Functional Programming Concept with Haskell

Tutorial for

Programming Language Laboratory (CS 431)

September – November 2020

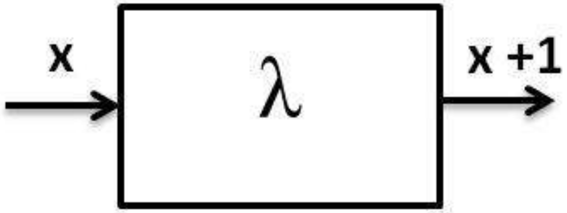
Indian Institute of Technology Guwahati

Functional Programming(FP)

- Key Idea - computation as ‘evaluation of mathematical functions’
  - Idea originated from Lambda Calculus formalism

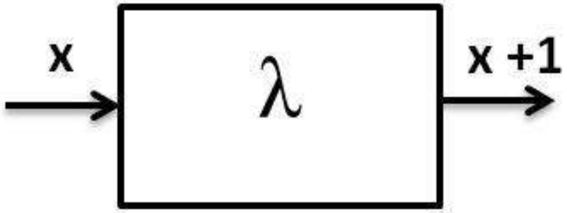
Functional Programming(FP)

- Key Idea - computation as ‘evaluation of mathematical functions’
  - Idea originated from Lambda Calculus formalism



Functional Programming(FP)

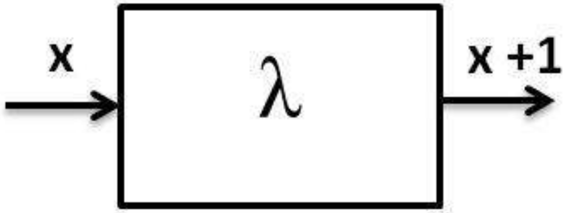
- Key Idea - computation as ‘evaluation of mathematical functions’
  - Idea originated from Lambda Calculus formalism



$\lambda x.\ x+1$

Functional Programming(FP)

- Key Idea - computation as ‘evaluation of mathematical functions’
  - Idea originated from Lambda Calculus formalism

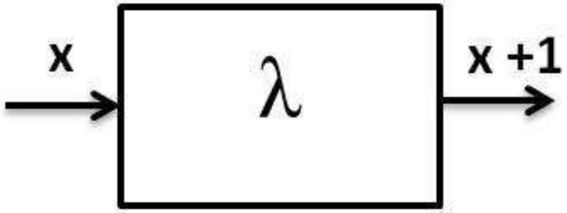


$\lambda x.\ x+1$

True:

Functional Programming(FP)

- Key Idea - computation as ‘evaluation of mathematical functions’
  - Idea originated from Lambda Calculus formalism

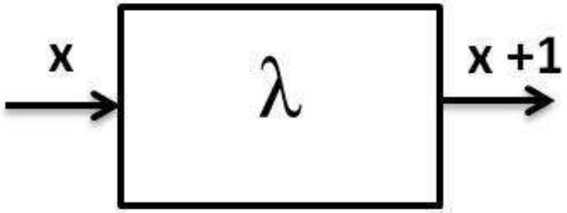


$\lambda x.\ x+1$

True: $\lambda x.\lambda y.x$

Functional Programming(FP)

- Key Idea - computation as ‘evaluation of mathematical functions’
  - Idea originated from Lambda Calculus formalism



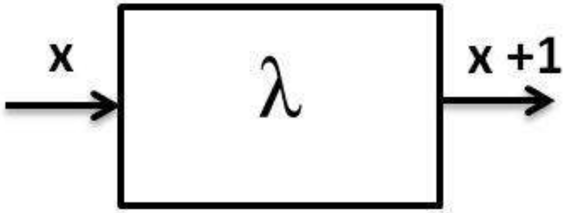
$\lambda x.\ x+1$

True: $\lambda x.\lambda y.x$

False:

Functional Programming(FP)

- Key Idea - computation as 'evaluation of mathematical functions'
  - Idea originated from Lambda Calculus formalism



$\lambda x.\ x+1$

True: $\lambda x. \lambda y. x$

False: $\lambda x. \lambda y. y$

Functional Programming(FP)

- Key Idea - computation as ‘evaluation of mathematical functions’
  - Idea originated from Lambda Calculus formalism
- Languages that follow functional programming paradigm
  - Haskell
  - LISP
  - Python
  - Erlang
  - Racket
  - F#
  - Clojure
  - Scala

Functional Programming

- Key Idea - computation as ‘evaluation of mathematical functions’
  - Idea originated from Lambda Calculus formalism
- Languages that follow functional programming paradigm
  - Haskell
  - LISP
  - Python
  - Erlang
  - Racket
  - F#
  - Clojure
  - Scala

we are going with Haskell this time

Haskell

- Standardized purely functional programming language
- Named after logician and mathematician Haskell Brooks Curry
- History
  - First version (“Haskell 1.0”) was introduced in 1990
  - The latest standard of Haskell is “Haskell 2010”

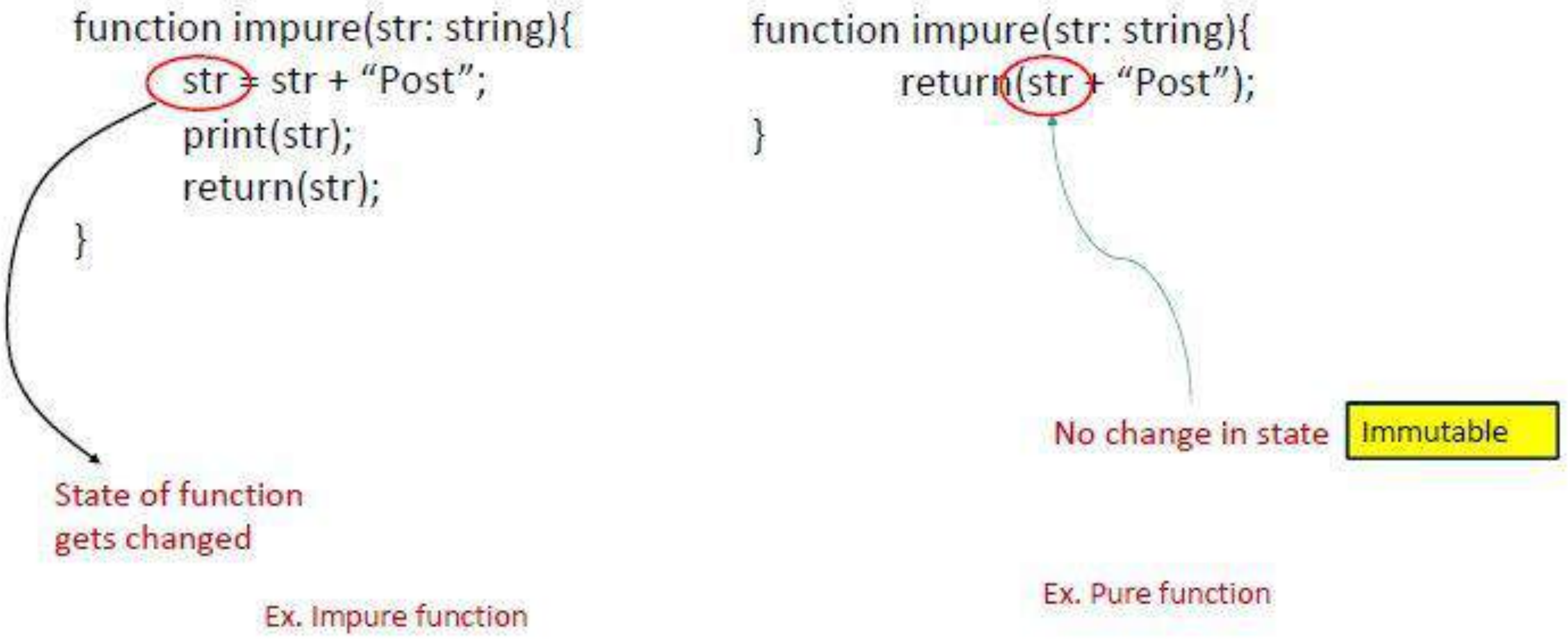
Haskell - Features

- Purely functional
- Statically typed
- Type inference
- Lazy
- Concurrent
- Packages

Purely functional

- Every function in Haskell is a function in the mathematical sense (i.e., "pure")
  - The pure function returns the same output every time for the same input
  - In a pure functional language, you can't do anything that has a side effect

Purely functional



```
function impure(str: string){
        str = str + "Post";
        print(str);
        return(str);
}
```

State of function gets changed

Ex. Impure function

```
function impure(str: string){
            return(str + "Post");
}
```

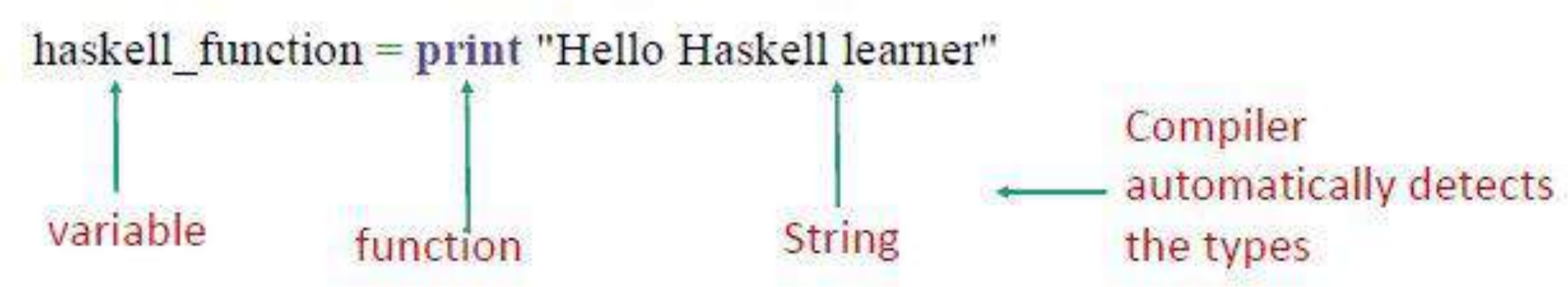
No change in state

Immutable

Ex. Pure function

Statically Typed

- Every expression in Haskell has a type which compile time is determined at compile time
  - The compiler knows which piece of code is a number, which is a string and so on

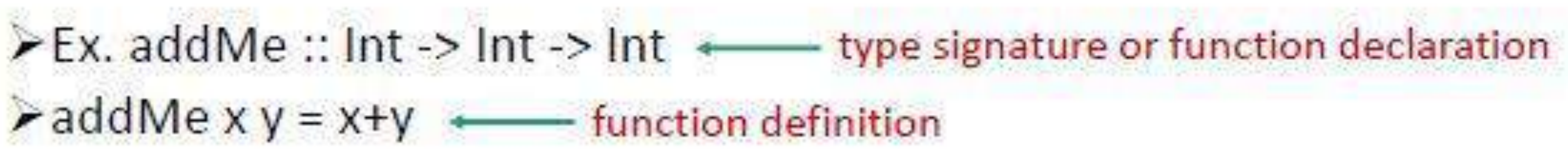


```
haskell_function = print "Hello Haskell learner"
```

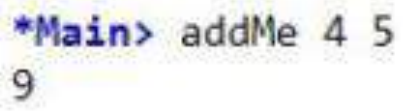
Compiler automatically detects the types

Statically Typed

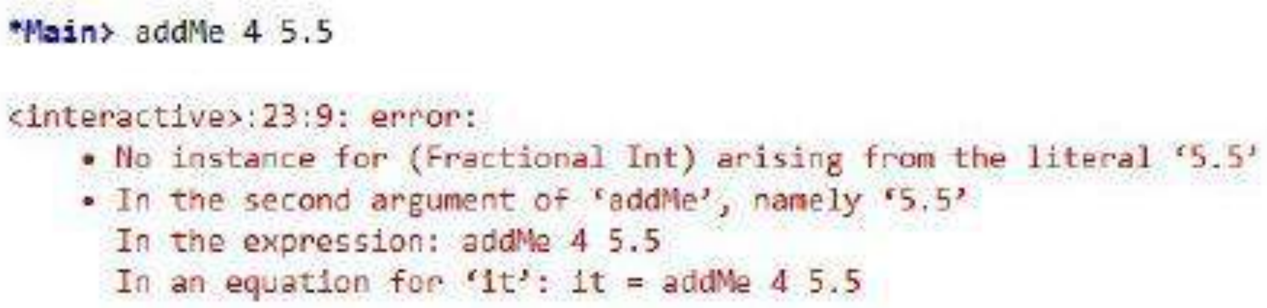
- ➢All the types composed together by function application have to match up. If they don't, the program will be rejected by the compiler



- ➢Ex. addMe :: Int -> Int -> Int ⟵ type signature or function declaration
- ➢addMe x y = x+y ⟵ function definition



```
*Main> addMe 4 5
9
```



```
*Main> addMe 4 5.5

<interactive>:23:9: error:
    • No instance for (Fractional Int) arising from the literal '5.5'
    • In the second argument of 'addMe', namely '5.5'
      In the expression: addMe 4 5.5
      In an equation for 'it': it = addMe 4 5.5
```

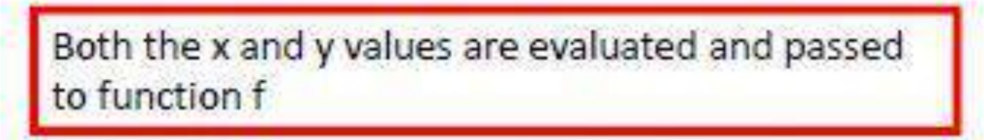
Type Inference

- You don't have to explicitly label every piece of code because the type system can intelligently figure it out
  - Eg. If we write a=5+4
    - Haskell will automatically infer that a is a number

