Day 6 Practice Problems

Solved by: SAPTAK DAS(saptakds@gmail.com)

Repetition Practice Problems with for 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.

```
#!/bin/bash -x
n=$1
for num in `seq 0 $n`
do
    echo $((2**$num))
done
```

```
+ n=10
++ seq 0 10
+ for num in `seq 0 $n`
+ echo 1
1
+ for num in `seq 0 $n`
+ echo 2
2
+ for num in `seq 0 $n`
+ echo 4
4
+ for num in `seq 0 $n`
+ echo 8
8
+ for num in `seq 0 $n`
+ echo 16
16
+ for num in `seq 0 $n`
+ echo 32
32
+ for num in `seq 0 $n`
+ echo 64
64
+ for num in `seq 0 $n`
+ echo 128
128
+ for num in `seq 0 $n`
+ echo 256
256
+ for num in `seq 0 $n`
+ echo 512
512
+ for num in `seq 0 $n`
+ echo 1024
1024
```

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

```
#!/bin/bash -x
n=$1
harmonic=0
for num in `seq $n`
do
   Hn=$(echo "scale=2;1/$num" | bc)
   harmonic=$(echo "scale=2;$harmonic+$Hn" | bc)
done
echo "Harmonic no.: $harmonic"
```

```
+ n=3
+ harmonic=0
++ seq 3
+ for num in `seq $n`
++ echo 'scale=2;1/1'
++ bc
+ Hn=1.00
++ echo 'scale=2;0+1.00'
++ bc
+ harmonic=1.00
+ for num in `seq $n`
++ echo 'scale=2;1/2'
++ bc
+ Hn=.50
++ echo 'scale=2;1.00+.50'
++ bc
+ harmonic=1.50
+ for num in `seq $n`
++ echo 'scale=2;1/3'
++ bc
+ Hn=.33
++ echo 'scale=2;1.50+.33'
++ bc
+ harmonic=1.83
+ echo 'Harmonic no.: 1.83'
Harmonic no.: 1.83
```

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

```
#!/bin/bash -x
function isPrime(){
   if [ $(($1 % 2)) -eq 0 ]
   then
      return 1
   fi
for(( i=3 ; i<=$(($1/2)) ; i+=2 ))
   do
      if [ $(($1 % $i)) -eq 0 ]
      then
         return 1
      fi
   done
   return 0
}
if isPrime $1
then
   echo "Prime no."
else
   echo "Not a prime no."
fi
```

```
+ isPrime 3
+ '[' 1 -eq 0 ']'
+ (( i=3  ))
+ (( i<=1  ))
+ true
+ echo 'Prime no.'
Prime no.
```

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

```
#!/bin/bash -x
function isPrime(){
   if [ $(($1 % 2)) -eq 0 ]
   then
      return 1
   fi
   for(( i=3 ; i<=$(($1/2)) ; i+=2 ))
   do
      if [ $(($1 % $i)) -eq 0 ]
      then
         return 1
      fi
   done
   return 0
}
```

```
for num in `seq $1 $2`
do
    if isPrime $num
    then
        echo $num
    fi
done
```

```
++ seq 10 15
+ for num in `seq $1 $2`
+ isPrime 10
+ '[' 0 -eq 0 ']'
+ return 1
+ for num in `seq $1 $2`
+ isPrime 11
+ '[' 1 -eq 0 ']'
+ (( i=3  ))
+ (( i<=5  ))
+ '[' 2 -eq 0 ']'
+ (( i+=2  ))
+ (( i<=5  ))
+ '[' 1 -eq 0 ']'
+ (( i+=2  ))
+ (( i<=5  ))
+ return 0
+ echo 11
11
+ for num in `seq $1 $2`
+ isPrime 12
+ '[' 0 -eq 0 ']'
+ return 1
+ for num in `seq $1 $2`
+ isPrime 13
+ '[' 1 -eq 0 ']'
+ (( i=3  ))
+ (( i<=6  ))
+ '[' 1 -eq 0 ']'
+ (( i+=2  ))
+ (( i<=6  ))
+ '[' 3 -eq 0 ']'
+ (( i+=2  ))
+ (( i<=6  ))
+ return 0
+ echo 13
13
+ for num in `seq $1 $2`
+ isPrime 14
+ '[' 0 -eq 0 ']'
+ return 1
+ for num in `seq $1 $2`
+ isPrime 15
+ '[' 1 -eq 0 ']'
+ (( i=3  ))
+ (( i<=7  ))
+ '[' 0 -eq 0 ']'
+ return 1
```

