**Assignment No.03**

**Name: -** Omprakash Khawshi

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**for loop**

**Q.1 Write a program that takes a command-line argument n and prints a table of the**

**powers of 2 that are less than or equal to 2^n.**

**Code: -**

read -p "Enter a number :- " Number

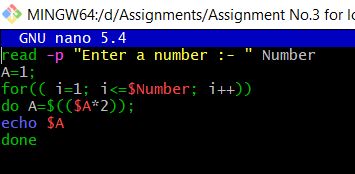
A=1;

for(( i=1; i<=$Number; i++))

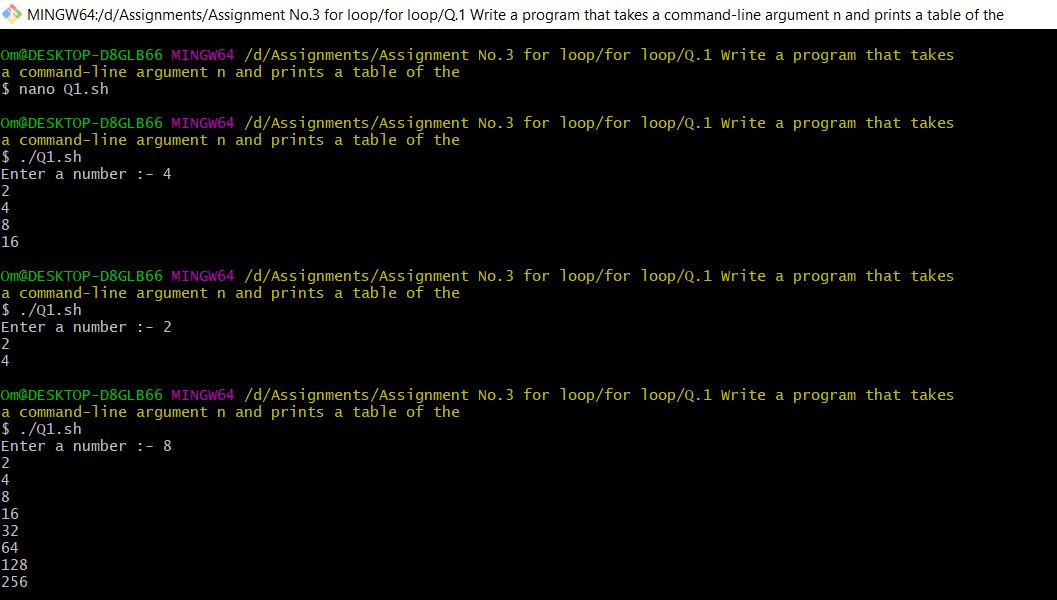
do A=$(($A\*2));

echo $A

done



**Output: -**



**Q.2 Write a program that takes a command-line argument n and prints the nth harmonic**

**number. Harmonic Number is of the form**

**Hn = 1/1+1/2+1/3+1/4+…...1/n**

**Code: -**

read -p "Enter a Number :- " n

i=1

sum=0

while [ $i -le $n ]

do

sum=`expr $sum + \( 10000 / $i \)`

i=`expr $i + 1`

done

echo "Harmonic Number (H$n) is :- "

i=1

while [ $i -le 5 ]

do

a=`echo $sum | cut -c $i`

echo -e "$a\c"

if [ $i -eq 1 ]

then

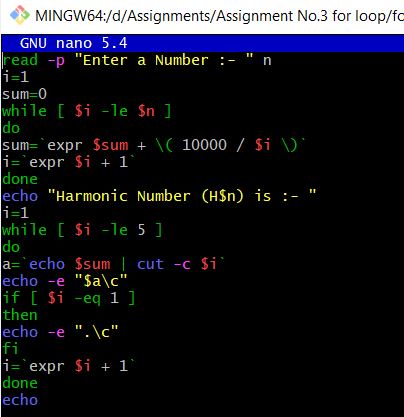
echo -e ".\c"

fi

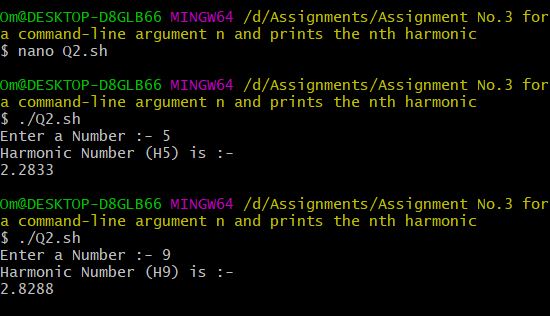
i=`expr $i + 1`

done

echo



**Output: -**



**Q.3 Write a program that takes a input and determines if the number is a prime.**

**Code: -**

read -p "Enter a Number:- " number

flag=0

b=2

temp=`expr $number / 2`

for (( i=$b; i<=$temp; i++))

do

if((`expr $number % $i` == 0 ))

then

flag=1

fi

done

if (($flag == 0 ))