Lazy

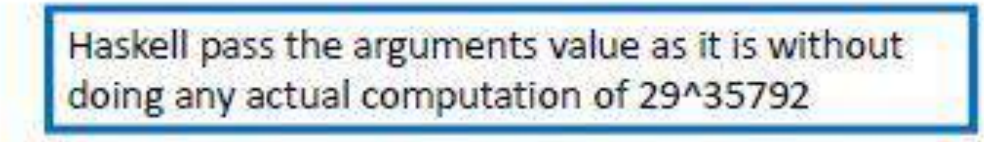
- ➢Nothing is evaluated unless it has to be

  Eg. Function call : f 5 (29^35792)

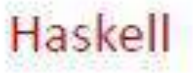


Both the x and y values are evaluated and passed to function f

Non lazy languages like C or Java



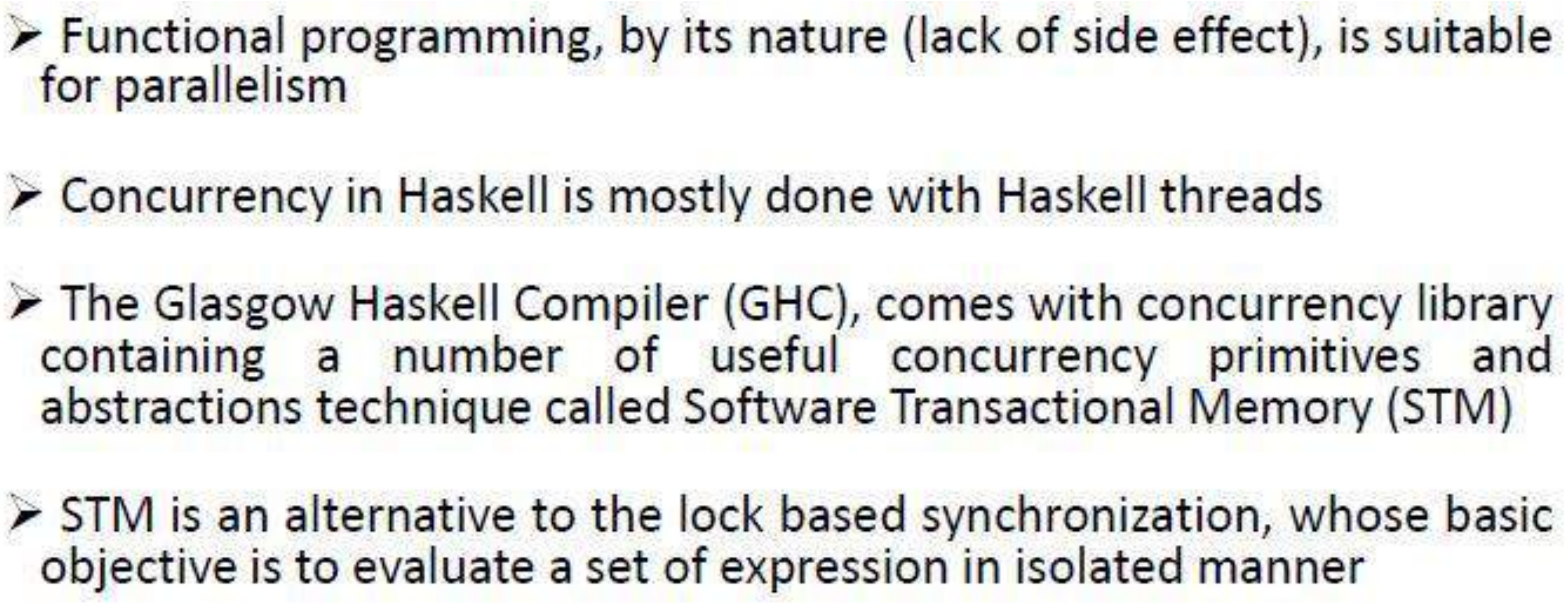
Haskell pass the arguments value as it is without doing any actual computation of 29^35792



Haskell

Saves on CPU usage and user's time!

Concurrency



- Functional programming, by its nature (lack of side effect), is suitable for parallelism

- Concurrency in Haskell is mostly done with Haskell threads

- The Glasgow Haskell Compiler (GHC), comes with concurrency library containing a number of useful concurrency primitives and abstractions technique called Software Transactional Memory (STM)

- STM is an alternative to the lock based synchronization, whose basic objective is to evaluate a set of expression in isolated manner

Haskell - Packages

- Open source contribution to Haskell is very active with a wide range of packages available on the public package servers
- There are 6,954 packages freely available; for instances

bytestring	Binary data	base	Prelude, IO, threads
network	Networking	text	Unicode text
parsec	Parser library	directory	File/directory
hspec	RSpec-like tests	attoparsec	Fast parser
monad-logger	Logging	persistent	Database ORM
template-haskell	Meta-programming	tar	Tar archives

Haskell - Application

Haskell - Application

Haskell - Application

- facebook anti-spam programs

Haskell - Application

- facebook anti-spam programs

Haskell - Application

- facebook anti-spam programs
- xmonad a window manager for the X Window System

Haskell - Application

- facebook �anti-spam programs
- X >>= xmonad �a window manager for the X Window System

darcs

Haskell - Application

- facebook ⍰anti-spam programs
- xmonad ⍰a window manager for the X Window System
- darcs ⍰ revision control system

Some other FP Applications

Some other FP Applications

twitter

Some other FP Applications

- twitter →Scala

Some other FP Applications

- twitter →Scala

  WhatsApp

Some other FP Applications

- twitter →Scala
- WhatsApp ⍰Erlang

Some other FP Applications

-
→Scala
-
Erlang

Some other FP Applications

- twitter →Scala
- WhatsApp ⍰Erlang
- ⍰ Lisp

Haskell

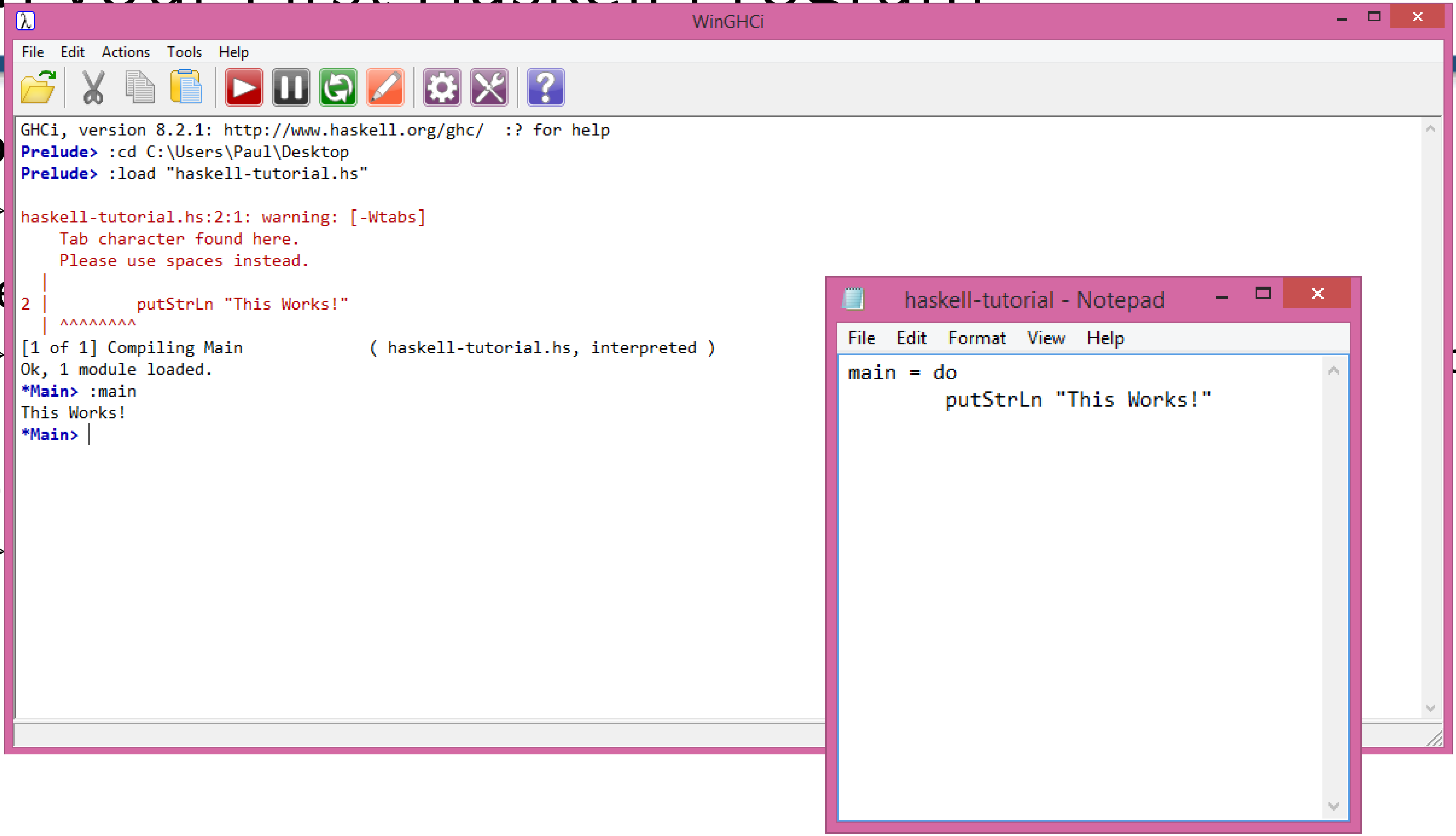
Lets try to understand basic features of Haskell with examples

Run your First Haskell Program

- Download and Install Haskell
  - Download link https://www.haskell.org/downloads
- File extension .hs
  - Open text editor, write your program, save your program with .hs extension (e.g., haskell-tutorail.hs)
- Compilation and Run
  - For Windows OS
    - Open WinGHCi from start menu
    - Load your program (File -> Load..)
    - Run the function you want

Run your First Haskell Program

- Do
  -
- File
  - …on
- Co
  -


WinGHCi
File Edit Actions Tools Help
haskell-tutorial - Notepad
File Edit Format View Help

```
GHCi, version 8.2.1: http://www.haskell.org/ghc/  :? for help
Prelude> :cd C:\Users\Paul\Desktop
Prelude> :load "haskell-tutorial.hs"

haskell-tutorial.hs:2:1: warning: [-Wtabs]
    Tab character found here.
    Please use spaces instead.
  |
2 |         putStrLn "This Works!"
  | ^^^^^^^^
[1 of 1] Compiling Main             ( haskell-tutorial.hs, interpreted )
Ok, 1 module loaded.
*Main> :main
This Works!
*Main>
```

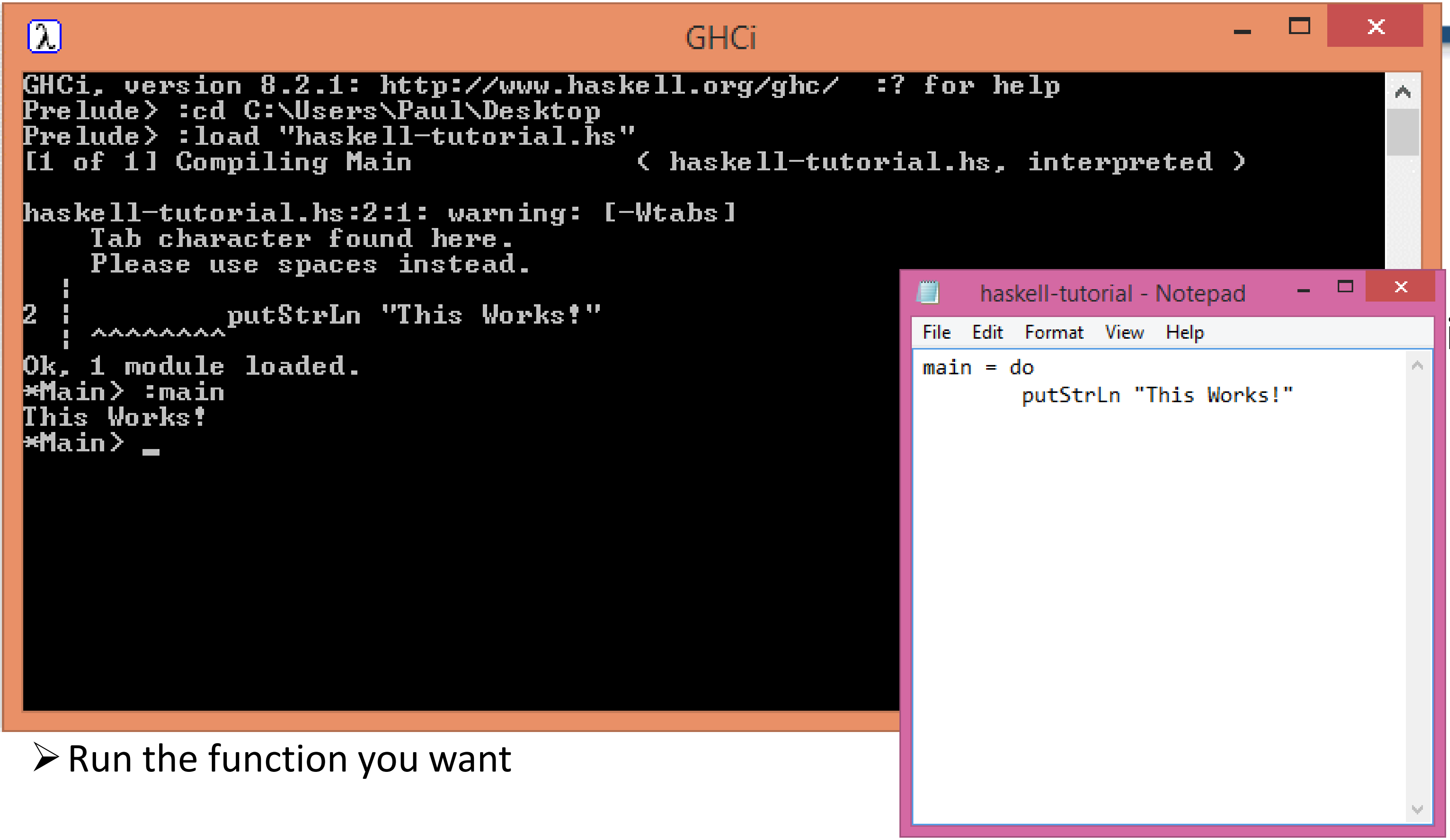
```haskell
main = do
        putStrLn "This Works!"
```

