## Task 01

### part 1.

##### Create a Bash script which will take 3 numbers as command line arguments. It will print to the screen the larger of the three numbers.

### Solution:

In this bash script code. **Echo** command is use to print string on console and **read** command is use to take input from console .

**If** command is a conditional Statement its syntax is . if [ **var1** compare **var2** ] .

**$** sign is used for compare two values and **-gt** **-ge** is operator use to compare values than.

Inside the square brackets n1 is compared with n2. Similarly **-ge** is used for greater than or Equal. after that use **then** keyword to continue it write echo command to show the output on terminal. And at last fi statement is used close the if condition like we close the loop with bracket.

#!/bin/bash  
echo "Enter no 1"  
read n1  
echo "Enter no 2"  
read n2  
echo "Enter no 3"  
read n3  
if [ $n1 -ge $n2 ] && [ $n1 -ge $n3 ]; then  
 echo "$n1 is the largest number."  
elif [ $n2 -ge $n1 ] && [ $n2 -ge $n3 ]; then  
 echo "$n2 is the largest number."  
else  
 echo "$n3 is the largest number."  
fi

#### Code:



#### Output:



### Part 2:

**Create a Bash script which will print a message based upon which day of the week it is (eg. 'Happy day' for Wednesday, 'blessed' for Friday etc) using switch statement.**

### Solution

In variable day we take a day from our system using **$(date +%A)**. date is carried out from our system. **Switch** is a statement like **if statement** in which we check cases between different cases. The one that matches output in that condition print on console.

#!/bin/bash  
  
day=$(date +%A)  
  
case $day in  
 Monday)  
 echo "focused.DBS day "  
 ;;  
 Tuesday)  
 echo "motivation.DBS Lab"  
 ;;  
 Wednesday)  
 echo "hoilyday!"  
 ;;  
 Thursday)  
 echo "keep pushing. Busy day"  
 ;;  
 Friday)  
 echo "Blessed, last day"  
 ;;  
 Saturday)  
 echo "Relax and recharge."  
 ;;  
 Sunday)  
 echo "ops.Last day of the week, "  
 ;;  
esac

#### Code:



#### Output:



# Task2:

### part1

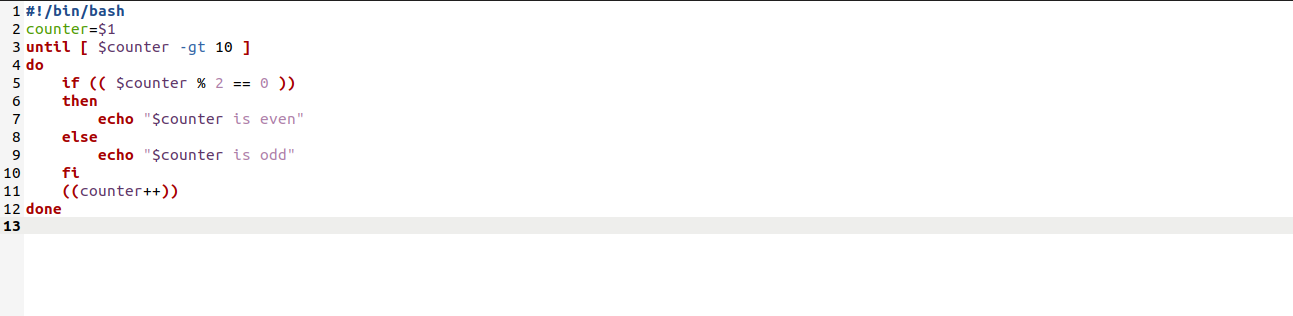
##### Create a simple script which will print the numbers 1 - 10 (each on a separate line) and whether they are even or odd.

### Solution

In this problem we use a counter variable in which we assign a value of **S1** which we provide when we run code.And we use a **until** keyword same for loop inside the bracket write the condition that if **counter variable greater than 10** .It will check 10 times inside the until statement we use do statement. do means do a task below that line in which 1st condition it checks if a number is divisible of 2 then it is an even number otherwise run the else statement in which print a message that the number is odd and close the if condition then increment in the counter variable by writing in double circle brackets counter++ and at last write done for complete the loop.