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

```
#!/bin/bash -x
function factorial(){
   if [ $1 -le 1 ]
   then
      echo 1
   else
      echo $(( $1 * $(factorial $(( $1 - 1 ))) ))
   fi
}
factorial 4
```

```
+ factorial 4
+ '[' 4 -le 1 ']'
++ factorial 3
++ '[' 3 -le 1 ']'
+++ factorial 2
+++ '[' 2 -le 1 ']'
++++ factorial 1
++++ '[' 1 -le 1 ']'
++++ echo 1
+++ echo 2
++ echo 6
+ echo 24
24
```

6. Write a program to compute Factors of a number N using prime factorization method.

```
#!/bin/bash -x
function isPrime(){
   if [ $1 -eq 2 ]
   then
      return 0
   elif [ $(($1 % 2)) -eq 0 ]
   then
      return 1
   fi
   for(( i=3 ; i<=$(($1/2)) ; i+=2 ))
   do
      if [ $(($1 % $i)) -eq 0 ]
      then
         return 1
      fi
   done
```

```
    return 0
}
read -p "Enter N: " N
for (( j=2 ; $((j*j))<=$N ; j++ ))
do
if [ $(($N % $j)) -eq 0 ]
    then
        if isPrime $j
        then
            echo $j
        else
            continue
        fi
    fi
done
```

```
+ read -p 'Enter N: ' N
Enter N: 30
+ (( j=2  ))
+ (( 4<=30  ))
+ '[' 0 -eq 0 ']'
+ isPrime 2
+ '[' 2 -eq 2 ']'
+ return 0
+ echo 2
2
+ (( j++  ))
+ (( 9<=30  ))
+ '[' 0 -eq 0 ']'
+ isPrime 3
+ '[' 3 -eq 2 ']'
+ '[' 1 -eq 0 ']'
+ (( i=3  ))
+ (( i<=1  ))
+ return 0
+ echo 3
3
+ (( j++  ))
+ (( 16<=30  ))
+ '[' 2 -eq 0 ']'
+ (( j++  ))
+ (( 25<=30  ))
+ '[' 0 -eq 0 ']'
+ isPrime 5
+ '[' 5 -eq 2 ']'
+ '[' 1 -eq 0 ']'
+ (( i=3  ))
+ (( i<=2  ))
+ return 0
+ echo 5
5
+ (( j++  ))
+ (( 36<=30  ))
```

Repetition Practice Problems with 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.

```
#!/bin/bash -x
n=$1
num=0
while [ $num -le $n -a $((2**$num)) -le 256 ]
do
   echo $((2**$num))
   ((num++))
done
```

```
+ n=10
+ num=0
+ '[' 0 -le 10 -a 1 -le 256 ']'
+ echo 1
1
+ (( num++ ))
+ '[' 1 -le 10 -a 2 -le 256 ']'
+ echo 2
2
+ (( num++ ))
+ '[' 2 -le 10 -a 4 -le 256 ']'
+ echo 4
4
+ (( num++ ))
+ '[' 3 -le 10 -a 8 -le 256 ']'
+ echo 8
8
+ (( num++ ))
+ '[' 4 -le 10 -a 16 -le 256 ']'
+ echo 16
16
+ (( num++ ))
+ '[' 5 -le 10 -a 32 -le 256 ']'
+ echo 32
32
+ (( num++ ))
+ '[' 6 -le 10 -a 64 -le 256 ']'
+ echo 64
64
+ (( num++ ))
+ '[' 7 -le 10 -a 128 -le 256 ']'
+ echo 128
128
+ (( num++ ))
+ '[' 8 -le 10 -a 256 -le 256 ']'
+ echo 256
256
+ (( num++ ))
+ '[' 9 -le 10 -a 512 -le 256 ']'
```