Run your First Haskell Program

- Download and Install Haskell
  - Download link https://www.haskell.org/downloads
- File extension .hs
  - Open text editor, write your program, save your program with .hs extension (e.g., haskell-tutorail.hs)
- Compilation and Run
  - Otherwise
    - Open GHCi
    - Enter into directory where you saved your program (:cd C:\Users\Paul\Desktop)
    - Load your program (:load "haskell-tutorial.hs")
    - Run the function you want

Run your First Haskell Program

- Do
  -
- File
  - ion
    (
- Cor
  - p)



```
GHCi, version 8.2.1: http://www.haskell.org/ghc/  :? for help
Prelude> :cd C:\Users\Paul\Desktop
Prelude> :load "haskell-tutorial.hs"
[1 of 1] Compiling Main             ( haskell-tutorial.hs, interpreted )

haskell-tutorial.hs:2:1: warning: [-Wtabs]
    Tab character found here.
    Please use spaces instead.
  |
2 |         putStrLn "This Works!"
  | ^^^^^^^^
Ok, 1 module loaded.
*Main> :main
This Works!
*Main> _
```

```
main = do
        putStrLn "This Works!"
```

- Run the function you want

Few things you may keep in mind

- Once you modify your program
  - Save it
  - Before running its function, recompile it - reload (*main> :r)
- Comment Line
  - --Comment
  - {-

    Multiple Comments

    -}
- Clear Screen
  - Ctrl+S

Date Types

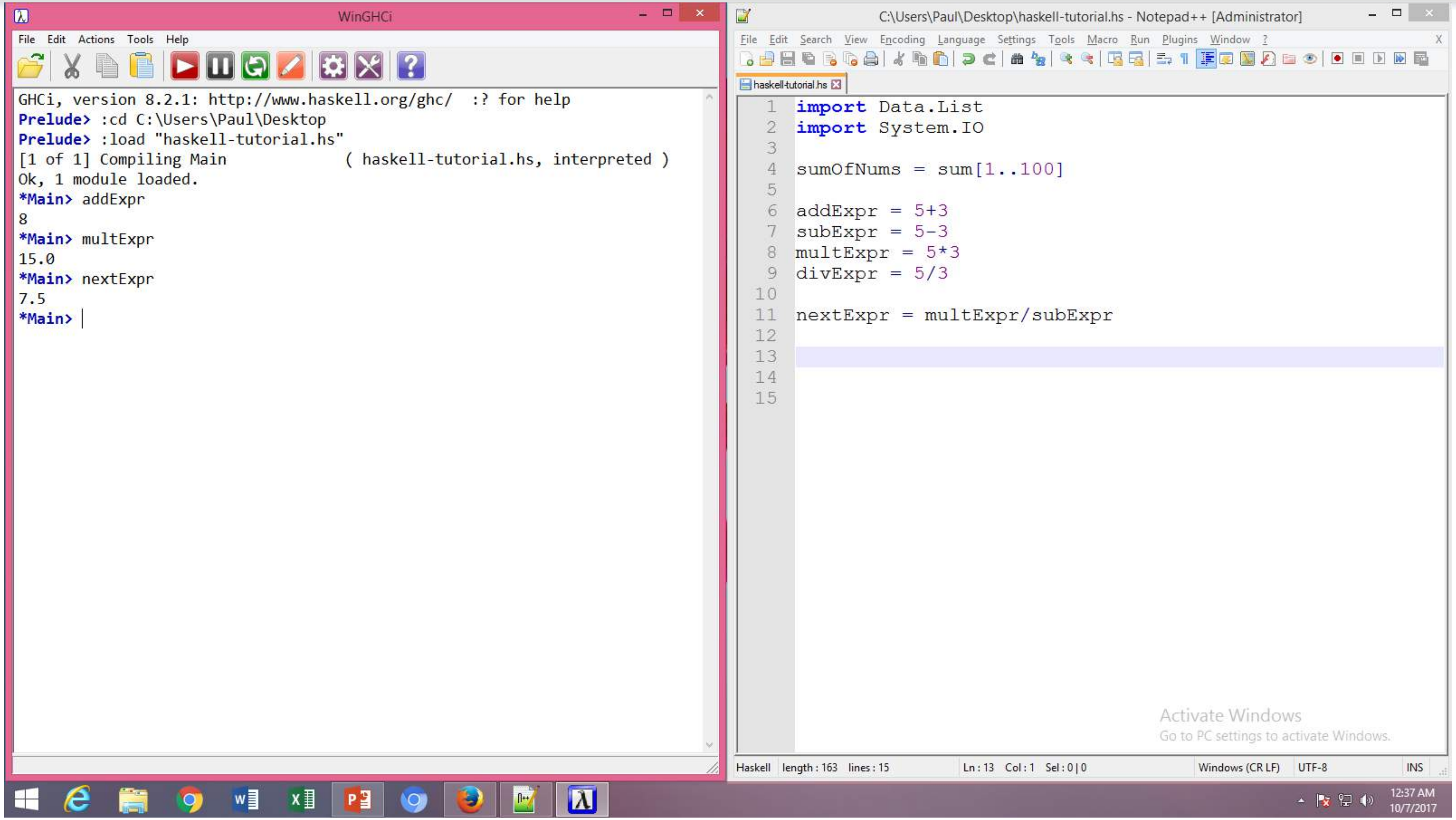
- Haskell uses type inference
  - Range of ‘Int’: -2^63 to 2^63
  - Range of ‘Integer’: Unbound -- as per the capability of memory of the system
  - Other data types: Float,Double, Bool, Char, Tuple -- will be discussing with example
  - permanent3 :: Int

    permanent3 = 3

permanent value for a variable

--Never Change

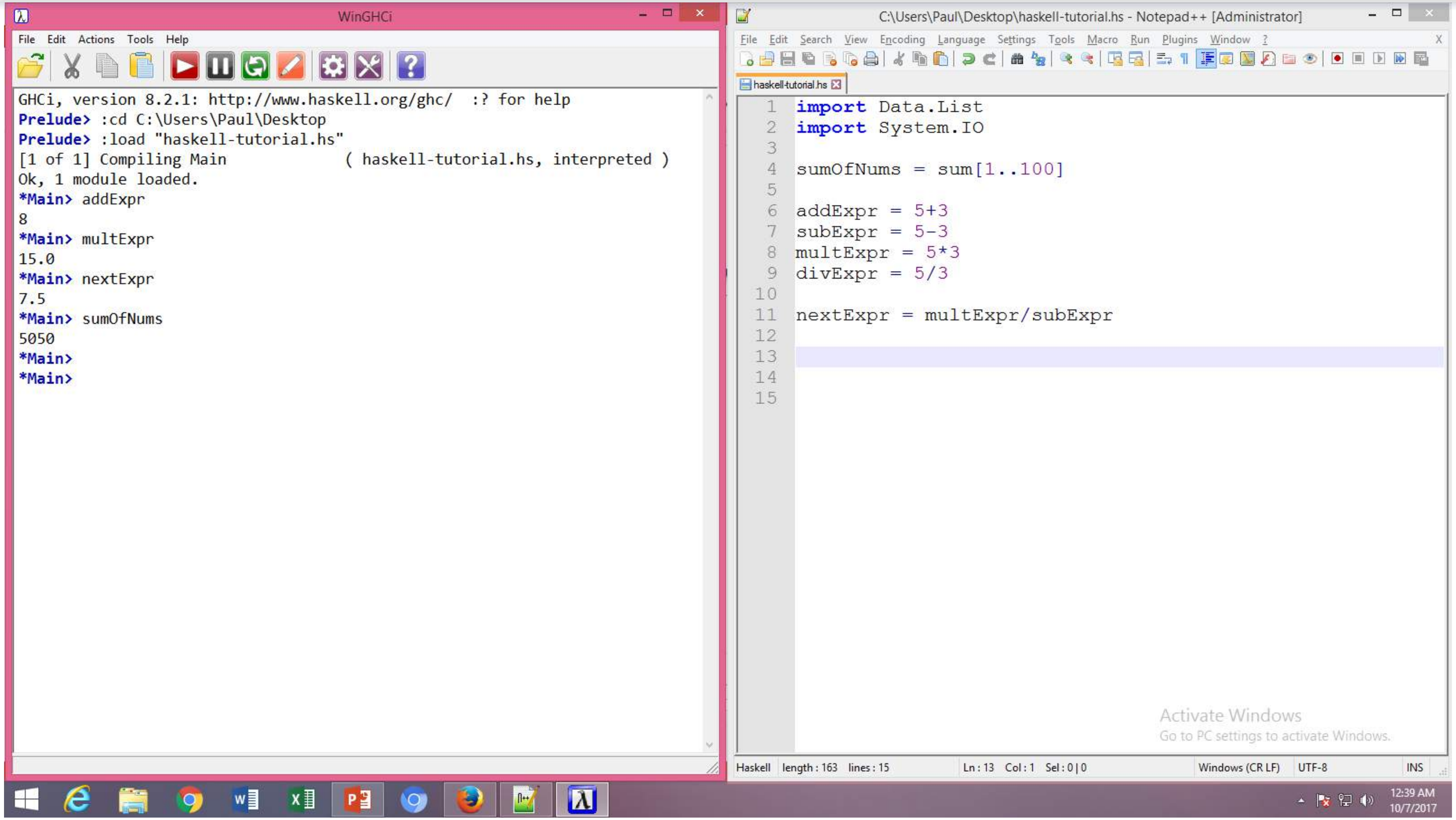
Expressions


WinGHCi
File Edit Actions Tools Help
C:\Users\Paul\Desktop\haskell-tutorial.hs - Notepad++ [Administrator]
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
haskell-tutorial.hs
Haskell length : 163 lines : 15 Ln : 13 Col : 1 Sel : 0 | 0 Windows (CR LF) UTF-8 INS
Activate Windows
Go to PC settings to activate Windows.
12:37 AM
10/7/2017

```
GHCi, version 8.2.1: http://www.haskell.org/ghc/  :? for help
Prelude> :cd C:\Users\Paul\Desktop
Prelude> :load "haskell-tutorial.hs"
[1 of 1] Compiling Main             ( haskell-tutorial.hs, interpreted )
Ok, 1 module loaded.
*Main> addExpr
8
*Main> multExpr
15.0
*Main> nextExpr
7.5
*Main>
```

```haskell
import Data.List
import System.IO

sumOfNums = sum[1..100]

addExpr = 5+3
subExpr = 5-3
multExpr = 5*3
divExpr = 5/3

nextExpr = multExpr/subExpr



```

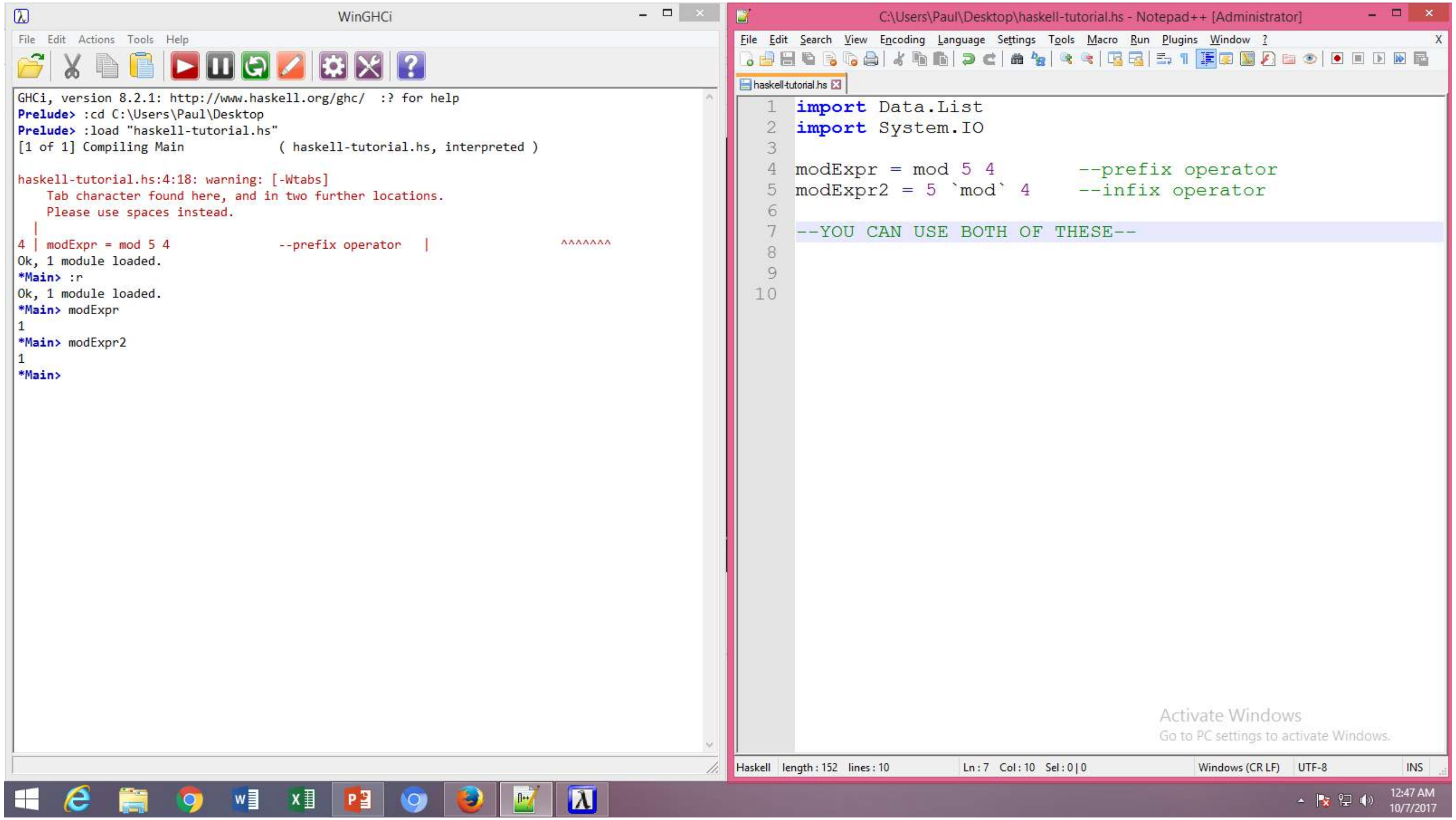
Expressions


WinGHCi
File Edit Actions Tools Help
C:\Users\Paul\Desktop\haskell-tutorial.hs - Notepad++ [Administrator]
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
haskell-tutorial.hs
Activate Windows
Go to PC settings to activate Windows.
Haskell length : 163 lines : 15 Ln : 13 Col : 1 Sel : 0 | 0 Windows (CR LF) UTF-8 INS
12:39 AM
10/7/2017

```
GHCi, version 8.2.1: http://www.haskell.org/ghc/  :? for help
Prelude> :cd C:\Users\Paul\Desktop
Prelude> :load "haskell-tutorial.hs"
[1 of 1] Compiling Main             ( haskell-tutorial.hs, interpreted )
Ok, 1 module loaded.
*Main> addExpr
8
*Main> multExpr
15.0
*Main> nextExpr
7.5
*Main> sumOfNums
5050
*Main>
*Main>
```

```haskell
import Data.List
import System.IO

sumOfNums = sum[1..100]

addExpr = 5+3
subExpr = 5-3
multExpr = 5*3
divExpr = 5/3

nextExpr = multExpr/subExpr
```

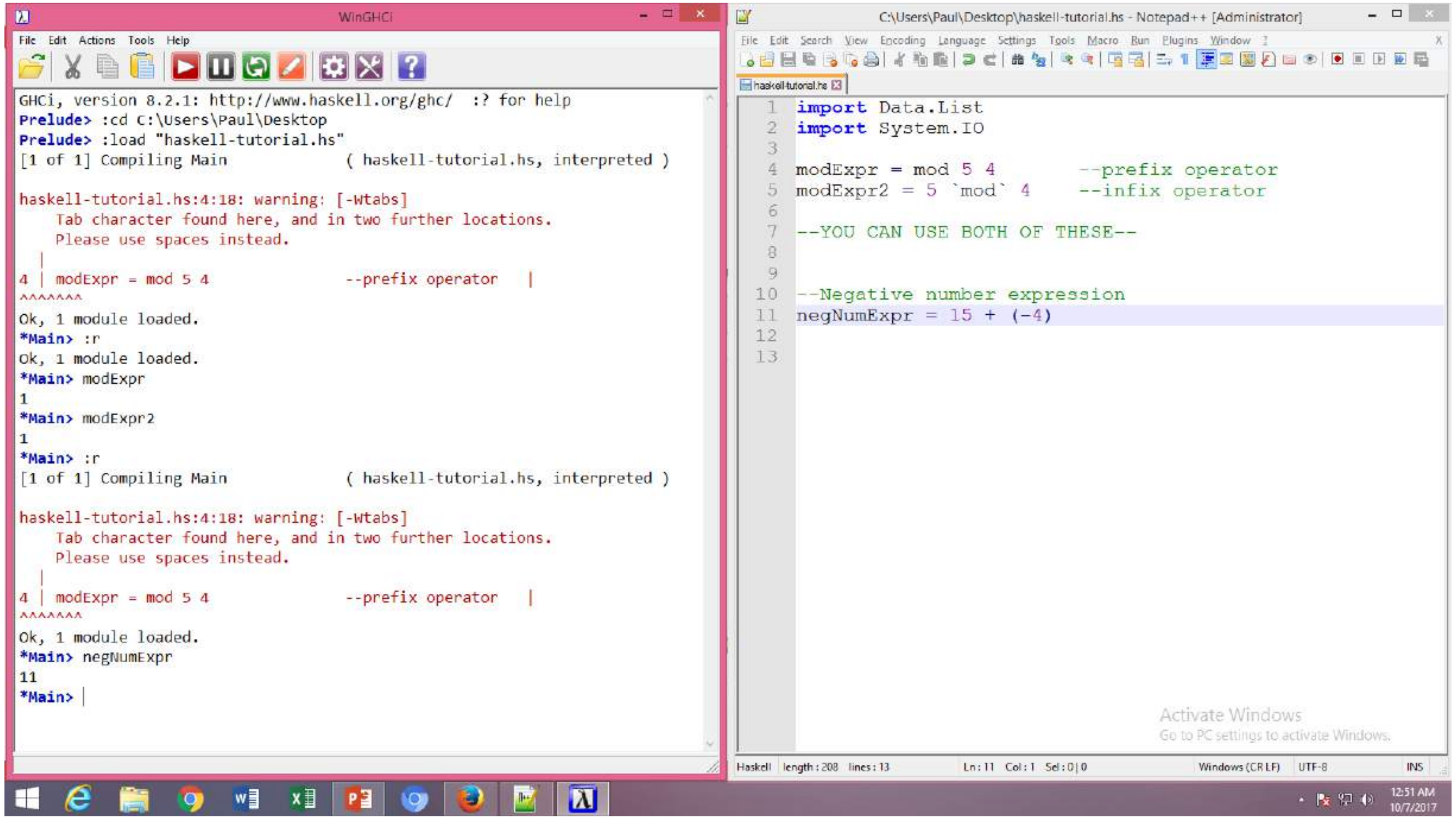
Infix and Prefix Operator


WinGHCi
File Edit Actions Tools Help
C:\Users\Paul\Desktop\haskell-tutorial.hs - Notepad++ [Administrator]
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
haskell-tutorial.hs
Activate Windows
Go to PC settings to activate Windows.
Haskell length : 152 lines : 10 Ln : 7 Col : 10 Sel : 0 | 0 Windows (CR LF) UTF-8 INS
12:47 AM
10/7/2017

```
GHCi, version 8.2.1: http://www.haskell.org/ghc/  :? for help
Prelude> :cd C:\Users\Paul\Desktop
Prelude> :load "haskell-tutorial.hs"
[1 of 1] Compiling Main             ( haskell-tutorial.hs, interpreted )

haskell-tutorial.hs:4:18: warning: [-Wtabs]
    Tab character found here, and in two further locations.
    Please use spaces instead.
  |
4 | modExpr = mod 5 4               --prefix operator   |                    ^^^^^^^
Ok, 1 module loaded.
*Main> :r
Ok, 1 module loaded.
*Main> modExpr
1
*Main> modExpr2
1
*Main>
```

```haskell
import Data.List
import System.IO

modExpr = mod 5 4           --prefix operator
modExpr2 = 5 `mod` 4        --infix operator

--YOU CAN USE BOTH OF THESE--



```

Negative Number Expression



WinGHCi
File Edit Actions Tools Help
C:\Users\Paul\Desktop\haskell-tutorial.hs - Notepad++ [Administrator]
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
haskell-tutorial.hs
Haskell length : 208 lines : 13 Ln : 11 Col : 1 Sel : 0 | 0 Windows (CR LF) UTF-8 INS
Activate Windows
Go to PC settings to activate Windows.
12:51 AM
10/7/2017

```
GHCi, version 8.2.1: http://www.haskell.org/ghc/  :? for help
Prelude> :cd C:\Users\Paul\Desktop
Prelude> :load "haskell-tutorial.hs"
[1 of 1] Compiling Main             ( haskell-tutorial.hs, interpreted )

haskell-tutorial.hs:4:18: warning: [-Wtabs]
    Tab character found here, and in two further locations.
    Please use spaces instead.
  |
4 | modExpr = mod 5 4              --prefix operator    |
^^^^^^^
Ok, 1 module loaded.
*Main> :r
Ok, 1 module loaded.
*Main> modExpr
1
*Main> modExpr2
1
*Main> :r
[1 of 1] Compiling Main             ( haskell-tutorial.hs, interpreted )

haskell-tutorial.hs:4:18: warning: [-Wtabs]
    Tab character found here, and in two further locations.
    Please use spaces instead.
  |
4 | modExpr = mod 5 4              --prefix operator    |
^^^^^^^
Ok, 1 module loaded.
*Main> negNumExpr
11
*Main>
```

```haskell
import Data.List
import System.IO

modExpr = mod 5 4          --prefix operator
modExpr2 = 5 `mod` 4       --infix operator

--YOU CAN USE BOTH OF THESE--


--Negative number expression
negNumExpr = 15 + (-4)

```

Other built-in Math Function

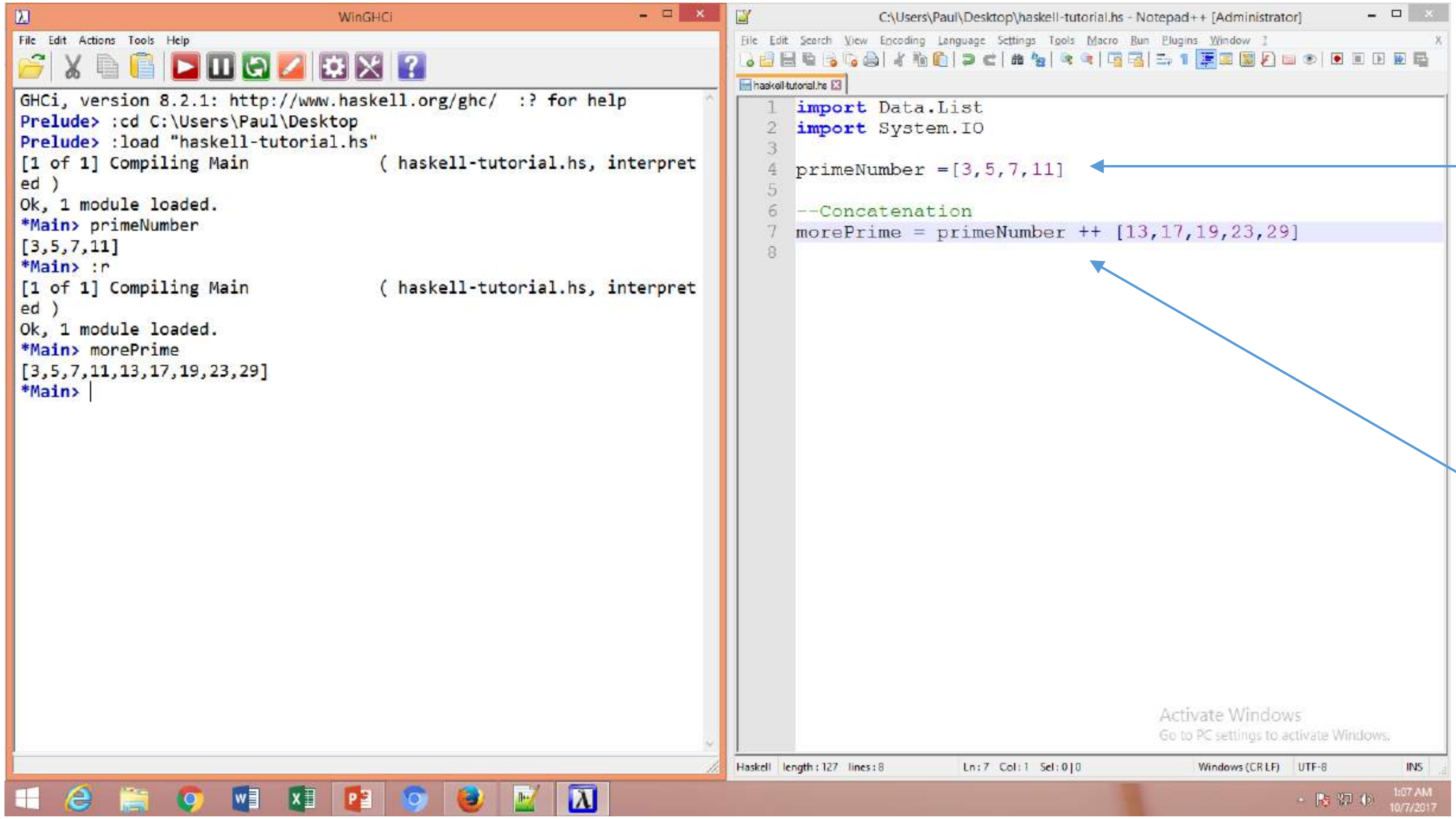
- piVal = pi
- ePow9 = exp 9
- logOf9 = log 9
- Squared9 = 9 ** 2
- truncateVal = truncate 9.999
- roundVal = round 9.999
- ceilingVal = ceiling 9.999
- floorVal = floor 9.999
- Also
  - sin, cos, tan, asign, acos, atan, signh, cosh, tanh, asignh, acosh, atanh

Other built-in Math Function

- piVal = pi
- ePow9 = exp 9
- logOf9 = log 9
- Squared9 = 9 ** 2
- truncateVal = truncate 9.999
- roundVal = round 9.999
- ceilingVal = ceiling 9.999
- floorVal = floor 9.999
- Also
  - sin, cos, tan, asign, acos, atan, signh, cosh, tanh, asignh, acosh, atanh