#!/bin/bash  
counter=$1  
until [ $counter -gt 10 ]  
do  
 if (( $counter % 2 == 0 ))  
 then  
 echo "$counter is even"   
 else  
 echo "$counter is odd"  
 fi  
 ((counter++))  
done

#### Code:



#### Output:



### part2

##### Write a program that read number as input, calculate, and return the summation of the all digits of the number.

##### Example:

##### If given number: 745

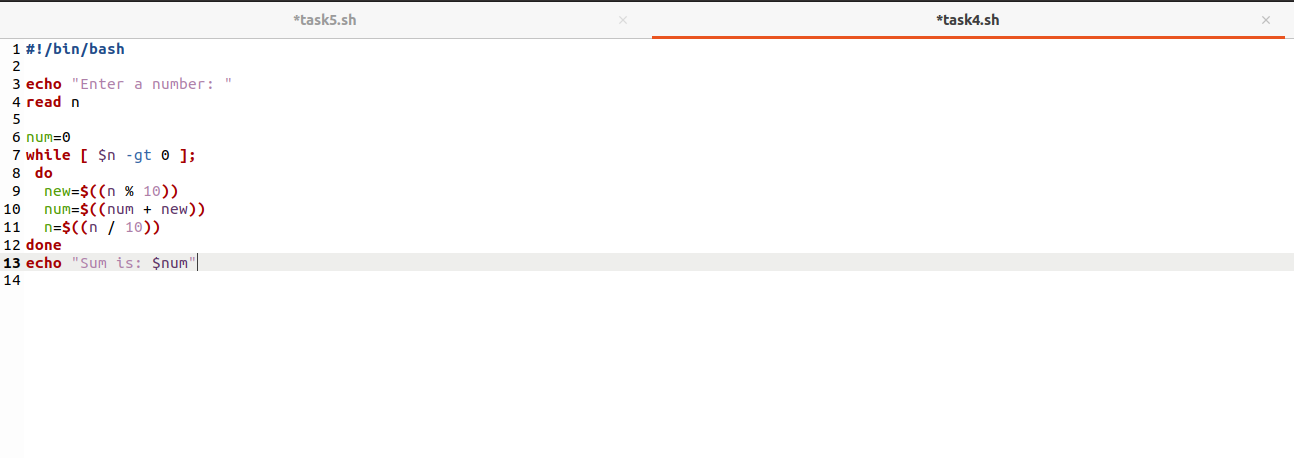
##### Then result will be: (7+4+5) = 16

### Solution

In this problem we get a number from terminal and create a variable **num** and assign it a value of 0 at the time of initiation the we use the while loop and the condition in that is when the number from the terminal is greater than 0 take the mod of that number and assign it into a new variable and sum it to the variable this process runs until the condition of greater than 0 false and the variable n is updated and at last of while write done to complete the loop.

#!/bin/bash  
  
echo "Enter a number: "  
read n  
  
num=0  
while [ $n -gt 0 ];  
 do  
 new=$((n % 10))  
 num=$((num + new))  
 n=$((n / 10))  
done  
echo "Sum is: $num"

#### Code:



#### Output:



### Task 3:

### part1

##### Write a shell script which takes a positive integer as an argument on the terminal and then checks if it is a palindrome or not. In order to find the reverse of this number it must be passed to function named reverse(), which computes the reverse and passes both, the number and its reverse to another function called palindromeCheck(). palindromeCheck() then compares the numbers and echoes if the number is a palindrome or not.

### Solution

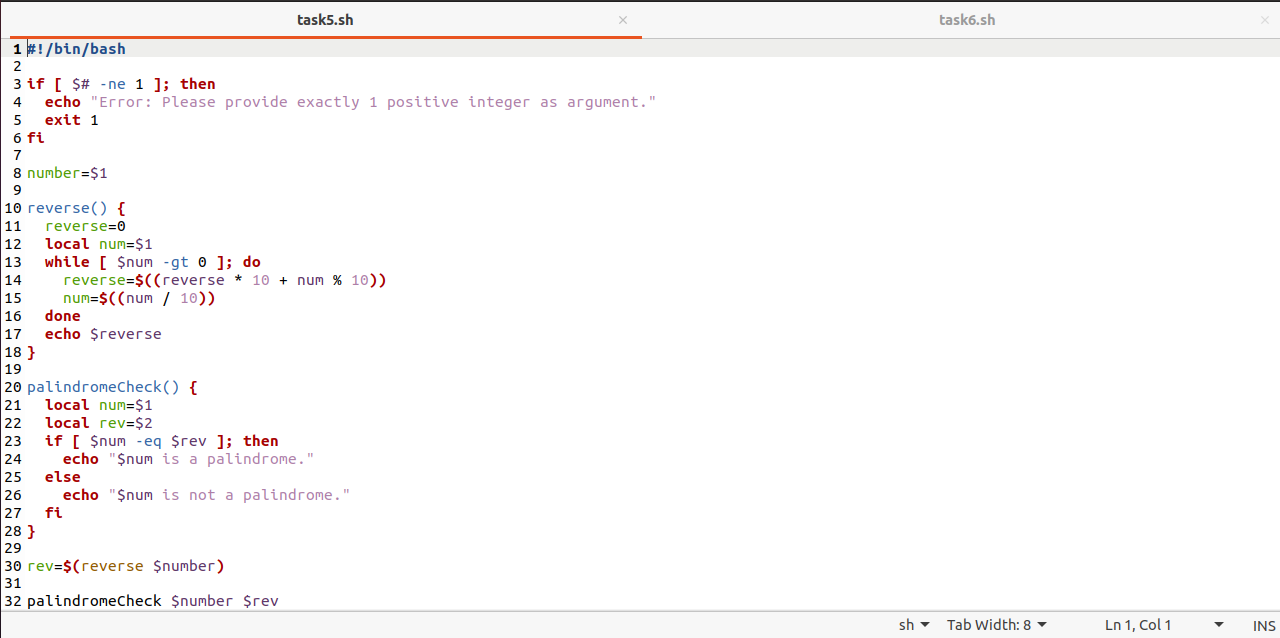
In this problem we create two function one is named for **palindrome** and **reverse** .

