REPITITION PRACTICE PROBLEMS WITH FOR LOOP

Problem 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: nano for1.sh

```
#!/bin/bash
echo "Enter the value of n: "
read n
for (( count=1; count<=n; count++ ))
do
        x=$((2**$count))
        echo "$count x $x"
done
```

Output: chmod +x for1.sh

```
Hp@DESKTOP-0AFPT6H MINGW64 ~/Desktop/bridgelabz
$  ./for1.sh
Enter the value of n:
5
1 x 2
2 x 4
3 x 8
4 x 16
5 x 32

Hp@DESKTOP-0AFPT6H MINGW64 ~/Desktop/bridgelabz
$  ./for1.sh
Enter the value of n:
8
1 x 2
2 x 4
3 x 8
4 x 16
5 x 32
6 x 64
7 x 128
8 x 256
```

Problem 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+$\frac{1}{2}+\frac{1}{3}+\frac{1}{4}+\frac{1}{5}+\cdots\ldots\ldots\ldots\frac{1}{n}$

Solution: nano for2.sh

```
#!/bin/bash
echo "Enter the value of n: "
read n
x=1

for (( count=1; count<=n; count++ ))
do
        num=`expr 1/$count`
        x=`echo "$x+$num "`
done
echo $x
```

Output: chmod +x for2.sh

```
Hp@DESKTOP-0AFPT6H MINGW64 ~/Desktop/bridgelabz
$  ./for2.sh
Enter the value of n:
5
1+ 1/1 + 1/2 + 1/3 + 1/4 + 1/5
```

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

Solution: nano six3.sh

```
#!/bin/bash
echo "Enter the number:"
read n
for (( count=1; count<=n; count++ ))
do
        x=`expr $n % 2`
        if [ $x -ne 0 ]
        then
                echo "It is prime number"
                        break
        else
                echo "Not a prime number"
                        break
        fi
done
```

Output: chmod +x six3.sh

```
Hp@DESKTOP-0AFPT6H MINGW64 ~/Desktop/bridgelabz
$  ./six3.sh
Enter the number:
2
Not a prime number

Hp@DESKTOP-0AFPT6H MINGW64 ~/Desktop/bridgelabz
$  ./six3.sh
Enter the number:
5
It is prime number

Hp@DESKTOP-0AFPT6H MINGW64 ~/Desktop/bridgelabz
$  ./six3.sh
Enter the number:
78
Not a prime number
```

Problem 4] Extend the program to take a range of a number as input and output the prime numbers in that range.

Solution: nano six4.sh

```
#!/bin/bash
echo "Enter the range:"
read n
x=2
echo "Prime numbers till $n are: "
for (( i=$x; i<=n; i++ ))
   do
   flag=0
   for (( j=2; j<=$i-1; j++ ))
      do
         if [ `expr $i % $j` -eq 0 ]
         then
            flag=1
            break
         fi
      done
      if [ $flag -eq 0 ]
         then
            echo $i
      fi
   done
```

Output: chmod +x six4.sh

```
Hp@DESKTOP-0AFPT6H MINGW64 ~/Desktop/bridgelabz
$  ./six4.sh
Enter the range:
5
Prime numbers till 5 are:
2
3
5

Hp@DESKTOP-0AFPT6H MINGW64 ~/Desktop/bridgelabz
$  ./six4.sh
Enter the range:
45
Prime numbers till 45 are:
2
3
5
7
11
13
17
19
23
29
31
37
41
43
```

Problem 5] Write a program that computes a factorial of a number taken as an input.

Solution: nano six5.sh

```
#!/bin/bash
echo "Enter the number to find the factorial: "
read n
fact=1
for (( count=1; count<=$n; count++ ))
do
   fact=$(( $fact * $count ))
done
echo  "The factorial of $n is: " $fact
```

Output: chmod +x six5.sh

```
Hp@DESKTOP-0AFPT6H MINGW64 ~/Desktop/bridgelabz
$  ./six5.sh
Enter the number to find the factorial:
5
The factorial of 5 is: 120
```

Problem 6] Write a program to compute factors of a number N using prime factorization method.