EXPLORE THESE

List - Concatenation


WinGHCi
File Edit Actions Tools Help
C:\Users\Paul\Desktop\haskell-tutorial.hs - Notepad++ [Administrator]
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
haskell-tutorial.hs
Activate Windows
Go to PC settings to activate Windows.
Haskell length : 127 lines : 8 Ln : 7 Col : 1 Sel : 0 | 0 Windows (CR LF) UTF-8 INS
1:07 AM
10/7/2017

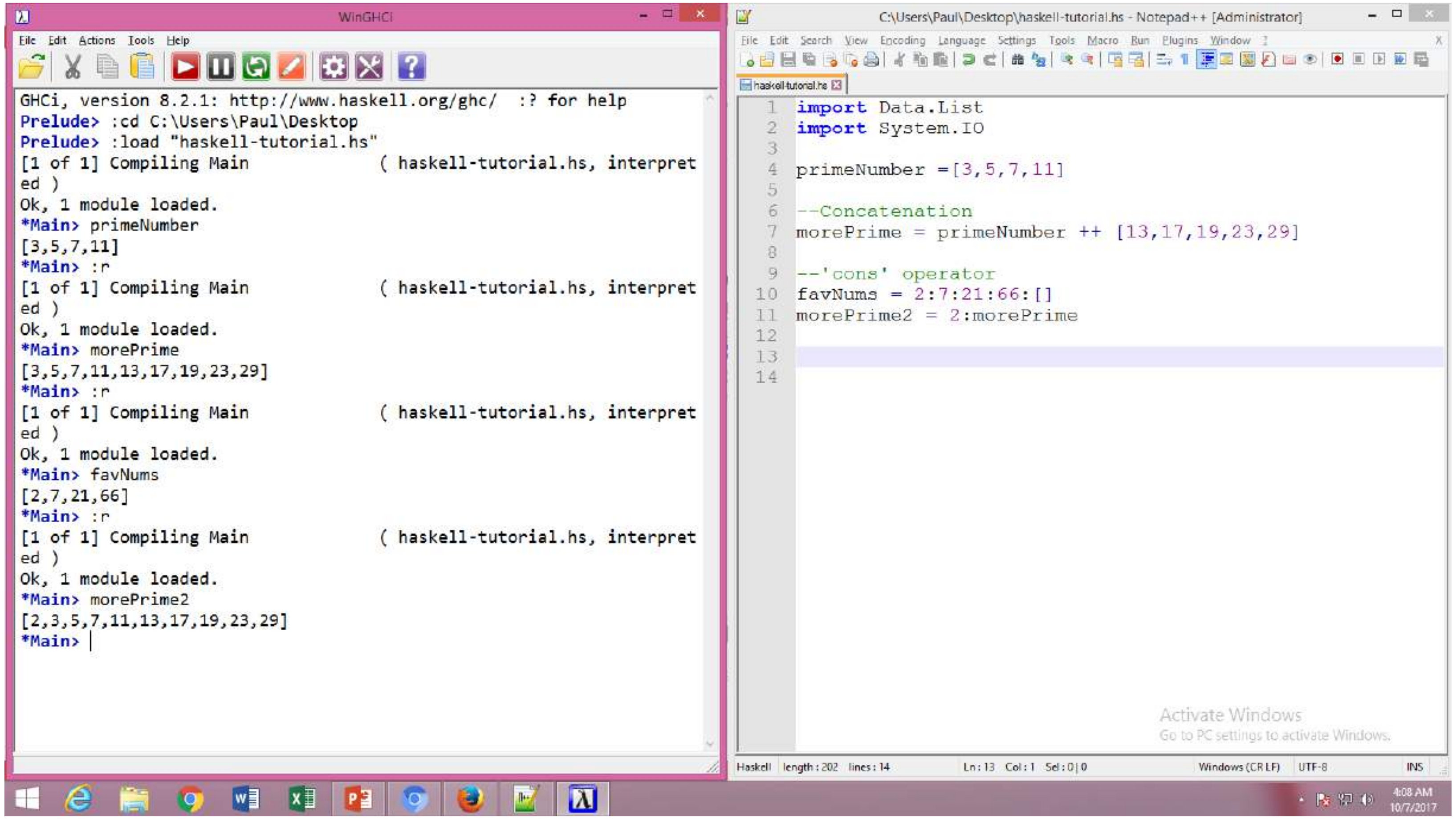
```
GHCi, version 8.2.1: http://www.haskell.org/ghc/  :? for help
Prelude> :cd C:\Users\Paul\Desktop
Prelude> :load "haskell-tutorial.hs"
[1 of 1] Compiling Main             ( haskell-tutorial.hs, interpret
ed )
Ok, 1 module loaded.
*Main> primeNumber
[3,5,7,11]
*Main> :r
[1 of 1] Compiling Main             ( haskell-tutorial.hs, interpret
ed )
Ok, 1 module loaded.
*Main> morePrime
[3,5,7,11,13,17,19,23,29]
*Main>
```

```haskell
import Data.List
import System.IO

primeNumber =[3,5,7,11]

--Concatenation
morePrime = primeNumber ++ [13,17,19,23,29]

```

Define a list

Concatenation of two lists

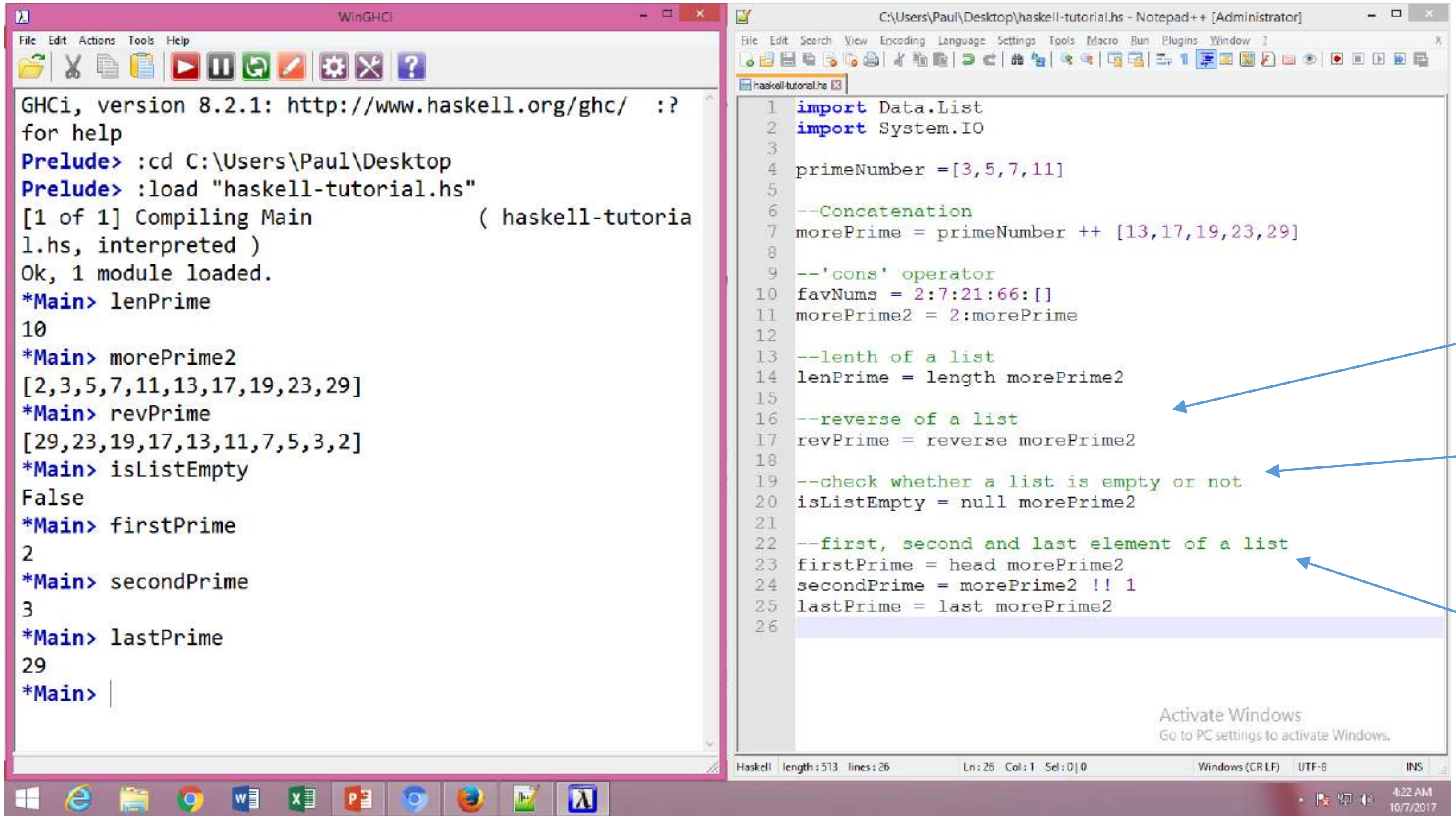
List – ‘cons’ operator


WinGHCi
File Edit Actions Tools Help
C:\Users\Paul\Desktop\haskell-tutorial.hs - Notepad++ [Administrator]
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
haskell-tutorial.hs
Haskell length : 202 lines : 14 Ln : 13 Col : 1 Sel : 0 | 0 Windows (CR LF) UTF-8 INS
4:08 AM
10/7/2017

```
GHCi, version 8.2.1: http://www.haskell.org/ghc/  :? for help
Prelude> :cd C:\Users\Paul\Desktop
Prelude> :load "haskell-tutorial.hs"
[1 of 1] Compiling Main             ( haskell-tutorial.hs, interpret
ed )
Ok, 1 module loaded.
*Main> primeNumber
[3,5,7,11]
*Main> :r
[1 of 1] Compiling Main             ( haskell-tutorial.hs, interpret
ed )
Ok, 1 module loaded.
*Main> morePrime
[3,5,7,11,13,17,19,23,29]
*Main> :r
[1 of 1] Compiling Main             ( haskell-tutorial.hs, interpret
ed )
Ok, 1 module loaded.
*Main> favNums
[2,7,21,66]
*Main> :r
[1 of 1] Compiling Main             ( haskell-tutorial.hs, interpret
ed )
Ok, 1 module loaded.
*Main> morePrime2
[2,3,5,7,11,13,17,19,23,29]
*Main>
```

```haskell
import Data.List
import System.IO

primeNumber =[3,5,7,11]

--Concatenation
morePrime = primeNumber ++ [13,17,19,23,29]

--'cons' operator
favNums = 2:7:21:66:[]
morePrime2 = 2:morePrime



```

More Operations on List


WinGHCi
File Edit Actions Tools Help
C:\Users\Paul\Desktop\haskell-tutorial.hs - Notepad++ [Administrator]
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
haskell-tutorial.hs
Activate Windows
Go to PC settings to activate Windows.
Haskell length : 513 lines : 26 Ln : 26 Col : 1 Sel : 0 | 0 Windows (CR LF) UTF-8 INS
4:22 AM
10/7/2017

```
GHCi, version 8.2.1: http://www.haskell.org/ghc/  :?
for help
Prelude> :cd C:\Users\Paul\Desktop
Prelude> :load "haskell-tutorial.hs"
[1 of 1] Compiling Main             ( haskell-tutoria
l.hs, interpreted )
Ok, 1 module loaded.
*Main> lenPrime
10
*Main> morePrime2
[2,3,5,7,11,13,17,19,23,29]
*Main> revPrime
[29,23,19,17,13,11,7,5,3,2]
*Main> isListEmpty
False
*Main> firstPrime
2
*Main> secondPrime
3
*Main> lastPrime
29
*Main>
```

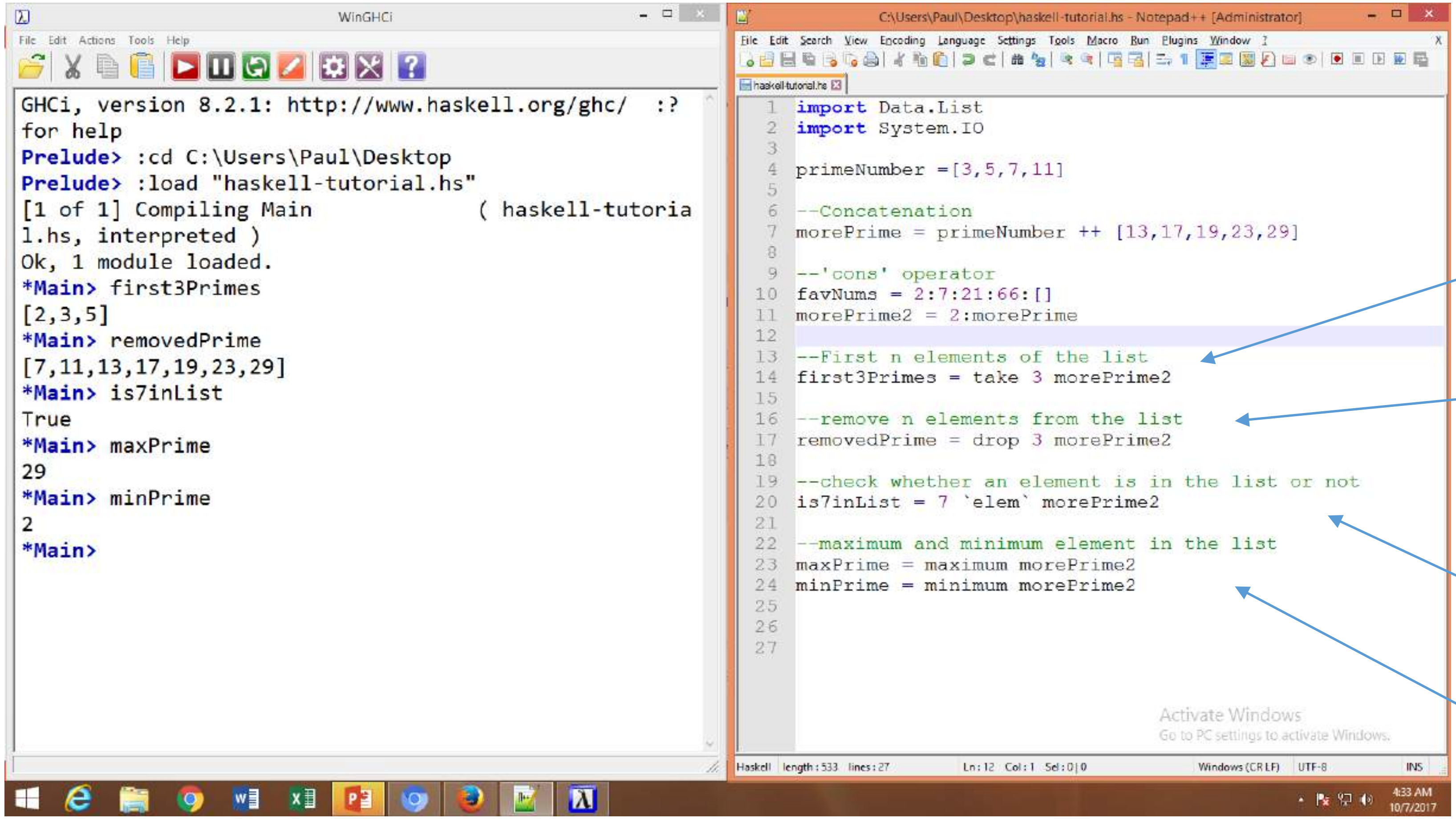
```haskell
import Data.List
import System.IO

primeNumber =[3,5,7,11]

--Concatenation
morePrime = primeNumber ++ [13,17,19,23,29]

--'cons' operator
favNums = 2:7:21:66:[]
morePrime2 = 2:morePrime

--lenth of a list
lenPrime = length morePrime2

--reverse of a list
revPrime = reverse morePrime2

--check whether a list is empty or not
isListEmpty = null morePrime2

--first, second and last element of a list
firstPrime = head morePrime2
secondPrime = morePrime2 !! 1
lastPrime = last morePrime2