**Palindrome number**. It is a number when it reversed it remain same as the number.

In that we created a reverse function which takes the mod of number and another number multiplied by 10 for example if it is 66 we take mod of it 6 is carried out then multiply the number with 10 it is690 then add remaining number which is 6 and it is 66. after creating reverse function Now create **palindromecheck()** in which number is checked with the number that was reversed if the value is same as reverse then it is a palindrome number.

#!/bin/bash  
  
if [ $# -ne 1 ]; then  
 echo "Error: Please provide exactly 1 positive integer as argument."  
 exit 1  
fi  
  
number=$1  
  
reverse() {  
 reverse=0  
 local num=$1  
 while [ $num -gt 0 ]; do  
 reverse=$((reverse \* 10 + num % 10))  
 num=$((num / 10))  
 done  
 echo $reverse  
}  
  
palindromeCheck() {  
 local num=$1  
 local rev=$2  
 if [ $num -eq $rev ]; then  
 echo "$num is a palindrome."  
 else  
 echo "$num is not a palindrome."  
 fi  
}  
  
rev=$(reverse $number)  
  
palindromeCheck $number $rev

#### Code:



#### Output:



#### Part 2:

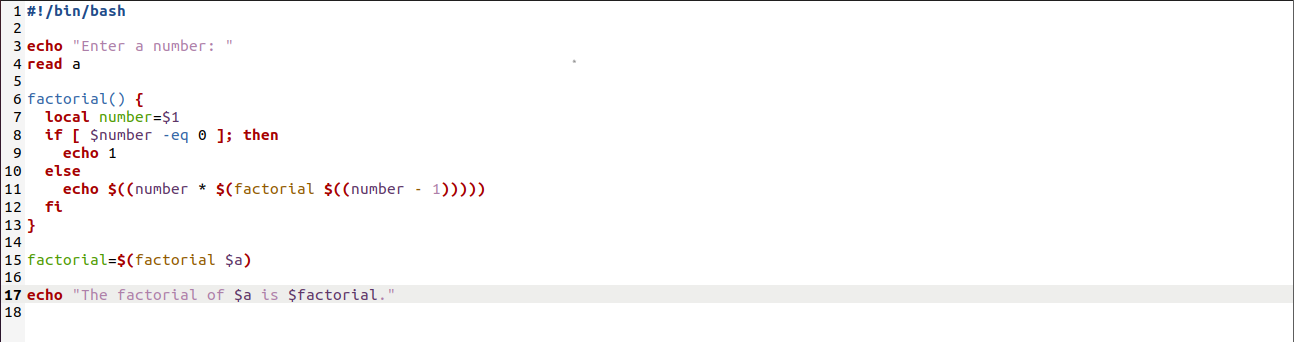
##### Write a shell script which reads a number from the user and passes it to function named factorial(). This function finds the factorial of the number and prints it on the terminal. This task must be done using recursion.

### Solution

In this problem we have to find factorial of a number using recursion we created a function named as **factorial** in which we created a local number and assigns it a value of **$1** and use recursion that if the number is 0 then print 1 other wise go to else statement where in this step the function calls itself again and again which is called recursion until the answer is 1. Then add up all the result and at last we get a number from terminal and give that number to the factorial function which returns its result and we save.

#!/bin/bash  
  
echo "Enter a number: "  
read a  
  
factorial() {  
 local number=$1  
 if [ $number -eq 0 ]; then  
 echo 1  
 else  
 echo $((number \* $(factorial $((number - 1)))))  
 fi  
}  
  
factorial=$(factorial $a)  
  
echo "The factorial of $a is $factorial."

#### Code:



#### Output:

