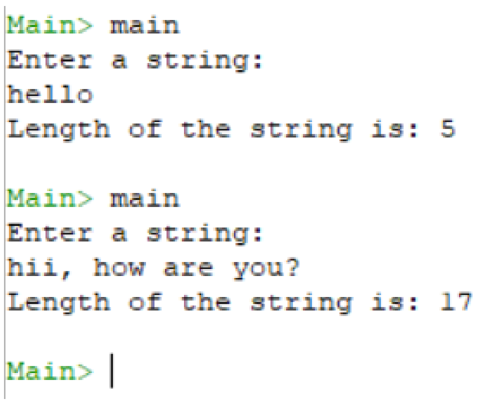
19CSE313 - Principles of Programming Languages

Labsheet – 9

SHOAIB AKHTAR
AM.EN.U4CSE20163

1) Write an IO program **to read a string** and **display its length**.

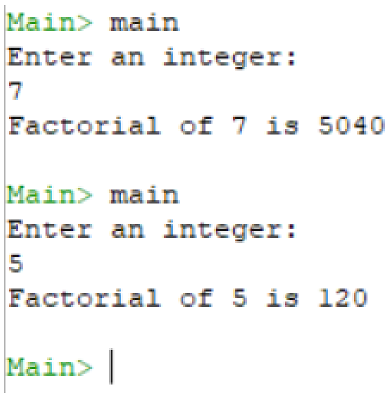
```
main = do
   putStrLn "Enter a string:"
   str <- getLine
   putStrLn ("Length of the string is: " ++ show (length str))
```



```
Main> main
Enter a string:
hello
Length of the string is: 5

Main> main
Enter a string:
hii, how are you?
Length of the string is: 17

Main> |
```

2) Write a program to **read an integer** from the user and print the **factorial** of the number.

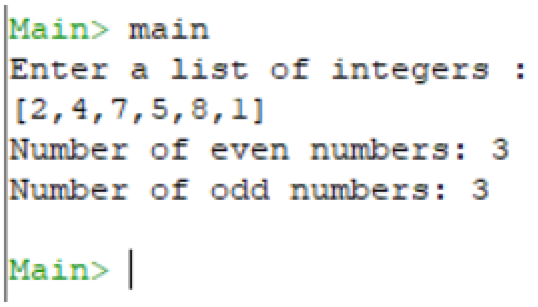
```
factorial :: Integer -> Integer
factorial n = product [1..n]


main = do
   putStrLn "Enter an integer:"
   input <- getLine
   let num = read input :: Integer
   putStrLn ("Factorial of " ++ show num ++ " is " ++ show (factorial num))
```



```
Main> main
Enter an integer:
7
Factorial of 7 is 5040

Main> main
Enter an integer:
5
Factorial of 5 is 120

Main> |
```

3) Write a program to **create a list of integers** entered by the user and display the **count of even and odd numbers**.

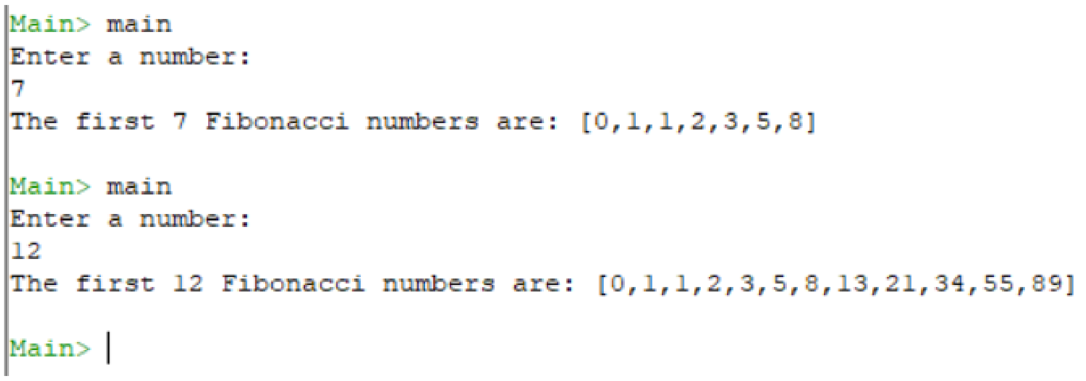
```
countEvenOdd :: [Int] -> (Int, Int)

countEvenOdd nums = (length evenNums, length oddNums)

   where evenNums = filter even nums

         oddNums = filter odd nums


main = do

   putStrLn "Enter a list of integers :"

   input <- getLine

   let nums = read input :: [Int]

   let (evenCount, oddCount) = countEvenOdd nums

   putStrLn ("Number of even numbers: " ++ show evenCount)

   putStrLn ("Number of odd numbers: " ++ show oddCount
```



```
Main> main
Enter a list of integers :
[2,4,7,5,8,1]
Number of even numbers: 3
Number of odd numbers: 3

Main> |
```

4) Write an IO program, to read a number ***n*** from the user and print the ***n Fibonacci numbers.***