```

reverse

List empty?

particular element

More Operations on List


WinGHCi
File Edit Actions Tools Help
C:\Users\Paul\Desktop\haskell-tutorial.hs - Notepad++ [Administrator]
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
haskell-tutorial.hs
Activate Windows
Go to PC settings to activate Windows.
Haskell length : 533 lines : 27 Ln : 12 Col : 1 Sel : 0 | 0 Windows (CR LF) UTF-8 INS
4:33 AM
10/7/2017

```
GHCi, version 8.2.1: http://www.haskell.org/ghc/  :?
for help
Prelude> :cd C:\Users\Paul\Desktop
Prelude> :load "haskell-tutorial.hs"
[1 of 1] Compiling Main             ( haskell-tutoria
l.hs, interpreted )
Ok, 1 module loaded.
*Main> first3Primes
[2,3,5]
*Main> removedPrime
[7,11,13,17,19,23,29]
*Main> is7inList
True
*Main> maxPrime
29
*Main> minPrime
2
*Main>
```

```haskell
import Data.List
import System.IO

primeNumber =[3,5,7,11]

--Concatenation
morePrime = primeNumber ++ [13,17,19,23,29]

--'cons' operator
favNums = 2:7:21:66:[]
morePrime2 = 2:morePrime

--First n elements of the list
first3Primes = take 3 morePrime2

--remove n elements from the list
removedPrime = drop 3 morePrime2

--check whether an element is in the list or not
is7inList = 7 `elem` morePrime2

--maximum and minimum element in the list
maxPrime = maximum morePrime2
minPrime = minimum morePrime2
```

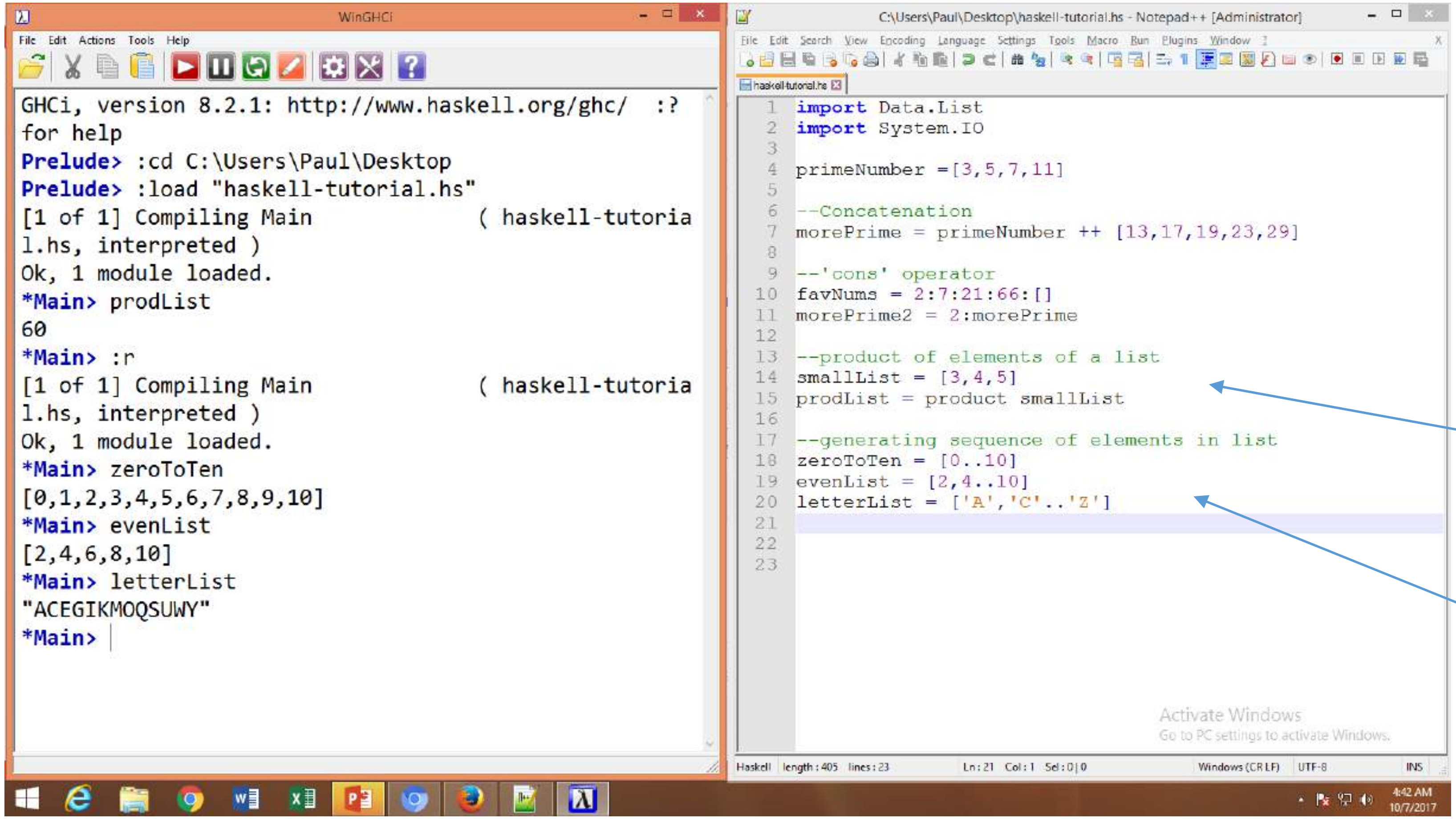
first n element

removing first n element

finding an element

max and min

More Operations on List


WinGHCi
File Edit Actions Tools Help
C:\Users\Paul\Desktop\haskell-tutorial.hs - Notepad++ [Administrator]
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
haskell-tutorial.hs
Haskell length : 405 lines : 23 Ln : 21 Col : 1 Sel : 0 | 0 Windows (CR LF) UTF-8 INS
Activate Windows
Go to PC settings to activate Windows.
4:42 AM
10/7/2017

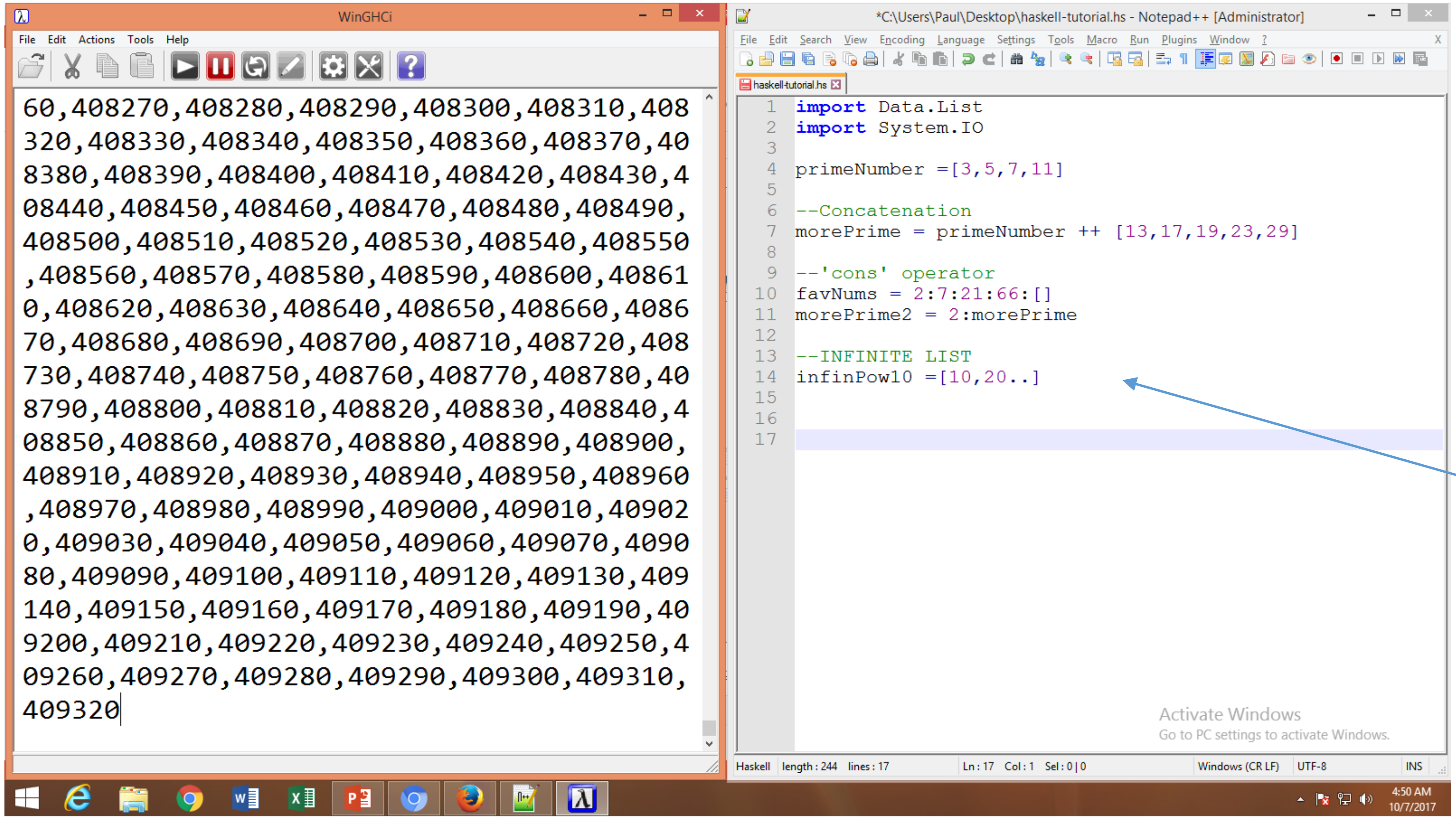
```
GHCi, version 8.2.1: http://www.haskell.org/ghc/  :?
for help
Prelude> :cd C:\Users\Paul\Desktop
Prelude> :load "haskell-tutorial.hs"
[1 of 1] Compiling Main             ( haskell-tutoria
l.hs, interpreted )
Ok, 1 module loaded.
*Main> prodList
60
*Main> :r
[1 of 1] Compiling Main             ( haskell-tutoria
l.hs, interpreted )
Ok, 1 module loaded.
*Main> zeroToTen
[0,1,2,3,4,5,6,7,8,9,10]
*Main> evenList
[2,4,6,8,10]
*Main> letterList
"ACEGIKMOQSUWY"
*Main>
```

```haskell
import Data.List
import System.IO

primeNumber =[3,5,7,11]

--Concatenation
morePrime = primeNumber ++ [13,17,19,23,29]

--'cons' operator
favNums = 2:7:21:66:[]
morePrime2 = 2:morePrime

--product of elements of a list
smallList = [3,4,5]
prodList = product smallList

--generating sequence of elements in list
zeroToTen = [0..10]
evenList = [2,4..10]
letterList = ['A','C'..'Z']
```