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.

```
#!/bin/bash -x
echo "Think of a number between 1 to 100"
choice=2
beg=1
end=100
while [ $choice -ne 1 ]
do
   mid=$((($beg + $end) / 2))
   read -p "Is your no. $mid?  1. Yes  2. Greater than this    3. Lesser than this" choice
   if [ $choice -eq 2 ]
   then
      beg=$mid
   elif [ $choice -eq 3 ]
   then
      end=$mid
   elif [ $choice -eq 1 ]
   then
      echo "Magic no. is $mid"
   else
      echo "Invalid input"
   fi
done
```

```
+ echo 'Think of a number between 1 to 100'
Think of a number between 1 to 100
+ choice=2
+ beg=1
+ end=100
+ '[' 2 -ne 1 ']'
+ mid=50
+ read -p 'Is your no. 50?      1. Yes  2. Greater than this    3. Lesser than this' choice
Is your no. 50? 1. Yes  2. Greater than this    3. Lesser than this2
+ '[' 2 -eq 2 ']'
+ beg=50
+ '[' 2 -ne 1 ']'
+ mid=75
+ read -p 'Is your no. 75?      1. Yes  2. Greater than this    3. Lesser than this' choice
Is your no. 75? 1. Yes  2. Greater than this    3. Lesser than this2
+ '[' 2 -eq 2 ']'
+ beg=75
+ '[' 2 -ne 1 ']'
+ mid=87
+ read -p 'Is your no. 87?      1. Yes  2. Greater than this    3. Lesser than this' choice
Is your no. 87? 1. Yes  2. Greater than this    3. Lesser than this3
+ '[' 3 -eq 2 ']'
+ '[' 3 -eq 3 ']'
+ end=87
+ '[' 3 -ne 1 ']'
+ mid=81
+ read -p 'Is your no. 81?      1. Yes  2. Greater than this    3. Lesser than this' choice
Is your no. 81? 1. Yes  2. Greater than this    3. Lesser than this2
+ '[' 2 -eq 2 ']'
+ beg=81
+ '[' 2 -ne 1 ']'
+ mid=84
+ read -p 'Is your no. 84?      1. Yes  2. Greater than this    3. Lesser than this' choice
Is your no. 84? 1. Yes  2. Greater than this    3. Lesser than this3
+ '[' 3 -eq 2 ']'
+ '[' 3 -eq 3 ']'
+ end=84
+ '[' 3 -ne 1 ']'
+ mid=82
+ read -p 'Is your no. 82?      1. Yes  2. Greater than this    3. Lesser than this' choice
Is your no. 82? 1. Yes  2. Greater than this    3. Lesser than this1
+ '[' 1 -eq 2 ']'
+ '[' 1 -eq 3 ']'
+ '[' 1 -eq 1 ']'
+ echo 'Magic no. is 82'
Magic no. is 82
+ '[' 1 -ne 1 ']'
```