Solution: nano six6.sh

```
#!/bin/bash
echo "Enter the number:"
read n
echo "Prime factors of $n are:"
for (( count=2; count<=n; count++ ))
do
   while [ $((n%count)) -eq 0 ]
   do
      echo $count
      n=$((n/$count))
   done
done
```

Output: chmod +x six6.sh

```
Hp@DESKTOP-0AFPT6H MINGW64 ~/Desktop/bridgelabz
$  ./six6.sh
Enter the number:
40
Prime factors of 40 are:
2
2
2
5

Hp@DESKTOP-0AFPT6H MINGW64 ~/Desktop/bridgelabz
$  ./six6.sh
Enter the number:
79
Prime factors of 79 are:
79

Hp@DESKTOP-0AFPT6H MINGW64 ~/Desktop/bridgelabz
$  ./six6.sh
Enter the number:
60
Prime factors of 60 are:
2
2
3
5
```

REPETITION PRACTICE PROBLEMS WITH WHILE LOOP

Problem 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: nano while1.sh

```
#!/bin/bash
echo "Enter the value of n: "
read n
count=0
num=1
while [ $n -ne $count ]
do
   num=$(expr $num \* 2)
   count=$(expr $count + 1)
done
if [ $num -gt 256 ]
then
   echo "Exceeds the value beyond 256"
else
   echo "2^$count is $num"
fi
```

Output: chmod +x while1.sh

```
Hp@DESKTOP-0AFPT6H MINGW64 ~/Desktop/bridgelabz
$  ./while1.sh
Enter the value of n:
5
2^5 is 32

Hp@DESKTOP-0AFPT6H MINGW64 ~/Desktop/bridgelabz
$  ./while1.sh
Enter the value of n:
6
2^6 is 64

Hp@DESKTOP-0AFPT6H MINGW64 ~/Desktop/bridgelabz
$  ./while1.sh
Enter the value of n:
8
2^8 is 256
```

```
Hp@DESKTOP-0AFPT6H MINGW64 ~/Desktop/bridgelabz
$  ./while1.sh
Enter the value of n:
9
Exceeds the value beyond 256
```

Problem 2] Find the magic number

a) **Ask the user to think of a number n between 1 and 100**
b) **Then check with the user whether the number is less than n/2 or greater.**
c) **Repeat till the magic number is reached.**

Solution: nano while2.sh

```
#!/bin/bash
echo "Think any number between 1 to 100"
min=1
max=100
while [ $min -le $max ]
do
        mid=$((($min+$max)/2))
        echo "Your guess is $mid"
        echo " If $mid is not your guessed number then "
        echo "Enter 1 for less than $mid value"
        echo "Enter 2 for more than $mid value"
        echo "Enter 3 for correct guess"
        read n
        if [ $n -eq 1 ]
        then
                max=$((mid-1))
        elif [ $n -eq 2 ]
        then
                min=$((mid+1))
        elif [ $n -eq 3 ]
        then
                exit
        else
                echo "Invalid input"
        fi
done
        echo "You have guessed $mid"
```

Output: chmod +x while2.sh

```
Hp@DESKTOP-0AFPT6H MINGW64 ~/Desktop/bridgelabz
$  ./while2.sh
Think any number between 1 to 100
Your guess is 50
 If 50 is not your guessed number then
Enter 1 for less than 50 value
Enter 2 for more than 50 value
Enter 3 for correct guess
2
Your guess is 75
 If 75 is not your guessed number then
Enter 1 for less than 75 value
Enter 2 for more than 75 value
Enter 3 for correct guess
2
Your guess is 88
 If 88 is not your guessed number then
Enter 1 for less than 88 value
Enter 2 for more than 88 value
Enter 3 for correct guess
2
Your guess is 94
 If 94 is not your guessed number then
Enter 1 for less than 94 value
Enter 2 for more than 94 value
Enter 3 for correct guess
1
```

```
Your guess is 91
 If 91 is not your guessed number then
Enter 1 for less than 91 value
Enter 2 for more than 91 value
Enter 3 for correct guess
1
Your guess is 89
 If 89 is not your guessed number then
Enter 1 for less than 89 value
Enter 2 for more than 89 value
Enter 3 for correct guess
2
Your guess is 90
 If 90 is not your guessed number then
Enter 1 for less than 90 value
Enter 2 for more than 90 value
Enter 3 for correct guess
3
```

Problem 3] Extend the Flip coin problem till either Heads or Tails wins 11 times.
Solution: nano while3.sh

```
#!/bin/bash
echo "Toss the coin"
head=0
tail=0
while [ $head -lt 11 ] && [ $tail -lt 11 ]
do
        f=$((RANDOM%2))
        if [ $f -eq 0 ]
        then
                ((head++))
                echo "HEADS: $head"
        else
                ((tail++))
                echo "TAILS: $tail"
        fi
done
if [ $head -ge 11 ]
then
        echo "HEADS is the winner"
elif [ $tail -ge 11 ]
then
        echo "TAILS is the winner"
else
        exit
fi
```