product

sequence

More Operations on List

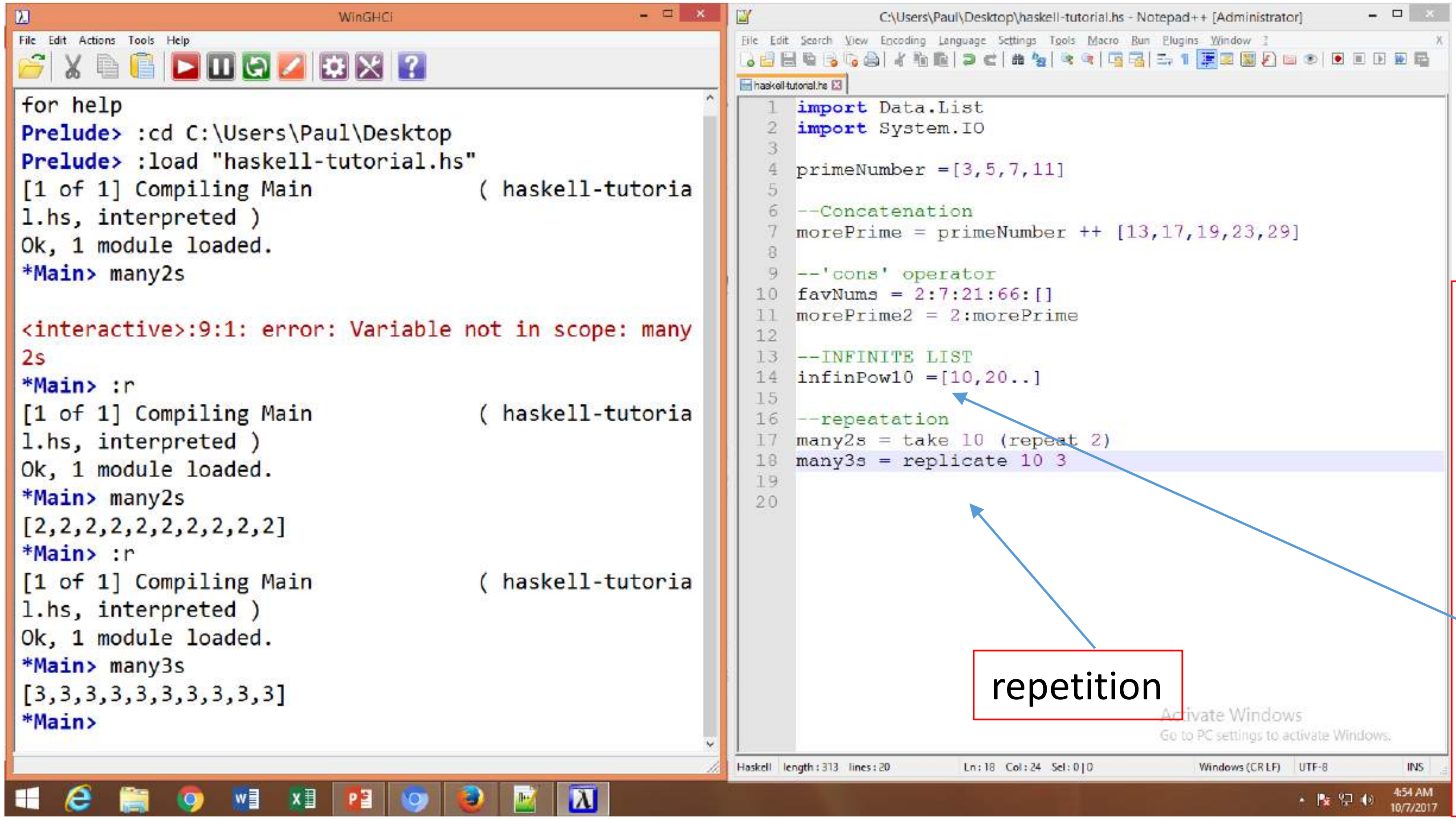

WinGHCi
File Edit Actions Tools Help
*C:\Users\Paul\Desktop\haskell-tutorial.hs - Notepad++ [Administrator]
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
haskell-tutorial.hs
Haskell length : 244 lines : 17 Ln : 17 Col : 1 Sel : 0 | 0 Windows (CR LF) UTF-8 INS
4:50 AM
10/7/2017

```
60,408270,408280,408290,408300,408310,408
320,408330,408340,408350,408360,408370,40
8380,408390,408400,408410,408420,408430,4
08440,408450,408460,408470,408480,408490,
408500,408510,408520,408530,408540,408550
,408560,408570,408580,408590,408600,40861
0,408620,408630,408640,408650,408660,4086
70,408680,408690,408700,408710,408720,408
730,408740,408750,408760,408770,408780,40
8790,408800,408810,408820,408830,408840,4
08850,408860,408870,408880,408890,408900,
408910,408920,408930,408940,408950,408960
,408970,408980,408990,409000,409010,40902
0,409030,409040,409050,409060,409070,4090
80,409090,409100,409110,409120,409130,409
140,409150,409160,409170,409180,409190,40
9200,409210,409220,409230,409240,409250,4
09260,409270,409280,409290,409300,409310,
409320
```

```haskell
import Data.List
import System.IO

primeNumber =[3,5,7,11]

--Concatenation
morePrime = primeNumber ++ [13,17,19,23,29]

--'cons' operator
favNums = 2:7:21:66:[]
morePrime2 = 2:morePrime

--INFINITE LIST
infinPow10 =[10,20..]
```

infinite list

More Operations on List


WinGHCi
File Edit Actions Tools Help
C:\Users\Paul\Desktop\haskell-tutorial.hs - Notepad++ [Administrator]
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
haskell-tutorial.hs
Haskell length : 313 lines : 20 Ln : 18 Col : 24 Sel : 0 | 0 Windows (CR LF) UTF-8 INS
Activate Windows
Go to PC settings to activate Windows.
4:54 AM
10/7/2017

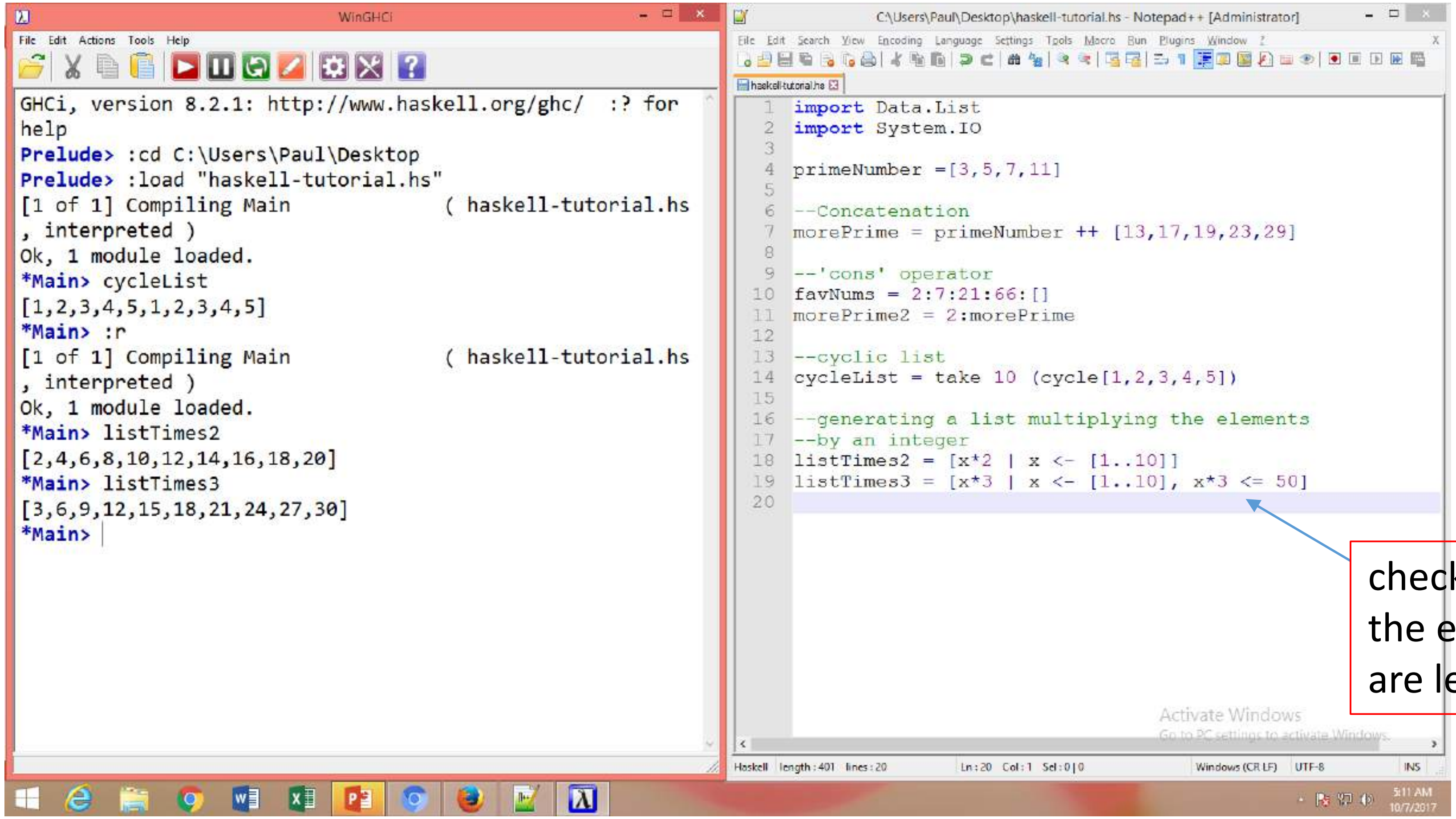
```
for help
Prelude> :cd C:\Users\Paul\Desktop
Prelude> :load "haskell-tutorial.hs"
[1 of 1] Compiling Main             ( haskell-tutoria
l.hs, interpreted )
Ok, 1 module loaded.
*Main> many2s

<interactive>:9:1: error: Variable not in scope: many
2s
*Main> :r
[1 of 1] Compiling Main             ( haskell-tutoria
l.hs, interpreted )
Ok, 1 module loaded.
*Main> many2s
[2,2,2,2,2,2,2,2,2,2]
*Main> :r
[1 of 1] Compiling Main             ( haskell-tutoria
l.hs, interpreted )
Ok, 1 module loaded.
*Main> many3s
[3,3,3,3,3,3,3,3,3,3]
*Main>
```

```haskell
import Data.List
import System.IO

primeNumber =[3,5,7,11]

--Concatenation
morePrime = primeNumber ++ [13,17,19,23,29]

--'cons' operator
favNums = 2:7:21:66:[]
morePrime2 = 2:morePrime

--INFINITE LIST
infinPow10 =[10,20..]

--repeatation
many2s = take 10 (repeat 2)
many3s = replicate 10 3
```

repetition

One of the examples of advantages of laziness property and functional approach: here, the presence of ***infinite list*** does not affect other expressions/ functions in the program

More Operations on List

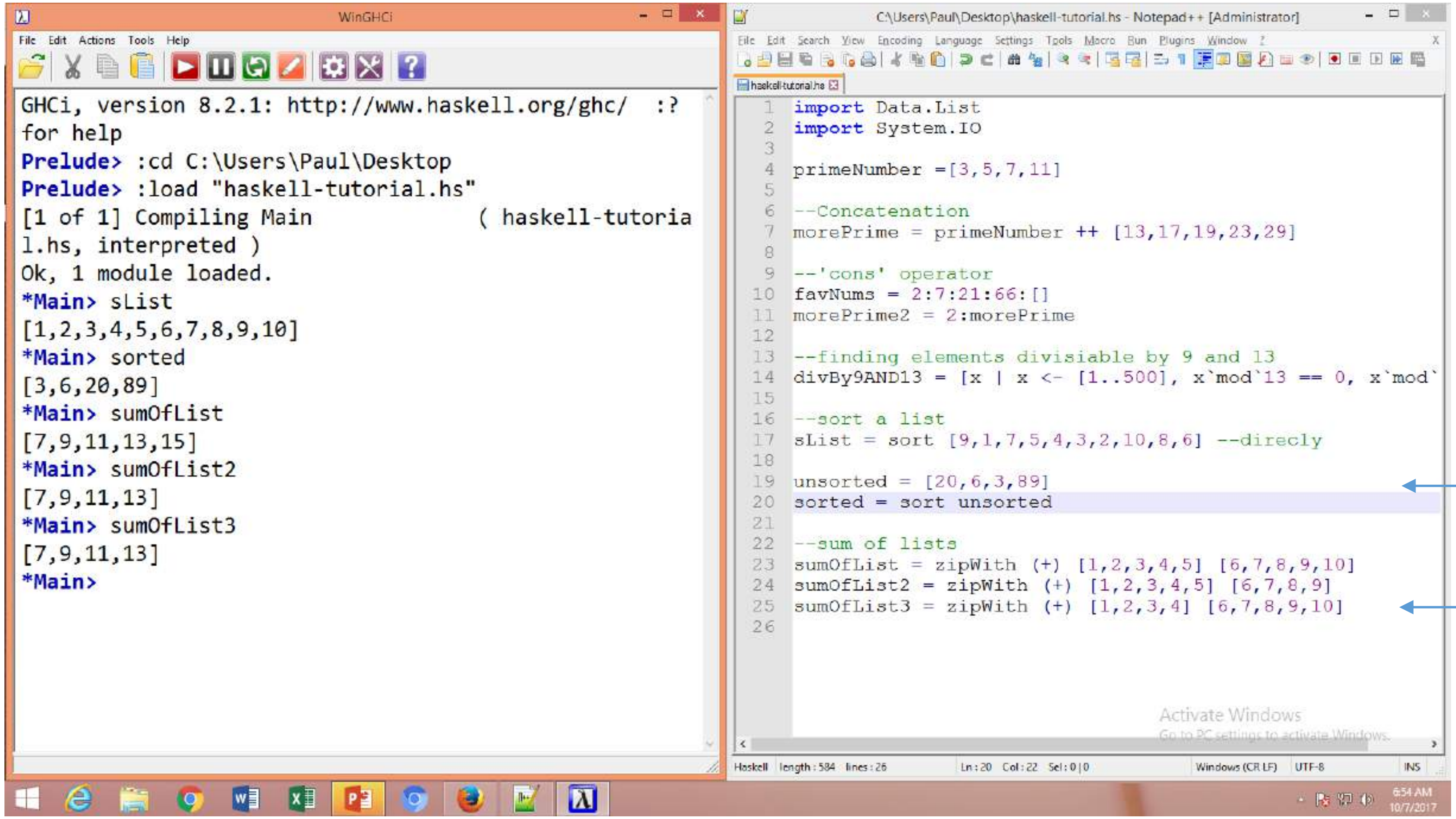

WinGHCi
File Edit Actions Tools Help
C:\Users\Paul\Desktop\haskell-tutorial.hs - Notepad++ [Administrator]
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
haskell-tutorial.hs
Activate Windows
Go to PC settings to activate Windows.
Haskell length : 401 lines : 20 Ln : 20 Col : 1 Sel : 0 | 0 Windows (CR LF) UTF-8 INS
5:11 AM
10/7/2017

```
GHCi, version 8.2.1: http://www.haskell.org/ghc/  :? for
help
Prelude> :cd C:\Users\Paul\Desktop
Prelude> :load "haskell-tutorial.hs"
[1 of 1] Compiling Main             ( haskell-tutorial.hs
, interpreted )
Ok, 1 module loaded.
*Main> cycleList
[1,2,3,4,5,1,2,3,4,5]
*Main> :r
[1 of 1] Compiling Main             ( haskell-tutorial.hs
, interpreted )
Ok, 1 module loaded.
*Main> listTimes2
[2,4,6,8,10,12,14,16,18,20]
*Main> listTimes3
[3,6,9,12,15,18,21,24,27,30]
*Main>
```

```haskell
import Data.List
import System.IO

primeNumber =[3,5,7,11]

--Concatenation
morePrime = primeNumber ++ [13,17,19,23,29]

--'cons' operator
favNums = 2:7:21:66:[]
morePrime2 = 2:morePrime

--cyclic list
cycleList = take 10 (cycle[1,2,3,4,5])

--generating a list multiplying the elements
--by an integer
listTimes2 = [x*2 | x <- [1..10]]
listTimes3 = [x*3 | x <- [1..10], x*3 <= 50]
```

check whether
the element generated
are less than 50 or not

More Operations on List


WinGHCi
File Edit Actions Tools Help
C:\Users\Paul\Desktop\haskell-tutorial.hs - Notepad++ [Administrator]
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
haskell-tutorial.hs
Haskell length : 584 lines : 26 Ln : 20 Col : 22 Sel : 0 | 0 Windows (CR LF) UTF-8 INS