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

```
#!/bin/bash -x
heads=0
tails=0
while [ $heads -ne 11 -a $tails -ne 11 ]
do
   echo "Flipping coin..."
   toss=$((1+RANDOM%2))
   if [ $toss -eq 1 ]
   then
      ((heads++))
   else
      ((tails++))
   fi
done
if [ $heads -eq 11 ]
then
   echo "Head wins"
else
```

```
    echo "Tail wins"
fi
```

```
+ heads=0
+ tails=0
+ '[' 0 -ne 11 -a 0 -ne 11 ']'
+ echo 'Flipping coin...'
Flipping coin...
+ toss=2
+ '[' 2 -eq 1 ']'
+ (( tails++ ))
+ '[' 0 -ne 11 -a 1 -ne 11 ']'
+ echo 'Flipping coin...'
Flipping coin...
+ toss=2
+ '[' 2 -eq 1 ']'
+ (( tails++ ))
+ '[' 0 -ne 11 -a 2 -ne 11 ']'
+ echo 'Flipping coin...'
Flipping coin...
+ toss=2
+ '[' 2 -eq 1 ']'
+ (( tails++ ))
+ '[' 0 -ne 11 -a 3 -ne 11 ']'
+ echo 'Flipping coin...'
Flipping coin...
+ toss=1
+ '[' 1 -eq 1 ']'
+ (( heads++ ))
+ '[' 1 -ne 11 -a 3 -ne 11 ']'
+ echo 'Flipping coin...'
Flipping coin...
+ toss=2
+ '[' 2 -eq 1 ']'
+ (( tails++ ))
+ '[' 1 -ne 11 -a 4 -ne 11 ']'
+ echo 'Flipping coin...'
Flipping coin...
+ toss=1
+ '[' 1 -eq 1 ']'
+ (( heads++ ))
+ '[' 2 -ne 11 -a 4 -ne 11 ']'
+ echo 'Flipping coin...'
Flipping coin...
+ toss=2
+ '[' 2 -eq 1 ']'
+ (( tails++ ))
+ '[' 2 -ne 11 -a 5 -ne 11 ']'
+ echo 'Flipping coin...'
Flipping coin...
+ toss=1
+ '[' 1 -eq 1 ']'
+ (( heads++ ))
+ '[' 3 -ne 11 -a 5 -ne 11 ']'
+ echo 'Flipping coin...'
Flipping coin...
+ toss=1
+ '[' 1 -eq 1 ']'
+ (( heads++ ))
+ '[' 4 -ne 11 -a 5 -ne 11 ']'
+ echo 'Flipping coin...'
Flipping coin...
+ toss=1
+ '[' 1 -eq 1 ']'
+ (( heads++ ))
+ '[' 5 -ne 11 -a 5 -ne 11 ']'
+ echo 'Flipping coin...'
Flipping coin...
+ toss=2
+ '[' 2 -eq 1 ']'
+ (( tails++ ))
+ '[' 5 -ne 11 -a 6 -ne 11 ']'
+ echo 'Flipping coin...'
Flipping coin...
+ toss=2
+ '[' 2 -eq 1 ']'
+ (( tails++ ))
+ '[' 5 -ne 11 -a 7 -ne 11 ']'
+ echo 'Flipping coin...'
Flipping coin...
+ toss=2
+ '[' 2 -eq 1 ']'
+ (( tails++ ))
+ '[' 5 -ne 11 -a 8 -ne 11 ']'
+ echo 'Flipping coin...'
Flipping coin...
+ toss=2
+ '[' 2 -eq 1 ']'
+ (( tails++ ))
+ '[' 5 -ne 11 -a 9 -ne 11 ']'
+ echo 'Flipping coin...'
Flipping coin...
+ toss=1
+ '[' 1 -eq 1 ']'
+ (( heads++ ))
+ '[' 6 -ne 11 -a 9 -ne 11 ']'
+ echo 'Flipping coin...'
Flipping coin...
+ toss=1
+ '[' 1 -eq 1 ']'
+ (( heads++ ))
+ '[' 7 -ne 11 -a 9 -ne 11 ']'
+ echo 'Flipping coin...'
Flipping coin...
+ toss=1
+ '[' 1 -eq 1 ']'
+ (( heads++ ))
+ '[' 8 -ne 11 -a 9 -ne 11 ']'
+ echo 'Flipping coin...'
```

```
Flipping coin...
+ toss=1
+ '[' 1 -eq 1 ']'
+ (( heads++ ))
+ '[' 9 -ne 11 -a 9 -ne 11 ']'
+ echo 'Flipping coin...'
Flipping coin...
+ toss=1
+ '[' 1 -eq 1 ']'
+ (( heads++ ))
+ '[' 10 -ne 11 -a 9 -ne 11 ']'
+ echo 'Flipping coin...'
Flipping coin...
+ toss=2
+ '[' 2 -eq 1 ']'
+ (( tails++ ))
+ '[' 10 -ne 11 -a 10 -ne 11 ']'
+ echo 'Flipping coin...'
Flipping coin...
+ toss=2
+ '[' 2 -eq 1 ']'
+ (( tails++ ))
+ '[' 10 -ne 11 -a 11 -ne 11 ']'
+ '[' 10 -eq 11 ']'
+ echo 'Tail wins'
Tail wins
```