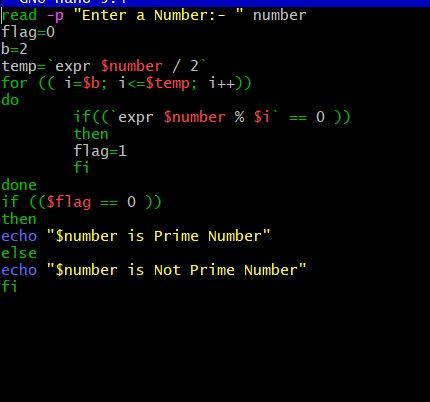
then

echo "$number is Prime Number"

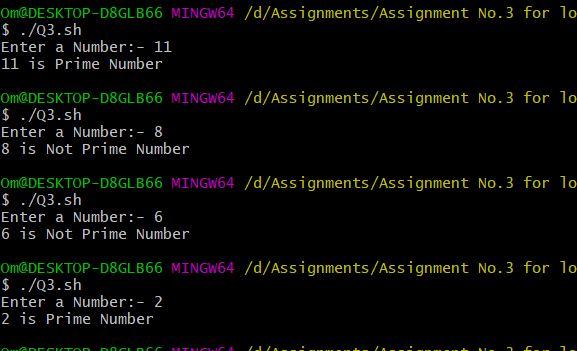
else

echo "$number is Not Prime Number"

fi



**Output: -**



**Q.4 Extend the program to take a range of number as input and output the Prime Numbers in that range.**

**Code: -**

function pn\_check()

{

num=$1

flag=0

m=2

temp=`expr $num / 2`

for (( i=$m; i<=$temp; i++))

do

if((`expr $num % $i` == 0 ))

then

flag=1

fi

done

if (( $flag == 0 ))

then

echo "$num"

fi

}

read -p "Enter First Number :- " Number\_1

read -p "Enter second Number :- " Number\_2

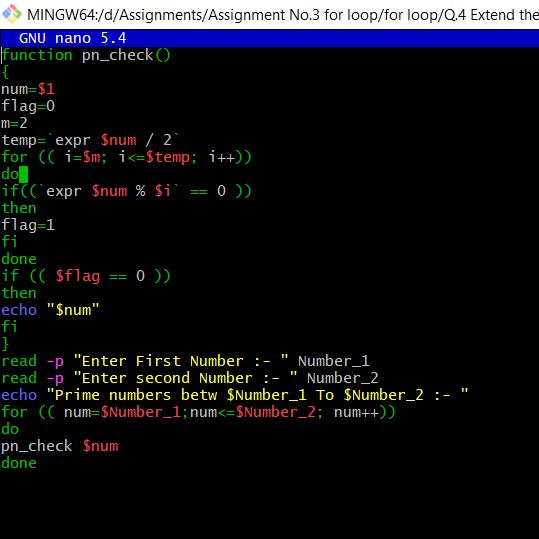
echo "Prime numbers betw $Number\_1 To $Number\_2 :- "

for (( num=$Number\_1;num<=$Number\_2; num++))

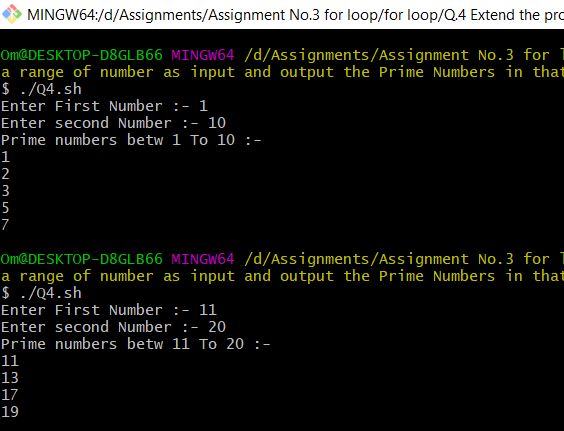
do

pn\_check $num

done



**Output: -**



**Q.5 Write a program that computes a factorial of a number taken as input. 5 Factorial – 5! = 1 \* 2 \* 3 \* 4 \* 5**

**Code: -**

read -p "Enter a Number:- " Number

res=1

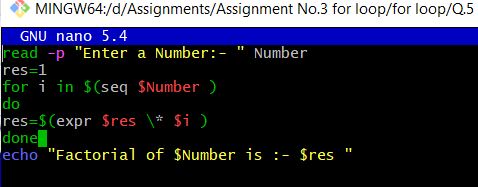
for i in $(seq $Number )

