**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.**

**Solution-**

#!/bin/bash -x

read -p "Enter the number" number

table=0

for ((i=0;i<number;i++))

do

table=$((table\*2))

echo $table

done

**Output-**

$ ./tableOfTwo.sh

+ read -p 'Enter the number' number

Enter the number5

+ table=0

+ (( i=0 ))

+ (( i<number ))

+ table=2

+ echo 2

2

+ (( i++ ))

+ (( i<number ))

+ table=4

+ echo 4

4

+ (( i++ ))

+ (( i<number ))

+ table=6

+ echo 6

6

+ (( i++ ))

+ (( i<number ))

+ table=8

+ echo 8

8

+ (( i++ ))

+ (( i<number ))

+ table=10

+ echo 10

10

+ (( i++ ))

+ (( i<number ))

**2] Write a program that takes a command-line argument n and prints the nth harmonic number. Harmonic Number is of the form**

**Solution-**

#!/bin/bash -x

read -p "Enter the number" number

harmonic=0

div=0

for ((i=1;i<=number;i++))

do

harmonic=$(($harmonic+(1000/$i)))

printf %.3f "$(($harmonic))e-3"

done

**Output-**

$ ./harmonicNumber.sh

+ read -p 'Enter the number' number

Enter the number3

+ harmonic=0

+ div=0

+ (( i=1 ))

+ (( i<=number ))

+ harmonic=1000

+ printf %.3f 1000e-3

1.000+ (( i++ ))

+ (( i<=number ))

+ harmonic=1500

+ printf %.3f 1500e-3

1.500+ (( i++ ))

+ (( i<=number ))

+ harmonic=1833

+ printf %.3f 1833e-3

1.833+ (( i++ ))

+ (( i<=number ))

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

**Solution-**

#!/bin/bash -x

read -p "Enter the number: " number

count=0

for ((i=2;i<number;i++))

do

if [[ $number%$i -eq 0 ]]

then

count=1

break

fi

done

if [[ count -eq 0 ]]

then

echo Prime number

else

echo NOt prime number

fi

**Output-**

$ ./primeNumber.sh

+ read -p 'Enter the number: ' number

Enter the number: 6

+ count=0

+ (( i=2 ))

+ (( i<number ))

+ =0

./primeNumber.sh: line 6: =0: command not found

+ [[ 6%2 -eq 0 ]]

+ count=1

+ break

+ echo 1

1

+ [[ count -eq 0 ]]

+ echo NOt prime number

NOt prime number

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

**Solution-**

#!/bin/bash -x

read -p "Enter the first number of range: " firstNum

read -p "Enter the last number of range: " lastNum

for (( i=$firstNum;i<$lastNum;i++ ))

do

count=0

for (( j=2;j<$i;j++ ))

do

if [[ $i%$j -eq 0 ]]

then

count=1;

break;

fi

done

if [[ count -eq 0 ]]

then

echo $i "is prime number."

fi

done

**Output-**

$ ./primeNumberRange.sh

+ read -p 'Enter the first number of range: ' firstNum

Enter the first number of range: 2

+ read -p 'Enter the last number of range: ' lastNum

Enter the last number of range: 6

+ (( i=2 ))

+ (( i<6 ))

+ count=0

+ (( j=2 ))

+ (( j<2 ))

+ [[ count -eq 0 ]]

+ echo 2 'is prime number'

2 is prime number

+ (( i++ ))

+ (( i<6 ))

+ count=0

+ (( j=2 ))

+ (( j<3 ))

+ [[ 3%2 -eq 0 ]]

+ (( j++ ))

+ (( j<3 ))

+ [[ count -eq 0 ]]

+ echo 3 'is prime number'

3 is prime number

+ (( i++ ))

+ (( i<6 ))

+ count=0

+ (( j=2 ))

+ (( j<4 ))

+ [[ 4%2 -eq 0 ]]