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.

```
#!/bin/bash -x
amt=100
won=0
bets=0
while [ $amt -gt 0 -a $amt -lt 200  ]
do
   ((bets++))
   result=$((1+RANDOM%2))
   if [ $result -eq 1 ]
   then
      ((won++))
      ((amt++))
   else
      ((amt--))
   fi
   echo "Won: $won"
   echo "Bets: $bets"
done
if [ $amt -eq 200 ]
then
```

```
    echo "Won"
else
    echo "Broke"
fi
```

```
+ result=2
+ '[' 2 -eq 1 ']'
+ (( amt-- ))
+ echo 'Won: 1482'
Won: 1482
+ echo 'Bets: 3062'
Bets: 3062
+ '[' 2 -gt 0 -a 2 -lt 200 ']'
+ (( bets++ ))
+ result=2
+ '[' 2 -eq 1 ']'
+ (( amt-- ))
+ echo 'Won: 1482'
Won: 1482
+ echo 'Bets: 3063'
Bets: 3063
+ '[' 1 -gt 0 -a 1 -lt 200 ']'
+ (( bets++ ))
+ result=2
+ '[' 2 -eq 1 ']'
+ (( amt-- ))
+ echo 'Won: 1482'
Won: 1482
+ echo 'Bets: 3064'
Bets: 3064
+ '[' 0 -gt 0 -a 0 -lt 200 ']'
+ '[' 0 -eq 200 ']'
+ echo Broke
Broke
```

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

```
#!/bin/bash -x
function degCConv(){
   if [ $1 -ge 0 -a $1 -le 100 ]
   then
      FTemp=$(echo "scale=2;1.8*$1" | bc)
      finalFTemp=$(echo "scale=2;$FTemp+32" | bc)
      echo $finalFTemp
   else
      echo "Invalid temperature"
   fi
}
function degFConv(){
   if [ $1 -ge 32 -a $1 -le 212 ]
   then
      CTemp=$(echo "scale=2;$1-32" | bc)
      finalCTemp=$(echo "scale=2;$CTemp*0.55" | bc)
      echo $finalCTemp
   else
      echo "Invalid temperature"
   fi
}
echo "1. degree C"
echo "2. degree F"
read -p "Enter your choice: " choice
case $choice in
   1)
      read -p "Enter temperature: " temp
      degFConv $temp
      ;;
   2)
      read -p "Enter temperature: " temp
      degCConv $temp
      ;;
   *)
      echo "Invalid Input!"
      ;;
esac
```

```
+ echo '1. degree C'
1. degree C
+ echo '2. degree F'
2. degree F
+ read -p 'Enter your choice: ' choice
Enter your choice: 1
+ case $choice in
+ read -p 'Enter temperature: ' temp
Enter temperature: 212
+ degFConv 212
+ '[' 212 -ge 32 -a 212 -le 212 ']'
++ echo 'scale=2;212-32'
++ bc
+ CTemp=180
++ echo 'scale=2;180*0.55'
++ bc
+ finalCTemp=99.00
+ echo 99.00
99.00
```