do

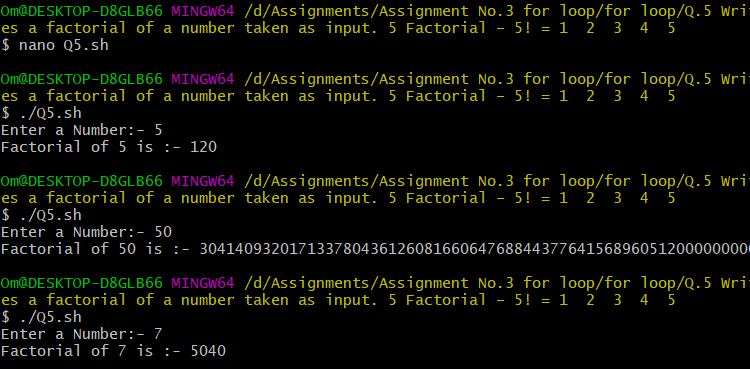
res=$(expr $res \\* $i )

done

echo "Factorial of $Number is :- $res "



**Output: -**



**Q.6 Write a program to compute Factors of a number N using prime factorization method. Logic -> Traverse till i\*i <= N instead of i <= N for efficiency. O/P -> Print the prime factors of number N**

**Code: -**

read -p "Enter a Number :- " Number

echo -ne "Prime Factors of $Number :-▼\n"

for i in $( seq `expr $Number / 2` )

do

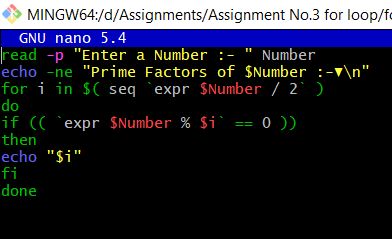
if (( `expr $Number % $i` == 0 ))

then

echo "$i"

fi

done



**Output: -**



**while loop**

**Q.1 Write a program that takes a command-line argument n and prints a table of the powers of 2 that are less than or equal to 2^n till 256 is reached.**

**Code: -**

read -p "Enter Power for 2^ Number ;- " Number

PowerOfTwo=1

count=0

LimitPowerTwo=256

while [ $Number -ne $count ]

do

PowerOfTwo=$(expr $PowerOfTwo \\* 2 )

count=$(expr $count + 1 )

done

if [ $PowerOfTwo -gt $LimitPowerTwo ]

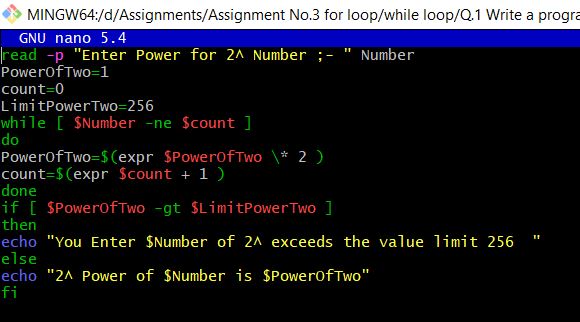
then

echo "You Enter $Number of 2^ exceeds the value limit 256 "