+ count=1

+ break

+ [[ count -eq 0 ]]

+ (( i++ ))

+ (( i<6 ))

+ count=0

+ (( j=2 ))

+ (( j<5 ))

+ [[ 5%2 -eq 0 ]]

+ (( j++ ))

+ (( j<5 ))

+ [[ 5%3 -eq 0 ]]

+ (( j++ ))

+ (( j<5 ))

+ [[ 5%4 -eq 0 ]]

+ (( j++ ))

+ (( j<5 ))

+ [[ count -eq 0 ]]

+ echo 5 'is prime number'

5 is prime number

+ (( i++ ))

+ (( i<6 ))

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

**Solution-**

#!/bin/bash -x

read -p "Enter the number " number

fact=1

for ((i=$number;i>1;i--))

do

fact=$(($fact\*$i))

done

echo $fact

**Output-**

$ ./factorial.sh

+ read -p 'Enter the number ' number

Enter the number 3

+ fact=1

+ (( i=3 ))

+ (( i>1 ))

+ fact=3

+ (( i-- ))

+ (( i>1 ))

+ fact=6

+ (( i-- ))

+ (( i>1 ))

+ echo 6

6

**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.**

**Solution-**

#!/bin/bash -x

read -p "Enter the first number : " number

prime=$number

for (( i=2;i<$number/2;i++ ))

do

count=0

for (( j=2;j<$i;j++ ))

do

if [[ $i%$j -eq 0 ]]

then

count=1;

break;

fi

done

if [[ count -eq 0 ]]

then

while [[ $prime%$i -eq 0 ]]

do

echo $i "prime fact"

prime=$(($prime/$i))

done

fi

done

**Output-**

$ ./primeFactors.sh

+ read -p 'Enter the first number : ' number

Enter the first number : 15

+ prime=15

+ (( i=2 ))

+ (( i<15/2 ))

+ count=0

+ (( j=2 ))

+ (( j<2 ))

+ [[ count -eq 0 ]]

+ [[ 15%2 -eq 0 ]]

+ (( i++ ))

+ (( i<15/2 ))

+ count=0

+ (( j=2 ))

+ (( j<3 ))

+ [[ 3%2 -eq 0 ]]

+ (( j++ ))

+ (( j<3 ))

+ [[ count -eq 0 ]]

+ [[ 15%3 -eq 0 ]]

+ echo 3 'prime fact'

3 prime fact

+ prime=5

+ [[ 5%3 -eq 0 ]]

+ (( i++ ))

+ (( i<15/2 ))

+ count=0

+ (( j=2 ))

+ (( j<4 ))

+ [[ 4%2 -eq 0 ]]

+ count=1

+ break

+ [[ count -eq 0 ]]

+ (( i++ ))

+ (( i<15/2 ))

+ count=0

+ (( j=2 ))

+ (( j<5 ))

+ [[ 5%2 -eq 0 ]]

+ (( j++ ))

+ (( j<5 ))

+ [[ 5%3 -eq 0 ]]

+ (( j++ ))

+ (( j<5 ))

+ [[ 5%4 -eq 0 ]]

+ (( j++ ))

+ (( j<5 ))

+ [[ count -eq 0 ]]

+ [[ 5%5 -eq 0 ]]

+ echo 5 'prime fact'

5 prime fact

+ prime=1

+ [[ 1%5 -eq 0 ]]

+ (( i++ ))

+ (( i<15/2 ))

+ count=0

+ (( j=2 ))

+ (( j<6 ))

+ [[ 6%2 -eq 0 ]]

+ count=1

+ break

+ [[ count -eq 0 ]]

+ (( i++ ))

+ (( i<15/2 ))

**WHILE LOOP**

**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..**

**Solution-**

#!/bin/bash -x

read -p "Enter the number: " number

table=1

count=0

while [ $table -ne 256 ]

do

table=$(($table\*2))