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

```
#!/bin/bash -x
function isPalindrome(){
    num=$1
    sum=0
    while [ $num -ne 0 ]
    do
        r=`expr $num % 10`
        sum=`expr $(($sum * 10)) + $r`
        num=`expr $num / 10`
    done
    if [ $1 -eq $sum ]
    then
        echo "Palindrome"
    else
        echo "Not palindrome"
    fi
}
read -p "Enter first no.: " a
read -p "Enter second no.: " b
isPalindrome $a
isPalindrome $b
```

```
+ read -p 'Enter first no.: ' a
Enter first no.: 121
+ read -p 'Enter second no.: ' b
Enter second no.: 123
+ isPalindrome 121
+ num=121
+ sum=0
+ '[' 121 -ne 0 ']'
++ expr 121 % 10
+ r=1
++ expr 0 + 1
+ sum=1
++ expr 121 / 10
+ num=12
+ '[' 12 -ne 0 ']'
++ expr 12 % 10
+ r=2
++ expr 10 + 2
+ sum=12
++ expr 12 / 10
+ num=1
+ '[' 1 -ne 0 ']'
++ expr 1 % 10
+ r=1
++ expr 120 + 1
+ sum=121
++ expr 1 / 10
+ num=0
+ '[' 0 -ne 0 ']'
+ '[' 121 -eq 121 ']'
+ echo Palindrome
Palindrome
+ isPalindrome 123
+ num=123
+ sum=0
+ '[' 123 -ne 0 ']'
++ expr 123 % 10
+ r=3
++ expr 0 + 3
+ sum=3
++ expr 123 / 10
+ num=12
+ '[' 12 -ne 0 ']'
++ expr 12 % 10
+ r=2
++ expr 30 + 2
+ sum=32
++ expr 12 / 10
+ num=1
+ '[' 1 -ne 0 ']'
++ expr 1 % 10
+ r=1
++ expr 320 + 1
+ sum=321
++ expr 1 / 10
+ num=0
+ '[' 0 -ne 0 ']'
+ '[' 123 -eq 321 ']'
+ echo 'Not palindrome'
Not palindrome
```

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

```
#!/bin/bash -x
function isPrime(){
   if [ $1 -eq 2 ]
   then
      return 0
   elif [ $(($1 % 2)) -eq 0 ]
   then
      return 1
   fi
   for(( i=3 ; i<=$(($1/2)) ; i+=2 ))
   do
      if [ $(($1 % $i)) -eq 0 ]
      then
         return 1
      fi
   done
   return 0
}
function getPalindrome(){
   num=$1
   sum=0
   while [ $num -ne 0 ]
   do
      r=$(($num % 10))
      sum=`expr $(($sum * 10)) + $r`
      num=`expr $num / 10`
   done
   return $sum
}
read -p "Enter a no.: " N
if isPrime $N
then
   echo "No. is prime"
   getPalindrome $N
   palin=$?
   if isPrime $palin
   then
      echo "Palindrome is also prime"
   else
      echo "Palindrome is not prime"
   fi
else
   echo "No. is not prime"
fi
```

```
+ read -p 'Enter a no.: ' N
Enter a no.: 13
+ isPrime 13
+ '[' 13 -eq 2 ']'
+ '[' 1 -eq 0 ']'
+ (( i=3  ))
+ (( i<=6  ))
+ '[' 1 -eq 0 ']'
+ (( i+=2  ))
+ (( i<=6  ))
+ '[' 3 -eq 0 ']'
+ (( i+=2  ))
+ (( i<=6  ))
+ return 0
+ echo 'No. is prime'
No. is prime
+ getPalindrome 13
+ num=13
+ sum=0
+ '[' 13 -ne 0 ']'
+ r=3
++ expr 0 + 3
+ sum=3
++ expr 13 / 10
+ num=1
+ '[' 1 -ne 0 ']'
+ r=1
++ expr 30 + 1
+ sum=31
++ expr 1 / 10
+ num=0
+ '[' 0 -ne 0 ']'
+ return 31
+ palin=31
+ isPrime 31
+ '[' 31 -eq 2 ']'
+ '[' 1 -eq 0 ']'
+ (( i=3  ))
+ (( i<=15  ))
+ '[' 1 -eq 0 ']'
+ (( i+=2  ))
+ (( i<=15  ))
+ '[' 1 -eq 0 ']'
+ (( i+=2  ))
+ (( i<=15  ))
+ '[' 3 -eq 0 ']'
+ (( i+=2  ))
+ (( i<=15  ))
+ '[' 4 -eq 0 ']'
+ (( i+=2  ))
+ (( i<=15  ))
+ '[' 9 -eq 0 ']'
+ (( i+=2  ))
+ (( i<=15  ))
+ '[' 5 -eq 0 ']'
+ (( i+=2  ))
+ (( i<=15  ))
+ '[' 1 -eq 0 ']'
+ (( i+=2  ))
+ (( i<=15  ))
+ return 0
+ echo 'Palindrome is also prime'
Palindrome is also prime
```