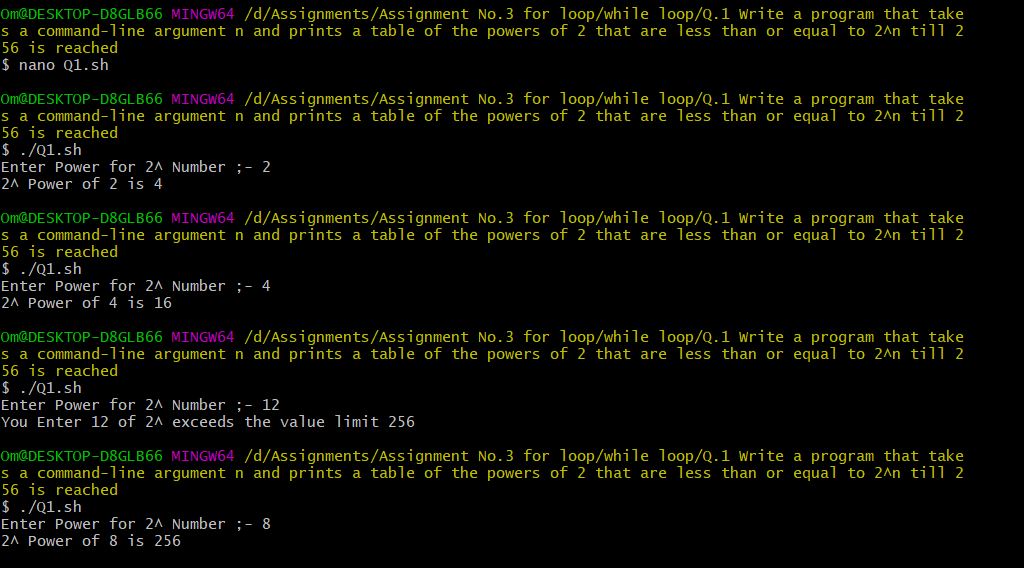
else

echo "2^ Power of $Number is $PowerOfTwo"

fi



**Output:-**



**Q.2 Find the Magic Number**

**a. Ask the user to think of a number n between 1 to 100**

**b. Then check with the user if the number is less then n/2 or greater**

**c. Repeat till the Magic Number is reached. . . .**

**Code: -**

read -p "Think of a Number :- " n

left=1

right=100

while [ $left -le $right ]

do

mid=$(( ($left + $right) / 2 ))

echo "Is Your Number " $mid "?" "True(1) False(0)"

read check

if [ $check -eq 1 ]

then

echo "☺☺☺☺ Congrats We found your Number"

break

else

echo "Is Your Number smaller than " $mid "True(1) False(0)"

read less

if [ $less -eq 1 ]

then

right=$(($mid-1))

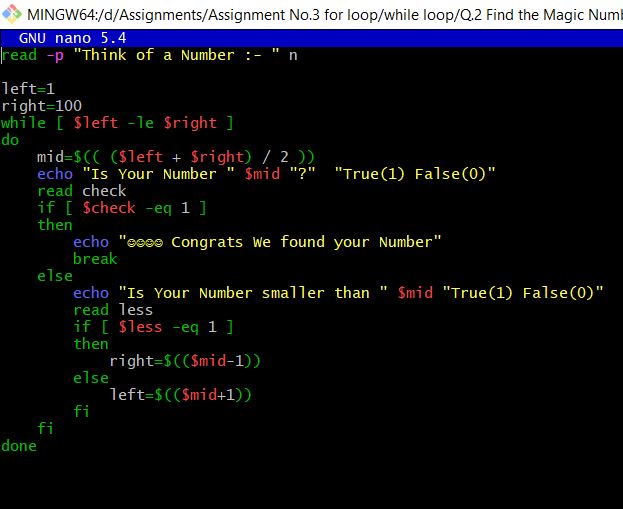
else

left=$(($mid+1))

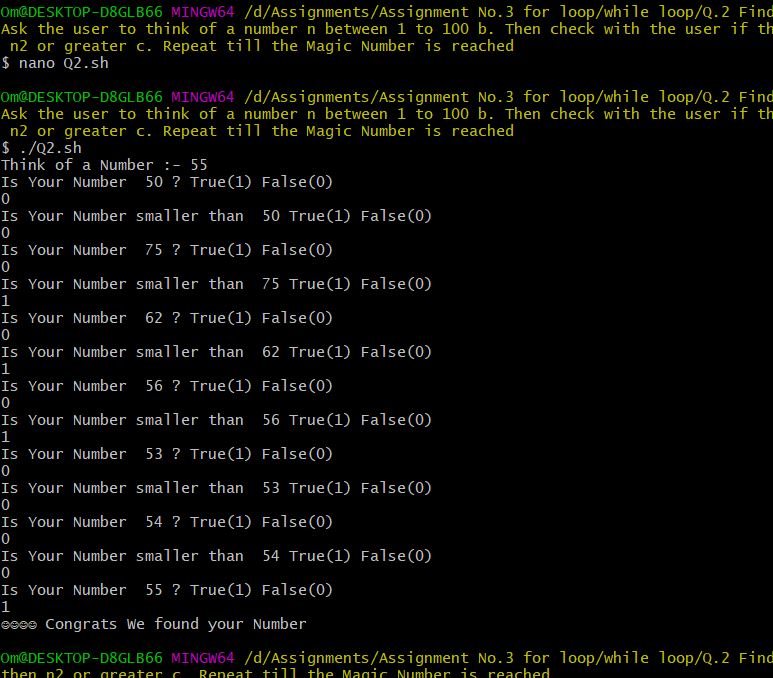
fi

fi

done



**Output: -**



**Q.3 Extend the Flip Coin problem till either Heads or Tails wins 11 times.**

**Code: -**

RANDOM=$$

Hcount=0

Tcount=0

i=1

b=11

while [ $Hcount -le 11 ] | [ $Tcount -le 11 ]