((count++))

echo $table

if [[ $count -eq number ]]

then

break

fi

done

**Output-**

$ ./powerOfTwoTable.sh

+ read -p 'Enter the number: ' number

Enter the number: 3

+ table=1

+ count=0

+ '[' 1 -ne 256 ']'

+ table=2

+ count=1

+ echo 2

2

+ [[ 1 -eq number ]]

+ '[' 2 -ne 256 ']'

+ table=4

+ count=2

+ echo 4

4

+ [[ 2 -eq number ]]

+ '[' 4 -ne 256 ']'

+ table=8

+ count=3

+ echo 8

8

+ [[ 3 -eq number ]]

+ break

**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..**

**Solution-**

**Output-**

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

**Solution-**

#!/bin/bash -x

head=0

tail=0

while [[ $tail -ne 11 || $head -ne 11 ]]

do

flip=$((RANDOM%2))

if [[ $flip -eq 0 ]]

then

echo "Head: "$head

((head++))

else

echo "Tail: "$tail

((tail++))

fi

if [[ $head -eq 11 || $tail -eq 11 ]]

then

break

fi

done

echo $head

echo $tail

Output-

$ ./flipCoin.sh

+ head=0

+ tail=0

+ [[ 0 -ne 11 ]]

+ flip=1

+ [[ 1 -eq 0 ]]

+ echo 'Tail: 0'

Tail: 0

+ tail=1

+ [[ 0 -eq 11 ]]

+ [[ 1 -eq 11 ]]

+ [[ 1 -ne 11 ]]

+ flip=0

+ [[ 0 -eq 0 ]]

+ echo 'Head: 0'

Head: 0

+ head=1

+ [[ 1 -eq 11 ]]

+ [[ 1 -eq 11 ]]

+ [[ 1 -ne 11 ]]

+ flip=0

+ [[ 0 -eq 0 ]]

+ echo 'Head: 1'

Head: 1

+ head=2

+ [[ 2 -eq 11 ]]

+ [[ 1 -eq 11 ]]

+ [[ 1 -ne 11 ]]

+ flip=1

+ [[ 1 -eq 0 ]]

+ echo 'Tail: 1'

Tail: 1

+ tail=2

+ [[ 2 -eq 11 ]]

+ [[ 2 -eq 11 ]]

+ [[ 2 -ne 11 ]]

+ flip=0

+ [[ 0 -eq 0 ]]

+ echo 'Head: 2'

Head: 2

+ head=3

+ [[ 3 -eq 11 ]]

+ [[ 2 -eq 11 ]]

+ [[ 2 -ne 11 ]]

+ flip=1

+ [[ 1 -eq 0 ]]

+ echo 'Tail: 2'

Tail: 2

+ tail=3

+ [[ 3 -eq 11 ]]

+ [[ 3 -eq 11 ]]

+ [[ 3 -ne 11 ]]

+ flip=1

+ [[ 1 -eq 0 ]]

+ echo 'Tail: 3'

Tail: 3

+ tail=4

+ [[ 3 -eq 11 ]]

+ [[ 4 -eq 11 ]]

+ [[ 4 -ne 11 ]]

+ flip=0

+ [[ 0 -eq 0 ]]

+ echo 'Head: 3'

Head: 3

+ head=4

+ [[ 4 -eq 11 ]]

+ [[ 4 -eq 11 ]]

+ [[ 4 -ne 11 ]]

+ flip=0

+ [[ 0 -eq 0 ]]

+ echo 'Head: 4'

Head: 4

+ head=5

+ [[ 5 -eq 11 ]]

+ [[ 4 -eq 11 ]]

+ [[ 4 -ne 11 ]]

+ flip=0

+ [[ 0 -eq 0 ]]

+ echo 'Head: 5'

Head: 5

+ head=6

+ [[ 6 -eq 11 ]]

+ [[ 4 -eq 11 ]]

+ [[ 4 -ne 11 ]]

+ flip=0

+ [[ 0 -eq 0 ]]