Output: chmod +x while3.sh

```
Hp@DESKTOP-0AFPT6H MINGW64 ~/Desktop/bridgelabz
$  ./while3.sh
Toss the coin
TAILS: 1
TAILS: 2
TAILS: 3
HEADS: 1
TAILS: 4
TAILS: 5
HEADS: 2
HEADS: 3
TAILS: 6
HEADS: 4
HEADS: 5
HEADS: 6
TAILS: 7
TAILS: 8
HEADS: 7
TAILS: 9
HEADS: 8
HEADS: 9
TAILS: 10
HEADS: 10
HEADS: 11
HEADS is the winner

Hp@DESKTOP-0AFPT6H MINGW64 ~/Desktop/bridgelabz
$  ./while3.sh
Toss the coin
```

```
TAILS: 1
HEADS: 1
HEADS: 2
TAILS: 2
HEADS: 3
HEADS: 4
HEADS: 5
HEADS: 6
HEADS: 7
TAILS: 3
TAILS: 4
TAILS: 5
TAILS: 6
HEADS: 8
HEADS: 9
HEADS: 10
HEADS: 11
HEADS is the winner

Hp@DESKTOP-0AFPT6H MINGW64 ~/Desktop/bridgelabz
Toss the coin
TAILS: 1
TAILS: 2
TAILS: 3
HEADS: 1
TAILS: 4
HEADS: 2
TAILS: 5
TAILS: 6
HEADS: 3
TAILS: 7
HEADS: 4
HEADS: 5
HEADS: 6
TAILS: 8
HEADS: 7
HEADS: 8
HEADS: 9
TAILS: 9
TAILS: 10
TAILS: 11
TAILS is the winner
```

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

Solution: nano while4.sh

```
#!/bin/bash
random=$((RANDOM))

hCount=100
lCount=100

echo "The game is started and gambler has invested Rs 100"
while [[ $hCount -le 200 && $lCount -ge 1 ]]
do
   r1=$(( $random % 10 ))
   r2=$(( $r1 % 2 ))

   if [ $r2 -eq 0 ]
   then
      hCount=$(expr $hCount + 1)
   elif [ $r2 -eq 1 ]
   then
      lCount=$(expr $lCount - 1)
   else
      echo "dummy"
   fi

   if [[ $hCount -ge 200 ]]
   then
      echo "Gambler won and reached the goal of 200"
      break
   elif [ $lCount -eq 1 ]
   then
      echo "Gambler lost all money and broke"
```

```
      break
   fi
done
```

Output :
```
Hp@DESKTOP-0AFPT6H MINGW64 ~/Desktop/bridgelabz
$  ./while4.sh
The game is started and gambler has invested Rs 100
Gambler lost all money and broke

Hp@DESKTOP-0AFPT6H MINGW64 ~/Desktop/bridgelabz
$  ./while4.sh
The game is started and gambler has invested Rs 100
Gambler lost all money and broke

Hp@DESKTOP-0AFPT6H MINGW64 ~/Desktop/bridgelabz
$  ./while4.sh
The game is started and gambler has invested Rs 100
Gambler lost all money and broke

Hp@DESKTOP-0AFPT6H MINGW64 ~/Desktop/bridgelabz
$  ./while4.sh
The game is started and gambler has invested Rs 100
Gambler won and reached the goal of 200

Hp@DESKTOP-0AFPT6H MINGW64 ~/Desktop/bridgelabz
$  ./while4.sh
The game is started and gambler has invested Rs 100
Gambler won and reached the goal of 200

Hp@DESKTOP-0AFPT6H MINGW64 ~/Desktop/bridgelabz
$  ./while4.sh
The game is started and gambler has invested Rs 100
Gambler lost all money and broke
```

FUNCTIONS PRACTICE PROBLEMS

Problem 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) **dedF = (degC*9/5)+32**
b) **degC= (degF-32)*5/9**

Solution: nano func1.sh

```
#!/bin/bash
temperatureconv(){
case $n in
        "1") echo "Enter temperature in Celsius:"
             read cel
             if [ $cel -ge 0 ] && [ $cel -le 100 ]
             then
                farentemp=`(awk "BEGIN{print $cel * 1.8+32}")`
                echo "Temperature in farenheit is $farentemp"
             else
                echo "Please enter valid value"
             fi
             ;;
        "2") echo "Enter temperature in Farenheit:"
             read far
             if [ $far -ge 32 ] && [ $far -le 212 ]
             then
                celsiustemp=`(awk "BEGIN{print ($far - 32)*0.555}")`
                echo "Temperature in celsius is $celsiustemp"
             else
                echo "Enter the valid value"
             fi
             ;;
             *) echo "Invalid input"
esac
}
```

```
echo "Choose the option: 1. Celsius to Farenheit, 2. Farenheit to Celsius"
read n
temperatureconv $n
```