do

r1=$(expr $RANDOM % 10)

r2=$(expr $r1 % 2)

echo "⊛ Coin Tossed $i Times"

if [ $r2 -eq 0 ]

then

#echo -ne "Head \n"

Hcount=$(expr $Hcount + 1)

elif [ $r2 -eq 1 ]

then

#echo -ne "Tail \n"

Tcount=$(expr $Tcount + 1)

fi

if [ $Hcount -eq 11 ]

then

echo "☺☺☺☺☺☺☺☺ Head Wins 11 Times ☺☺☺☺☺☺☺☺"

break

elif [ $Tcount -eq 11 ]

then

echo "☺☺☺☺☺☺☺☺ Tails wins 11 Times ☺☺☺☺☺☺☺☺"

break

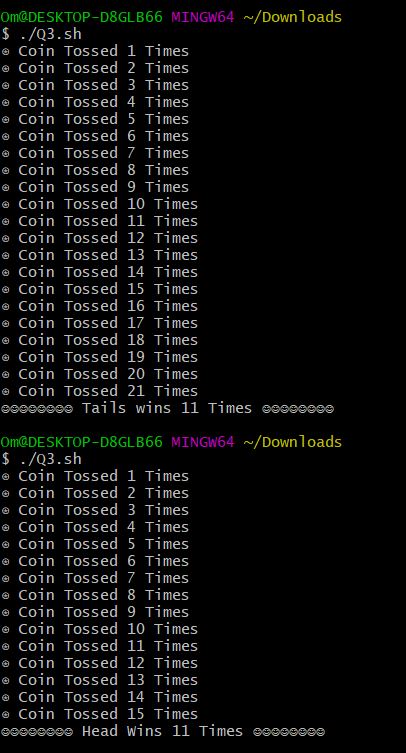
fi

i=$(expr $i + 1 )

done



**Output: -**



**Q.4 Write a Program where a gambler starts with Rs 100 and places Re 1 bet until he/she goes broke i.e. no more money to gamble or reaches the goal of Rs 200. Keeps track of number of times won and number of bets made.**

**Code: -**

read -p "Enter Amount to Start Play :- " am

temp=$(($am))

if [[ $am -gt 100 && $am -lt 200 ]]

then

Won=0

Loss=0

while(( $am > 0 && $am <=200 ))

do

Random\_Num=$(( $RANDOM%2 ))

case $Random\_Num in

1)

am=$(( $am + 1 ))

#echo -n "$am"

Won=$(( $Won + 1 ))

;;

0)

am=$(( $am -1 ))

Loss=$(( $Loss + 1 ))

;;

esac

done

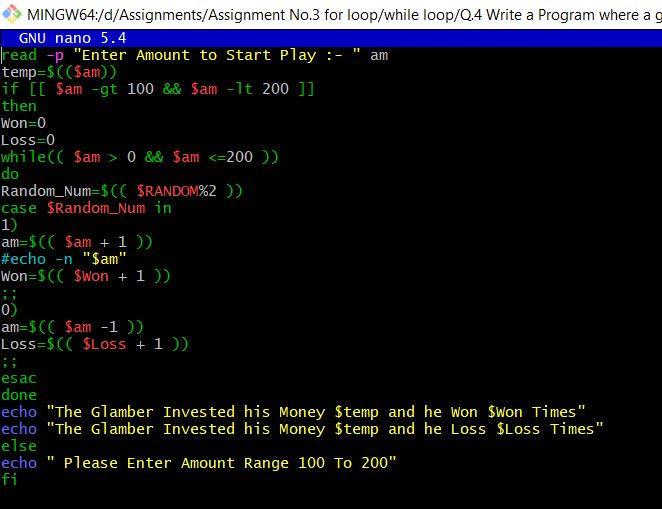
echo "The Glamber Invested his Money $temp and he Won $Won Times"

echo "The Glamber Invested his Money $temp and he Loss $Loss Times"

