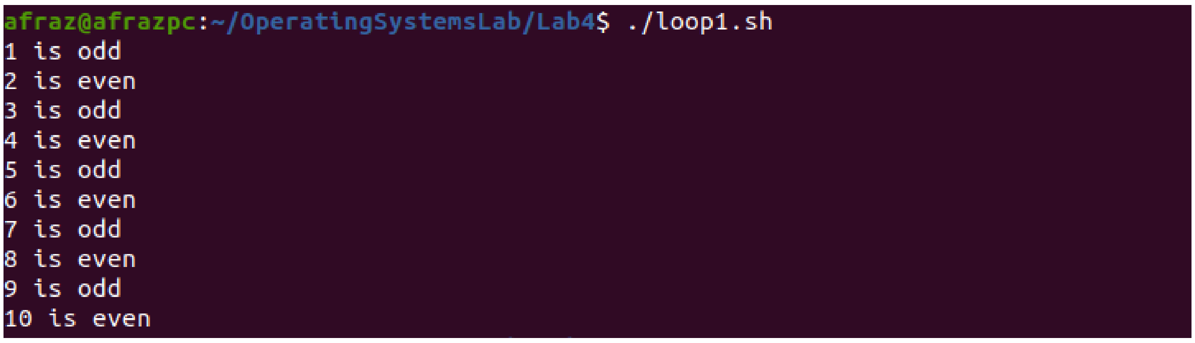
   We use a switch case in this task. First, we use the `date=$(date +%u)` to convert the day to an integer value. Then we apply a switch case to check what was the day and print the message. The syntax of a switch statement can be seen in the code block attached below. Generally, an switch block is not recommended for usage because of readability issues.



afraz@afrazpc: ~/OperatingSystemsLab/Lab4

```
  GNU nano 4.8                    Descision2.sh
#!/bin/bash
day=$(date +%u)

echo -n "Today is: "
case $day in
 1) echo "Monday, stay working";;
 2) echo "Tuesday, stay shining";;
 3) echo "Wednesday, stay working";;
 4) echo "Thursday, stay enduring";;
 5) echo "Blessed Friday";;
 6) echo "Saturday, weekend";;
 7) echo "Sunday, get ready for tomorrow pkl";;
esac

                               [ Read 13 lines ]
^G Get Help   ^O Write Out  ^W Where Is   ^K Cut Text   ^J Justify    ^C Cur Pos
^X Exit       ^R Read File  ^\ Replace    ^U Paste Text ^T To Spell   ^_ Go To Line
```

This code gives the output of:



```
afraz@afrazpc:~/OperatingSystemsLab/Lab4$ ./loop1.sh
1 is odd
2 is even
3 is odd
4 is even
5 is odd
6 is even
7 is odd
8 is even
9 is odd
10 is even
```

Part 2: Loops

Syntax:

`Syntax of a for loop`

`for i in {values}`

`do`

`#do something`

`done`

`Syntax of a while loop`

`while [ some test ]`

`do`

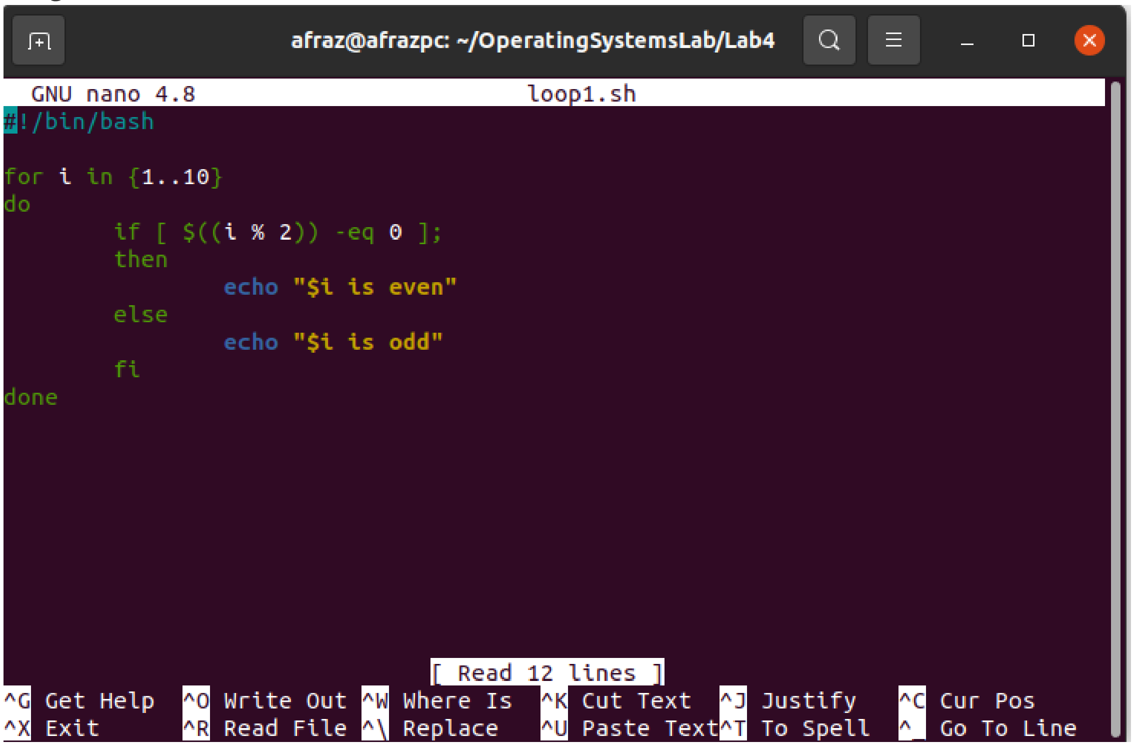
`#do something`

`done`

Tasks:

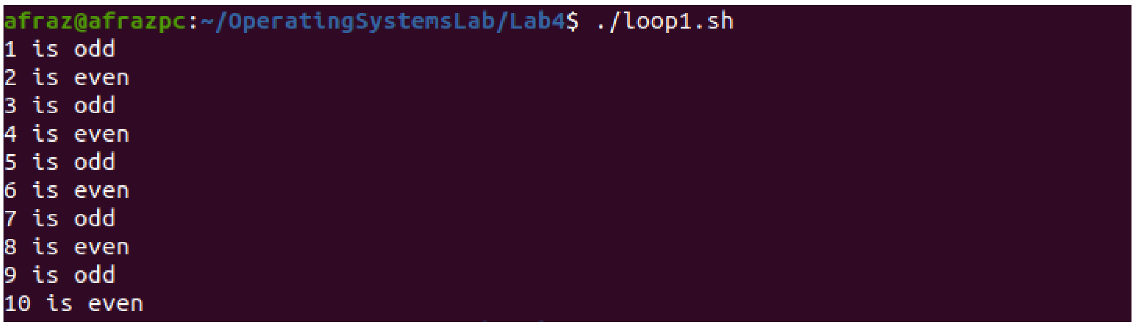
3. Create a script that prints each number from 1 to 10 and whether they are even or odd.

   We use the if-else block covered in first part of the manual to check whether the number is even or odd. A simple for loop is required in this case, whose syntax can be seen in the image below.



```
#!/bin/bash

for i in {1..10}
do
	if [ $((i % 2)) -eq 0 ];
	then
		echo "$i is even"
	else
		echo "$i is odd"
	fi
done
```