+ echo 'Head: 6'

Head: 6

+ head=7

+ [[ 7 -eq 11 ]]

+ [[ 4 -eq 11 ]]

+ [[ 4 -ne 11 ]]

+ flip=0

+ [[ 0 -eq 0 ]]

+ echo 'Head: 7'

Head: 7

+ head=8

+ [[ 8 -eq 11 ]]

+ [[ 4 -eq 11 ]]

+ [[ 4 -ne 11 ]]

+ flip=0

+ [[ 0 -eq 0 ]]

+ echo 'Head: 8'

Head: 8

+ head=9

+ [[ 9 -eq 11 ]]

+ [[ 4 -eq 11 ]]

+ [[ 4 -ne 11 ]]

+ flip=1

+ [[ 1 -eq 0 ]]

+ echo 'Tail: 4'

Tail: 4

+ tail=5

+ [[ 9 -eq 11 ]]

+ [[ 5 -eq 11 ]]

+ [[ 5 -ne 11 ]]

+ flip=0

+ [[ 0 -eq 0 ]]

+ echo 'Head: 9'

Head: 9

+ head=10

+ [[ 10 -eq 11 ]]

+ [[ 5 -eq 11 ]]

+ [[ 5 -ne 11 ]]

+ flip=0

+ [[ 0 -eq 0 ]]

+ echo 'Head: 10'

Head: 10

+ head=11

+ [[ 11 -eq 11 ]]

+ break

**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.**

**Solution-**

#!/bin/bash -x

money=100

won=0

lose=0

random=0

bet=0

while [[ money -ne 200 ]]

do

random=$((RANDOM%2))

if [[ $random -eq 0 ]]

then

lose=$(($lose+1))

money=$(($money-1))

else

win=$(($win+1))

money=$(($money+1))

fi

if [[ $money -eq 0 ]]

then

then

break

fi

bet=$(($bet+1))

done

**Functions Practice Problems**

**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**

**Solution-**

#!/bin/bash -x

echo "1] Fahrenheit (32F-212F) to celsius 2] Celsius(0C-100C) to fehrenheit"

read -p "Enter the choice " choice

read -p "Enter the temperature " temp

conversion=0

if [[ ($choice -eq 1 && ($temp -ge 32 && $temp -le 212)) ||

($choice -eq 2 && ($temp -ge 0 && $temp -le 100)) ]]

then

function ferenhit() {

degF=$((($temp\*9000/5)+32))

echo "Temperature in farenheit: "

printf %.3f "$((degF))e-3"

}

function celsius() {

degC=$((($temp-32)\*5000/9))

echo "Temperature in celsius is: "

printf %.3f "$((degC))e-3"

}

case $choice in

1)

echo $temp

conversion="$( celsius $temp )"

;;

2)

conversion="$( ferenhit $temp )"

;;

\*)

echo "Enter the correct choice"

;;

esac

else

echo "Give correct temperature"

fi

**Output-**

$ ./temperatureConv.sh

+ echo '1] Fahrenheit to celsius 2] Celsius to fehrenheit'

1] Fahrenheit to celsius 2] Celsius to fehrenheit

+ read -p 'Enter the choice ' choice

Enter the choice 1

+ read -p 'Enter the temperature' temp

Enter the temperature108

+ conversion=0

+ [[ 1 -eq 1 ]]

+ [[ 108 -ge 32 ]]

+ [[ 108 -le 212 ]]

+ case $choice in

++ celsius 108

++ degC=42222

++ echo 'Temperature in celsius is: '

++ printf %.3f 42222e-3

+ conversion='Temperature in celsius is:

42.222'