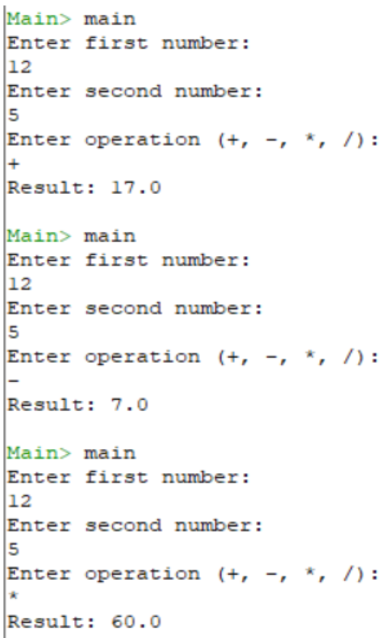
```
fibonacci :: Int -> [Int]

fibonacci 0 = []

fibonacci 1 = [0]

fibonacci 2 = [0, 1]

fibonacci n = take n (fibonaccinum 0 1)

   where fibonaccinum a b = a : fibonaccinum b (a+b)


main = do

   putStrLn "Enter a number:"

   input <- getLine

   let n = read input :: Int

   let fibNums = fibonacci n

   putStrLn ("The first " ++ show n ++ " Fibonacci numbers are: " ++ show fibNums)
```



```
Main> main
Enter a number:
7
The first 7 Fibonacci numbers are: [0,1,1,2,3,5,8]

Main> main
Enter a number:
12
The first 12 Fibonacci numbers are: [0,1,1,2,3,5,8,13,21,34,55,89]

Main> |
```

5) Write an IO program, to create a **simple calculator** with the operations +, -, /, *. Read two numbers and the operation, compute the operation and print the result.

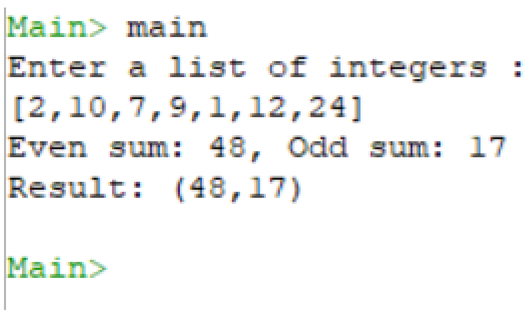
```
main = do
   putStrLn "Enter first number:"
   input1 <- getLine
   let num1 = read input1 :: Float

   putStrLn "Enter second number:"
   input2 <- getLine
   let num2 = read input2 :: Float

   putStrLn "Enter operation (+, -, *, /):"
   op <- getLine

   let result = case op of
               "+" -> num1 + num2
               "-" -> num1 - num2
               "*" -> num1 * num2
               "/" -> num1 / num2
               _   -> error "Invalid operation"
   putStrLn ("Result: " ++ show result)
```



```
Main> main
Enter first number:
12
Enter second number:
5
Enter operation (+, -, *, /):
+
Result: 17.0

Main> main
Enter first number:
12
Enter second number:
5
Enter operation (+, -, *, /):
-
Result: 7.0

Main> main
Enter first number:
12
Enter second number:
5
Enter operation (+, -, *, /):
*
Result: 60.0
```

6) Write an IO program that **reads the list of integers** from the user and prints a **tuple pair** with an **even sum** and the **odd sum** of the elements from the list.

```
import Data.List
main = do
   putStrLn "Enter a list of integers :"
   input <- getLine
   let nums = read input :: [Int]
   let evenSum = sum $ filter even nums
   let oddSum = sum $ filter odd nums
   putStrLn ("Even sum: " ++ show evenSum ++ ", Odd sum: " ++ show oddSum)
   putStrLn ("Result: " ++ show (evenSum, oddSum))
```

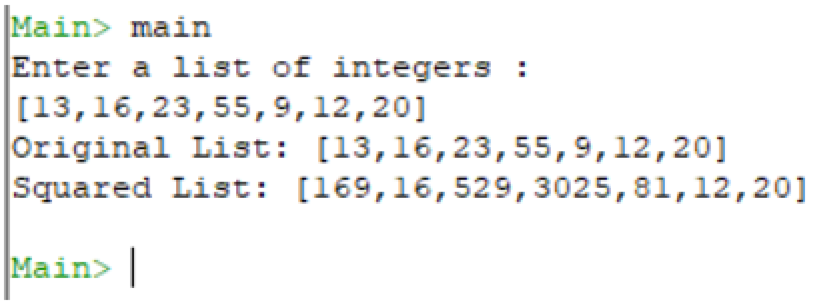


```
Main> main
Enter a list of integers :
[2,10,7,9,1,12,24]
Even sum: 48, Odd sum: 17
Result: (48,17)

Main>
```

7) Write an IO program that reads a list of integers from the user which prints a list of integers, except that each odd element of the list is replaced by the square of that element.

```
import Data.List
```

```
main = do
   putStrLn "Enter a list of integers :"
   input <- getLine
   let nums = read input :: [Int]
   let squared = map (\x -> if odd x then x^2 else x) nums
   putStrLn ("Original List: " ++ show nums)
   putStrLn ("Squared List: " ++ show squared)
```



```
Main> main
Enter a list of integers :
[13,16,23,55,9,12,20]
Original List: [13,16,23,55,9,12,20]
Squared List: [169,16,529,3025,81,12,20]

Main> |
```