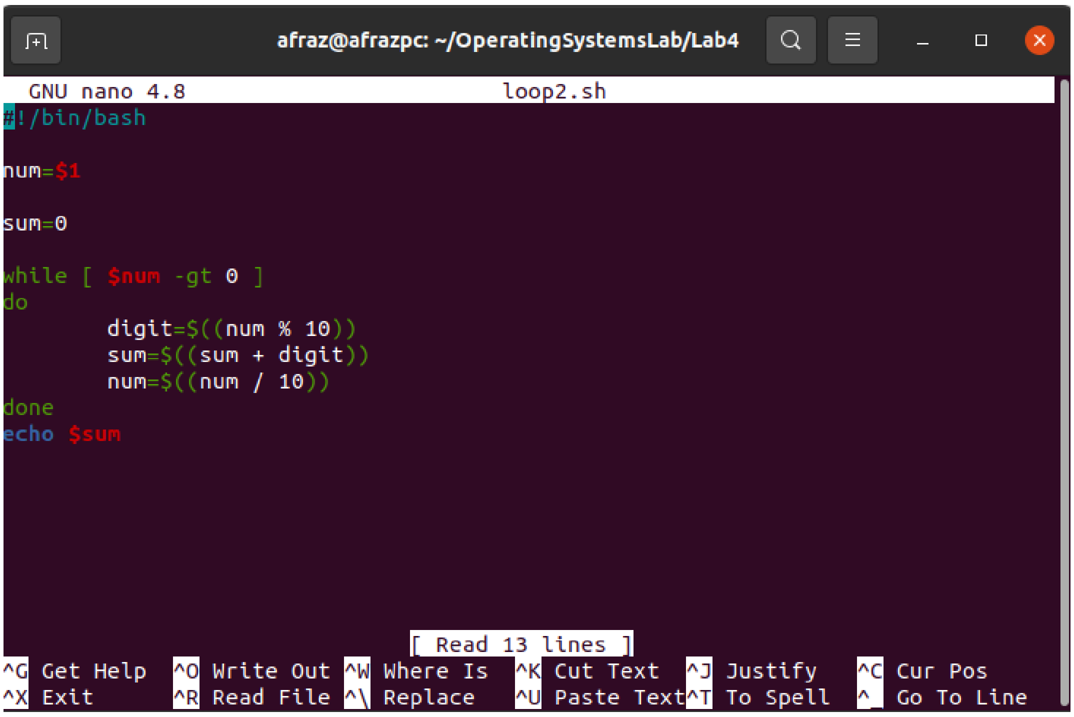
The output for this is:



```
afraz@afrazpc:~/OperatingSystemsLab/Lab4$ ./loop1.sh
1 is odd
2 is even
3 is odd
4 is even
5 is odd
6 is even
7 is odd
8 is even
9 is odd
10 is even
```

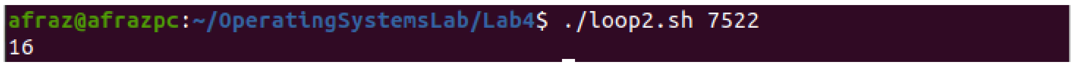
4. Create a bash script that adds all numbers in a command line arguments.

   Recall from Programming 101 that a string is an array of characters. We simply traverse each character by character and add all values.



```
afraz@afrazpc: ~/OperatingSystemsLab/Lab4
  GNU nano 4.8                          loop2.sh
#!/bin/bash

num=$1

sum=0

while [ $num -gt 0 ]
do
        digit=$((num % 10))
        sum=$((sum + digit))
        num=$((num / 10))
done
echo $sum









                               [ Read 13 lines ]
^G Get Help  ^O Write Out ^W Where Is  ^K Cut Text  ^J Justify   ^C Cur Pos
^X Exit      ^R Read File ^\ Replace   ^U Paste Text^T To Spell  ^_ Go To Line
```

The output for the code is:



```
afraz@afrazpc:~/OperatingSystemsLab/Lab4$ ./loop2.sh 7522
16
```

Part 3: Functions

Description and Syntax:

In functions, linux behaves a little differently.

Basically, the syntax of a function is:

`function nameOfFunction(){`

 `do something in function`

 `echo $RETURN VALUE        #to return something if applicable`

`}`

Here two important things are required for clarity.

First of all, a clash may occur in global and local variables. This is resolved by `local` added before variable name.

Secondly, the function call in linux differs from c++. Here, a function is called by the syntax:

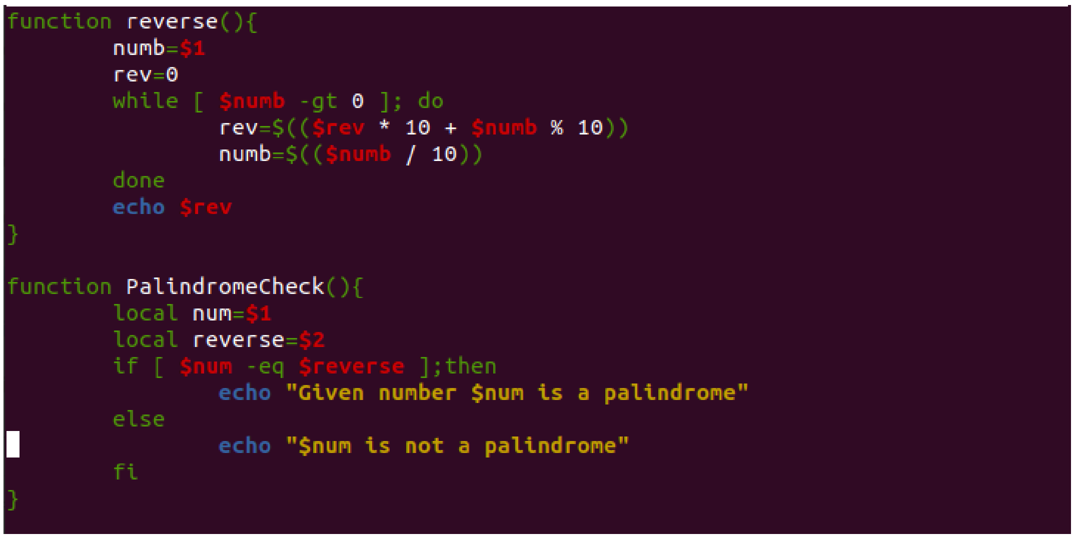
`functionName $param1 $param2 ....`

and they are *caught* in the function by the `$` directive.

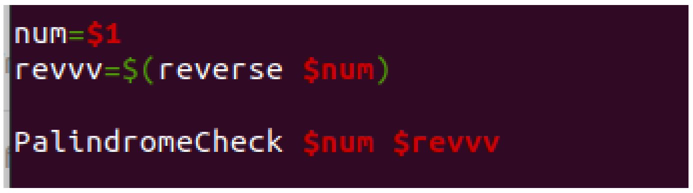
Sample Tasks:

5. Create a shell script that checks for whether a number is a palindrome or not.

This involves some mathematical computations. Else is simply loops and if statements covered in parts 1 and 2.

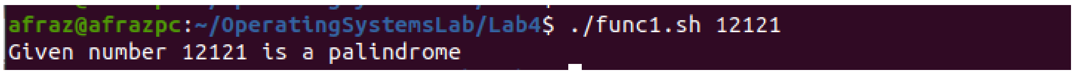


```
function reverse(){
        numb=$1
        rev=0
        while [ $numb -gt 0 ]; do
                rev=$(($rev * 10 + $numb % 10))
                numb=$(($numb / 10))
        done
        echo $rev
}

function PalindromeCheck(){
        local num=$1
        local reverse=$2
        if [ $num -eq $reverse ];then
                echo "Given number $num is a palindrome"
        else
                echo "$num is not a palindrome"
        fi
}
```



```
num=$1
revvv=$(reverse $num)

PalindromeCheck $num $revvv
```

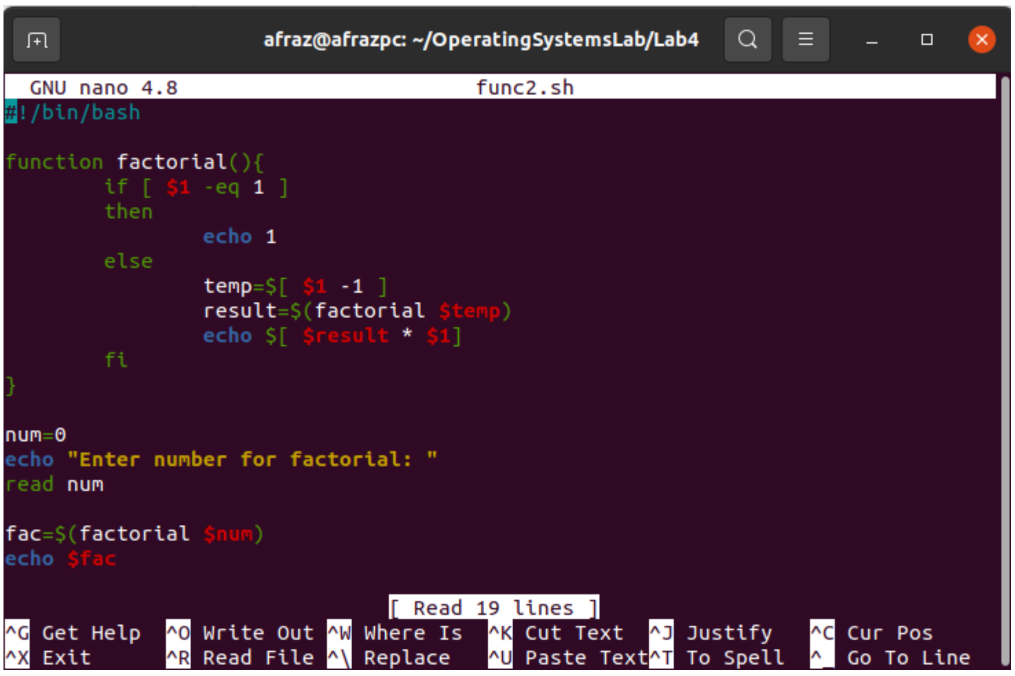
The output for this script is



```
afraz@afrazpc:~/OperatingSystemsLab/Lab4$ ./func1.sh 12121
Given number 12121 is a palindrome
```

7. Recursively calculate the factorial of a number.

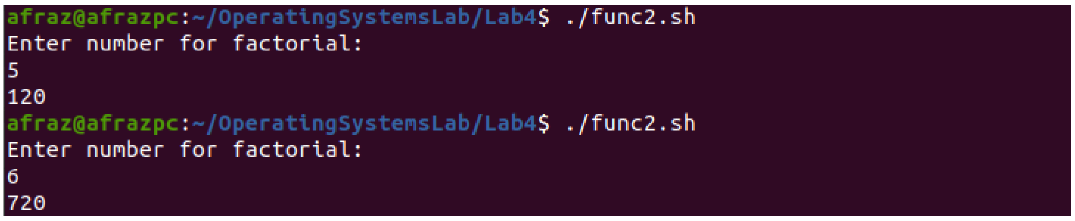
   Recursion is when a function calls itself. There needs to be given a break condition for the function else the recursive call will overflow the stack. The code is:



afraz@afrazpc: ~/OperatingSystemsLab/Lab4

```
  GNU nano 4.8                              func2.sh
#!/bin/bash

function factorial(){
        if [ $1 -eq 1 ]
        then
                echo 1
        else
                temp=$[ $1 -1 ]
                result=$(factorial $temp)
                echo $[ $result * $1]
        fi
}

num=0
echo "Enter number for factorial: "
read num

fac=$(factorial $num)
echo $fac

                              [ Read 19 lines ]
^G Get Help   ^O Write Out  ^W Where Is   ^K Cut Text   ^J Justify    ^C Cur Pos
^X Exit       ^R Read File  ^\ Replace    ^U Paste Text ^T To Spell   ^_ Go To Line
```

The output:



```
afraz@afrazpc:~/OperatingSystemsLab/Lab4$ ./func2.sh
Enter number for factorial:
5
120
afraz@afrazpc:~/OperatingSystemsLab/Lab4$ ./func2.sh
Enter number for factorial:
6
720
```