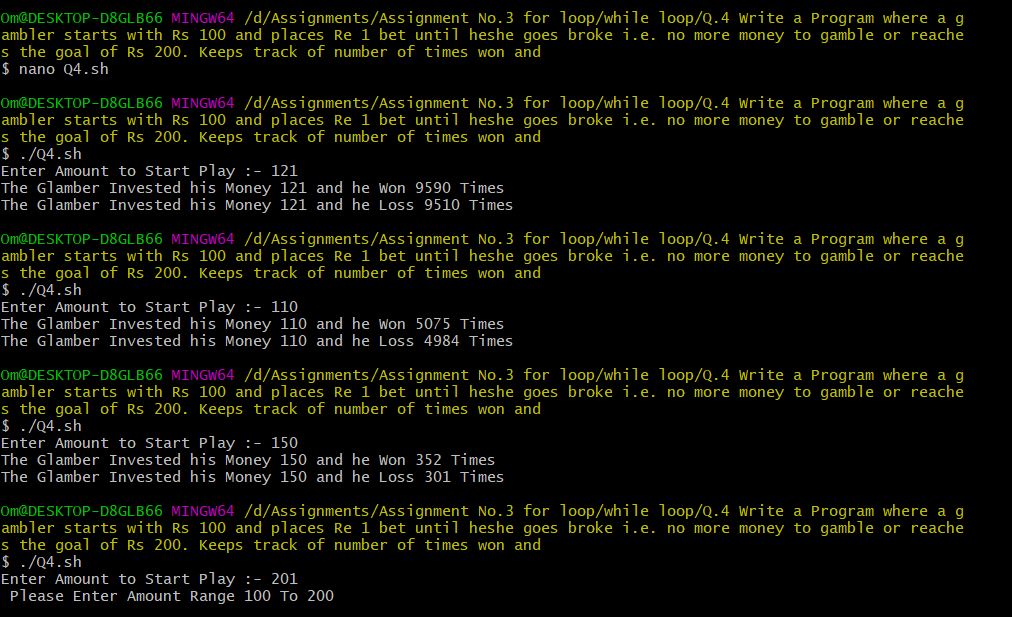
else

echo " Please Enter Amount Range 100 To 200"

fi



**Output: -**



**Function**

**Q.1 Help user find degF or degC based on their Conversion Selection. Use Case Statement and ensure that the inputs are within the Freezing Point ( 0 °C / 32 °F ) and the Boiling Point of Water**

**( 100 °C / 212 °F )**

**a. degF = (degC \* 9/5) + 32**

**b. degC = (degF – 32) \* 5/9**

**Code: -**

function degF()

{

temp=$((($c\*9/5)+32))

echo "Given Temprature in Degree Farhenite is $temp Farhenite"

}

function degC()

{

temp=$((($f-32)\*5/9))

echo "Given Temprature in Degree Celsius is $temp Celsius"

}

echo -ne "1.Degree Celsius To Farhenite \n2.Degree Farhenite To Degree Celcius \nEnter the coice:-"

read choice

case $choice in

1)

echo -ne "Enter Degree Celsius Temprature :- "

read c

degF $c

;;

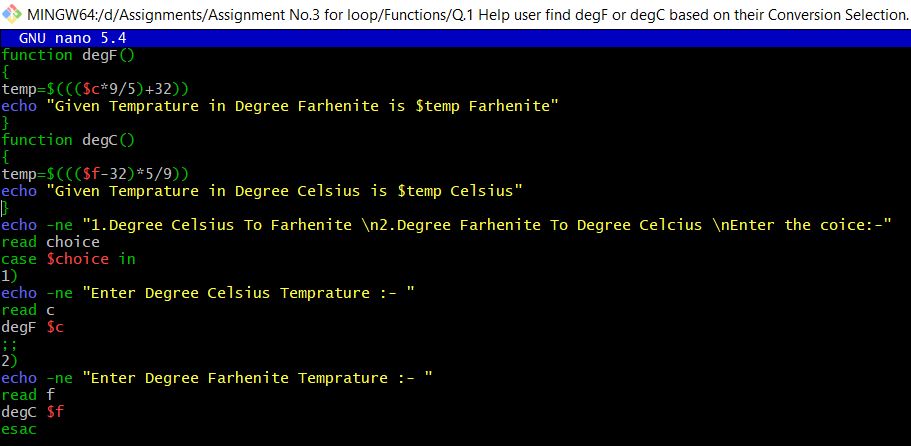
2)

echo -ne "Enter Degree Farhenite Temprature :- "

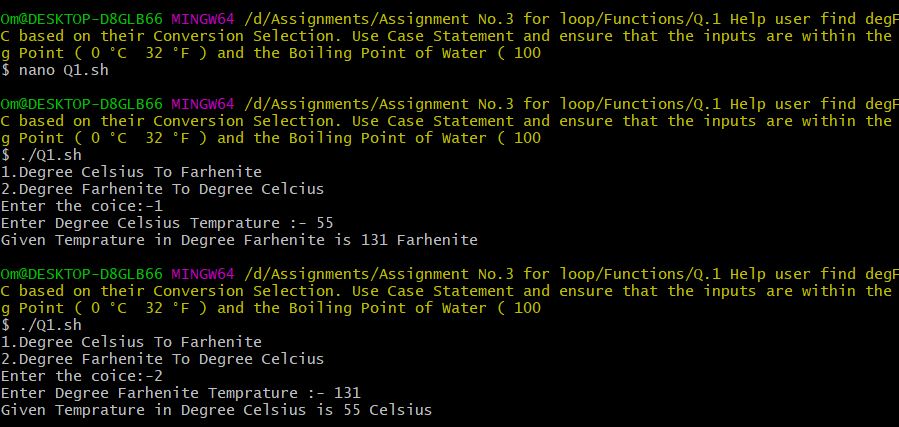
read f

degC $f

esac



**Output: -**



**Q.2 Write a function to check if the two numbers are Palindromes**

**Code: -**

function Palindrome()

{

Number\_1=$1

Number\_2=$2

Reverse=""

Temp=$Number\_1

sd=0

while [ $Temp -gt 0 ]

do

sd=$(( $Temp % 10 ))

Temp=$(( $Temp / 10 ))

Reverse=$( echo ${Reverse}${sd} )

done

if [ $Number\_2 -eq $Reverse ]

then

echo -ne "\nNumber $Number\_1 and $Number\_2 are Palindrome\n"

else

echo -ne "\nNumber $Number\_1 and $Nuber\_2 are Not Palindrome\n"

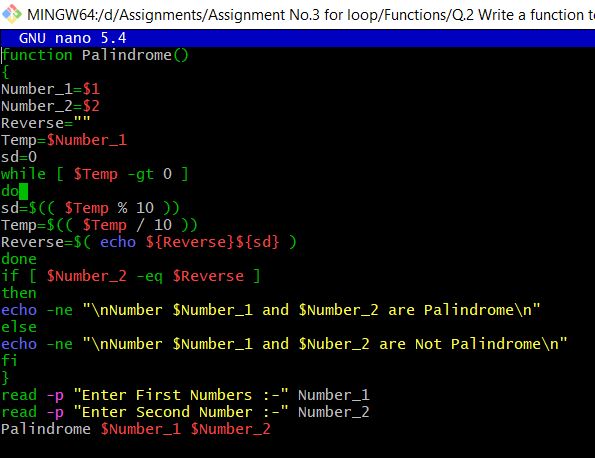
fi

}

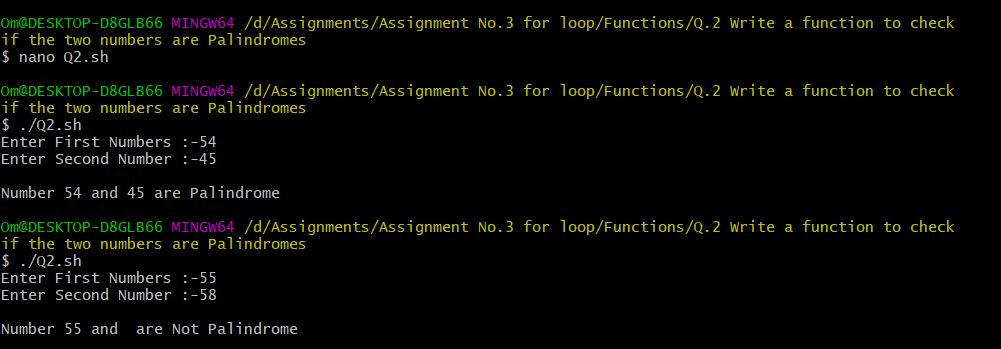
read -p "Enter First Numbers :-" Number\_1

read -p "Enter Second Number :-" Number\_2

Palindrome $Number\_1 $Number\_2



**Output: -**



**Q.3 Take a number from user and check if the number is a Prime then show that its palindrome is also prime**

**a. Write function check if number is Prime**

**b. Write function to get the Palindrome.**

**c. Check if the Palindrome number is also prime**

**Code: -**

function Palindrome()