Output: chmod +x func1.sh

```
Hp@DESKTOP-0AFPT6H MINGW64 ~/Desktop/bridgelabz
$  ./func1.sh
Choose the option: 1. Celsius to Farenheit, 2. Farenheit to Celsius
1
Enter temperature in Celsius:
75
Temperature in farenheit is 167

Hp@DESKTOP-0AFPT6H MINGW64 ~/Desktop/bridgelabz
$  ./func1.sh
Choose the option: 1. Celsius to Farenheit, 2. Farenheit to Celsius
2
Enter temperature in Farenheit:
195
Temperature in celsius is 90.465

Hp@DESKTOP-0AFPT6H MINGW64 ~/Desktop/bridgelabz
$  ./func1.sh
Choose the option: 1. Celsius to Farenheit, 2. Farenheit to Celsius
1
Enter temperature in Celsius:
0
Temperature in farenheit is 32

Hp@DESKTOP-0AFPT6H MINGW64 ~/Desktop/bridgelabz
$  ./func1.sh
Choose the option: 1. Celsius to Farenheit, 2. Farenheit to Celsius
2
Enter temperature in Farenheit:
212
Temperature in celsius is 99.9
```

Problem 2] Write a function to check if the two numbers are palindromes.

Solution: nano func2.sh

```
#!/bin/bash
Palindrome(){
x=0
rev=""
temp=$num
while [ $num -gt 0 ]
do
        x=$(( $num % 10 ))
        num=$(( $num / 10 ))
        rev=$( echo ${rev}${x} )
done

if [ $temp -eq $rev ]
then
        echo "Number is palindrome"
else
        echo "Number is not palindrome"
fi
}

echo "Enter first number:"
read num
Palindrome $num
echo "Enter second number:"
read num
Palindrome $num
```

Output: chmod +x func2.sh

```
Hp@DESKTOP-0AFPT6H MINGW64 ~/Desktop/bridgelabz
$  ./func2.sh
Enter first number:
1881
Number is palindrome
Enter second number:
8945
Number is not palindrome
```

Problem 3] Take a number from user and check if the number is a Prime then show its Palindrome is also prime.

a) **Write function to check number is prime**
b) **Write function to get the Palindrome**
c) **Check if Palindrome number is also prime**

Solution: nano func3.sh

```
#!/bin/bash
prime(){
for (( count=1; count<=n; count++ ))
do
        x=`expr $num % 2`
        if [ $x -eq 0 ]
        then
                echo "It is prime number"
                     break
        else
                echo "Not a prime number"
                     break
        fi
done
}

palindrome(){
x=0
rev=""
temp=$num
while [ $num -gt 0 ]
do
        x=$(( $num % 10 ))
        num=$(( $num / 10 ))
        rev=$( echo ${rev}${x} )
done

if [ $temp -eq $rev ]
then
        echo "Number is palindrome"
else
        echo "Number is not palindrome"
fi
}

echo "1] To check prime number: , 2] To check palindrome: , 3] To check prime and
palindrome:"
read n
case $n in
   "1") echo "Enter the number:"
        read num
        prime $num
    ;;
   "2") echo "Enter the number:"
        read num
        palindrome $num
    ;;
   "3") echo "Enter the number:"
        read num
        prime $num
        palindrome $num
    ;;
    *) echo "Invalid input"
    ;;
```

```
esac
```

Output :

```
Hp@DESKTOP-0AFPT6H MINGW64 ~/Desktop/bridgelabz
$  ./func3.sh
1] To check prime number: , 2] To check palindrome: , 3] To check prime and
palindrome:
1
Enter the number:
45
Not a prime number

Hp@DESKTOP-0AFPT6H MINGW64 ~/Desktop/bridgelabz
$  ./func3.sh
1] To check prime number: , 2] To check palindrome: , 3] To check prime and
palindrome:
2
Enter the number:
485
Number is not palindrome

Hp@DESKTOP-0AFPT6H MINGW64 ~/Desktop/bridgelabz
$  ./func3.sh
1] To check prime number: , 2] To check palindrome: , 3] To check prime and
palindrome:
2
Enter the number:
141
Number is palindrome

Hp@DESKTOP-0AFPT6H MINGW64 ~/Desktop/bridgelabz
$  ./func3.sh
1] To check prime number: , 2] To check palindrome: , 3] To check prime and
palindrome:
1
Enter the number:
42
It is prime number

Hp@DESKTOP-0AFPT6H MINGW64 ~/Desktop/bridgelabz
$  ./func3.sh
1] To check prime number: , 2] To check palindrome: , 3] To check prime and
palindrome:
3
Enter the number:
456
It is prime number
Number is not palindrome

Hp@DESKTOP-0AFPT6H MINGW64 ~/Desktop/bridgelabz
$  ./func3.sh
1] To check prime number: , 2] To check palindrome: , 3] To check prime and
palindrome:
1441
Invalid input

Hp@DESKTOP-0AFPT6H MINGW64 ~/Desktop/bridgelabz
$  ./func3.sh
1] To check prime number: , 2] To check palindrome: , 3] To check prime and
palindrome:
3
Enter the number:
1441
Not a prime number
Number is palindrome
```