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

**Solution-**

#!/bin/bash -x

read -p "Enter the first number: " firstNumber

read -p "Enter the second number: " secondNumber

function palindrome() {

local number=$1

reverse=0

num=$number

while [ $num -gt 0 ]

do

add=$num%10

reverse=$((($reverse\*10)+add))

num=$(($num/10))

done

if [[ $number -eq $reverse ]]

then

echo Number $number is palindrome

else

echo Number $number is not palindrome

fi

}

check="$( palindrome $firstNumber )"

check="$( palindrome $secondNumber)

**Output-**

$ ./palindrome.sh

+ read -p 'Enter the first number: ' firstNumber

Enter the first number: 13

+ read -p 'Enter the second number: ' secondNumber

Enter the second number: 121

++ palindrome 13

++ local number=13

++ reverse=0

++ num=13

++ '[' 13 -gt 0 ']'

++ add=13%10

++ reverse=3

++ num=1

++ '[' 1 -gt 0 ']'

++ add=1%10

++ reverse=31

++ num=0

++ '[' 0 -gt 0 ']'

++ [[ 13 -eq 31 ]]

++ echo Number 13 is not palindrome

+ check='Number 13 is not palindrome'

++ palindrome 121

++ local number=121

++ reverse=0

++ num=121

++ '[' 121 -gt 0 ']'

++ add=121%10

++ reverse=1

++ num=12

++ '[' 12 -gt 0 ']'

++ add=12%10

++ reverse=12

++ num=1

++ '[' 1 -gt 0 ']'

++ add=1%10

++ reverse=121

++ num=0

++ '[' 0 -gt 0 ']'

++ [[ 121 -eq 121 ]]

++ echo Number 121 is palindrome

+ check='Number 121 is palindrom

**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**

**Solution-**

#!/bin/bash -x

read -p "Enter the number: " number

function prime() {

count=0

for ((i=2;i<$number;i++))

do

if [[ $number%$i -eq 0 ]]

then

count=1

break

fi

done

echo $count

if [[ count -eq 0 ]]

then

echo "The number is prime"

else

echo "The number is not prime"

fi

}

function palindrome() {

reverse=0

num=$number

while [ $num -gt 0 ]

do

add=$num%10

reverse=$((($reverse\*10)+add))

num=$(($num/10))

done

if [[ $number -eq $reverse ]]

then

echo "Number is palindrome"

else

echo "The number is not palindrome"

fi

}

check="$( palindrome $number )"

check="$( prime $number )"

**Output-**

$ ./primePalindrome.sh

+ read -p 'Enter the number: ' number

Enter the number: 11

++ palindrome 11

++ reverse=0

++ num=11

++ '[' 11 -gt 0 ']'

++ add=11%10

++ reverse=1

++ num=1

++ '[' 1 -gt 0 ']'

++ add=1%10

++ reverse=11

++ num=0

++ '[' 0 -gt 0 ']'

++ [[ 11 -eq 11 ]]

++ echo 'Number is palindrome'

+ check='Number is palindrome'

++ prime 11

++ count=0

++ (( i=2 ))

++ (( i<11 ))

++ [[ 11%2 -eq 0 ]]

++ (( i++ ))

++ (( i<11 ))

++ [[ 11%3 -eq 0 ]]

++ (( i++ ))

++ (( i<11 ))

++ [[ 11%4 -eq 0 ]]

++ (( i++ ))

++ (( i<11 ))

++ [[ 11%5 -eq 0 ]]

++ (( i++ ))

++ (( i<11 ))

++ [[ 11%6 -eq 0 ]]

++ (( i++ ))

++ (( i<11 ))

++ [[ 11%7 -eq 0 ]]

++ (( i++ ))

++ (( i<11 ))

++ [[ 11%8 -eq 0 ]]

++ (( i++ ))

++ (( i<11 ))

++ [[ 11%9 -eq 0 ]]

++ (( i++ ))

++ (( i<11 ))

++ [[ 11%10 -eq 0 ]]

++ (( i++ ))

++ (( i<11 ))

++ echo 0

++ [[ count -eq 0 ]]

++ echo 'The number is prime'

+ check='0

The number is prime'