{

Temp=$1

onum=$Temp

sd=0

Reverse=""

while [ $Temp -gt 0 ]

do

sd=$(( $Temp % 10 ))

Temp=$(( $Temp / 10 ))

Reverse=$( echo ${Reverse}${sd} )

done

if [ $onum -eq $Reverse ]

then

echo -ne "\nNumber $onum are Palindrome\n"

else

echo -ne "\nNumber $onum are Not Palindrome\n"

fi

}

function PrimeNumber()

{

Temp=$1

i=2

f=0

while [ $i -le `expr $Temp / 2` ]

do

if [ $(expr $Temp % $i) -eq 0 ]

then

f=1

fi

i=$(expr $i + 1 )

done

if [ $f -eq 1 ]

then

echo -ne "Number $Temp is Not Prime Number"

else

echo -ne "Number $Temp is Prime Number"

fi

}

echo -ne "1.To Check Prime Number \n2.To check Palindrome Number \n3.To check Prime as well as palondrome \nEnter Your Choice :- "

read choice

case $choice in

1)

read -p "Enter The Number :-" Temp

PrimeNumber $Temp

;;

2)

read -p "Enter The Number :-" Temp

Palindrome $Temp

;;

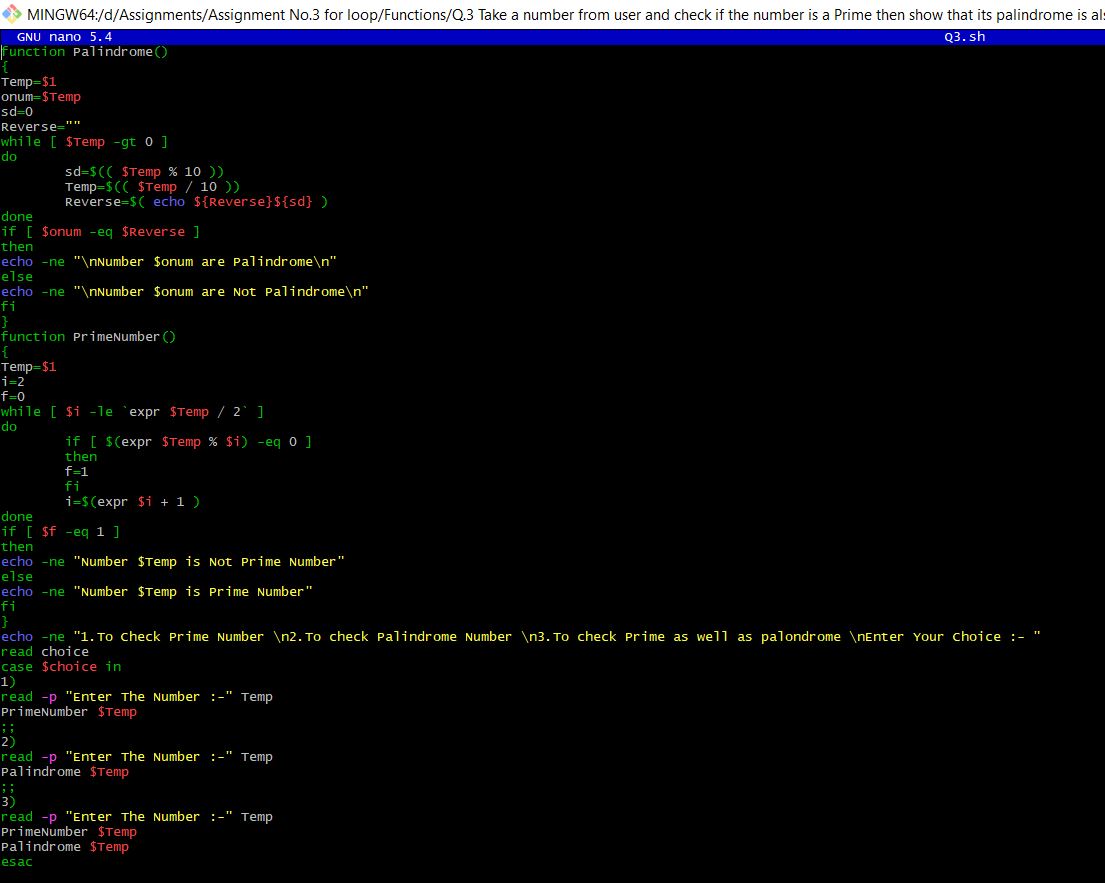
3)

read -p "Enter The Number :-" Temp

PrimeNumber $Temp

Palindrome $Temp

esac



**Output: -**