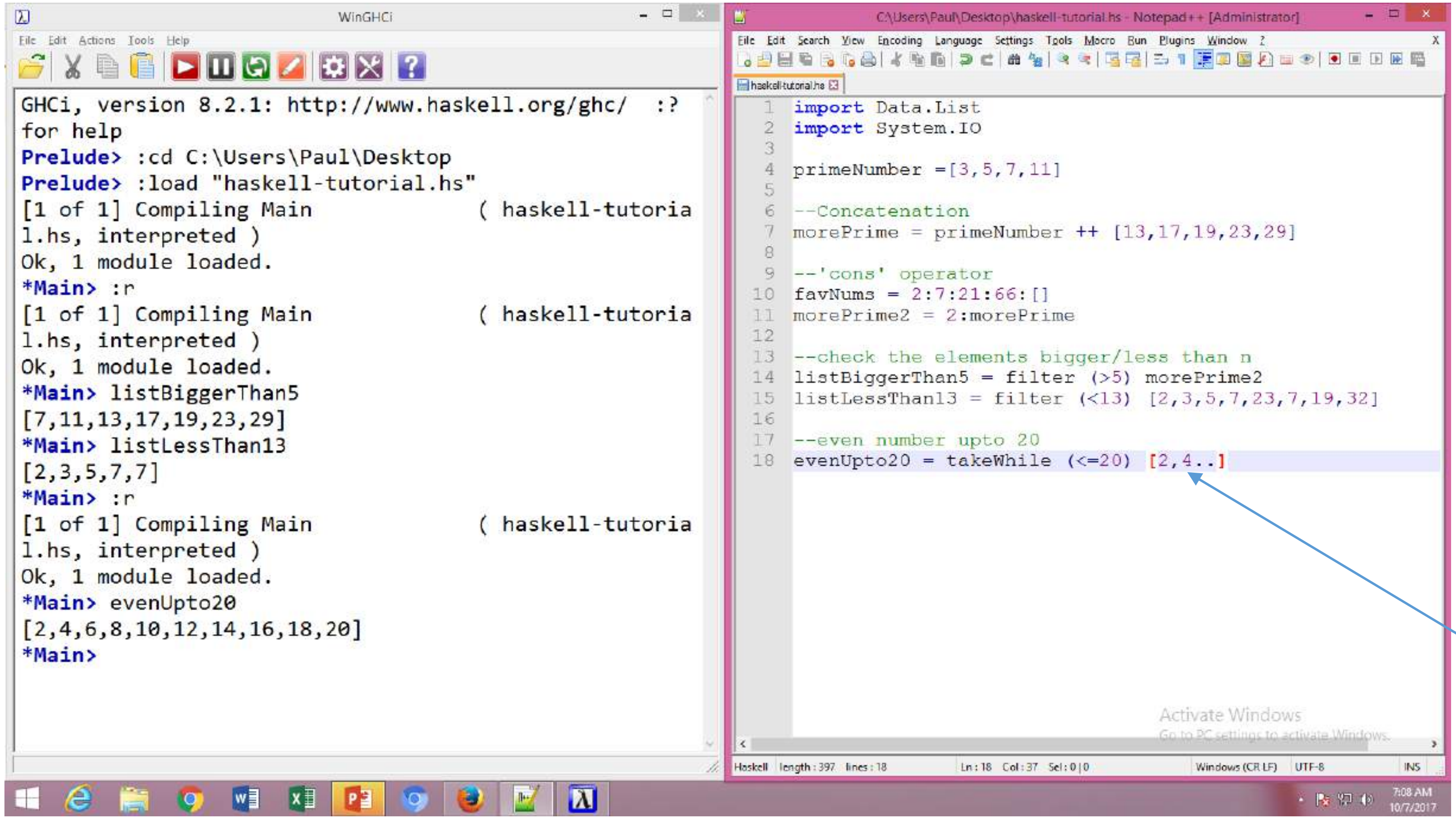
```
GHCi, version 8.2.1: http://www.haskell.org/ghc/  :?
for help
Prelude> :cd C:\Users\Paul\Desktop
Prelude> :load "haskell-tutorial.hs"
[1 of 1] Compiling Main             ( haskell-tutoria
l.hs, interpreted )
Ok, 1 module loaded.
*Main> sList
[1,2,3,4,5,6,7,8,9,10]
*Main> sorted
[3,6,20,89]
*Main> sumOfList
[7,9,11,13,15]
*Main> sumOfList2
[7,9,11,13]
*Main> sumOfList3
[7,9,11,13]
*Main>
```

```haskell
import Data.List
import System.IO

primeNumber =[3,5,7,11]

--Concatenation
morePrime = primeNumber ++ [13,17,19,23,29]

--'cons' operator
favNums = 2:7:21:66:[]
morePrime2 = 2:morePrime

--finding elements divisiable by 9 and 13
divBy9AND13 = [x | x <- [1..500], x`mod`13 == 0, x`mod`

--sort a list
sList = sort [9,1,7,5,4,3,2,10,8,6] --direcly

unsorted = [20,6,3,89]
sorted = sort unsorted

--sum of lists
sumOfList = zipWith (+) [1,2,3,4,5] [6,7,8,9,10]
sumOfList2 = zipWith (+) [1,2,3,4,5] [6,7,8,9]
sumOfList3 = zipWith (+) [1,2,3,4] [6,7,8,9,10]

```

sorting

summation

More Operations on List

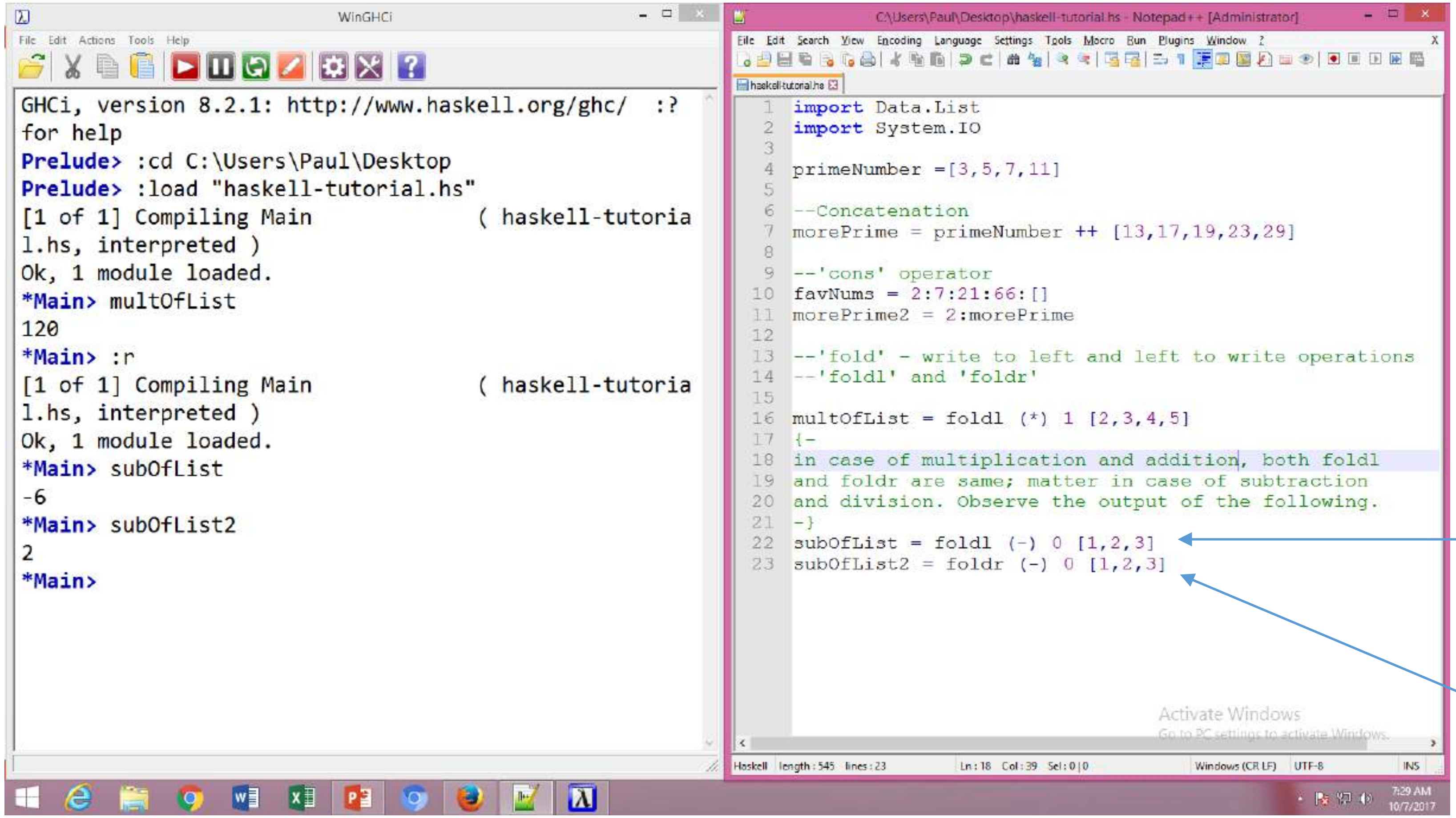

WinGHCi
File Edit Actions Tools Help
C:\Users\Paul\Desktop\haskell-tutorial.hs - Notepad++ [Administrator]
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
haskell-tutorial.hs
Activate Windows
Go to PC settings to activate Windows.
Haskell length : 397 lines : 18 Ln : 18 Col : 37 Sel : 0 | 0 Windows (CR LF) UTF-8 INS
7:08 AM
10/7/2017

```
GHCi, version 8.2.1: http://www.haskell.org/ghc/  :?
for help
Prelude> :cd C:\Users\Paul\Desktop
Prelude> :load "haskell-tutorial.hs"
[1 of 1] Compiling Main             ( haskell-tutoria
l.hs, interpreted )
Ok, 1 module loaded.
*Main> :r
[1 of 1] Compiling Main             ( haskell-tutoria
l.hs, interpreted )
Ok, 1 module loaded.
*Main> listBiggerThan5
[7,11,13,17,19,23,29]
*Main> listLessThan13
[2,3,5,7,7]
*Main> :r
[1 of 1] Compiling Main             ( haskell-tutoria
l.hs, interpreted )
Ok, 1 module loaded.
*Main> evenUpto20
[2,4,6,8,10,12,14,16,18,20]
*Main>
```

```haskell
import Data.List
import System.IO

primeNumber =[3,5,7,11]

--Concatenation
morePrime = primeNumber ++ [13,17,19,23,29]

--'cons' operator
favNums = 2:7:21:66:[]
morePrime2 = 2:morePrime

--check the elements bigger/less than n
listBiggerThan5 = filter (>5) morePrime2
listLessThan13 = filter (<13) [2,3,5,7,23,7,19,32]

--even number upto 20
evenUpto20 = takeWhile (<=20) [2,4..]
```

another example of laziness; although infinite list, check up to 20

More Operations on List


WinGHCi
File Edit Actions Tools Help
C:\Users\Paul\Desktop\haskell-tutorial.hs - Notepad++ [Administrator]
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
haskell-tutorial.hs
Activate Windows
Go to PC settings to activate Windows.
Haskell length : 545 lines : 23 Ln : 18 Col : 39 Sel : 0 | 0 Windows (CR LF) UTF-8 INS
7:29 AM
10/7/2017

```
GHCi, version 8.2.1: http://www.haskell.org/ghc/  :?
for help
Prelude> :cd C:\Users\Paul\Desktop
Prelude> :load "haskell-tutorial.hs"
[1 of 1] Compiling Main             ( haskell-tutoria
l.hs, interpreted )
Ok, 1 module loaded.
*Main> multOfList
120
*Main> :r
[1 of 1] Compiling Main             ( haskell-tutoria
l.hs, interpreted )
Ok, 1 module loaded.
*Main> subOfList
-6
*Main> subOfList2
2
*Main>
```

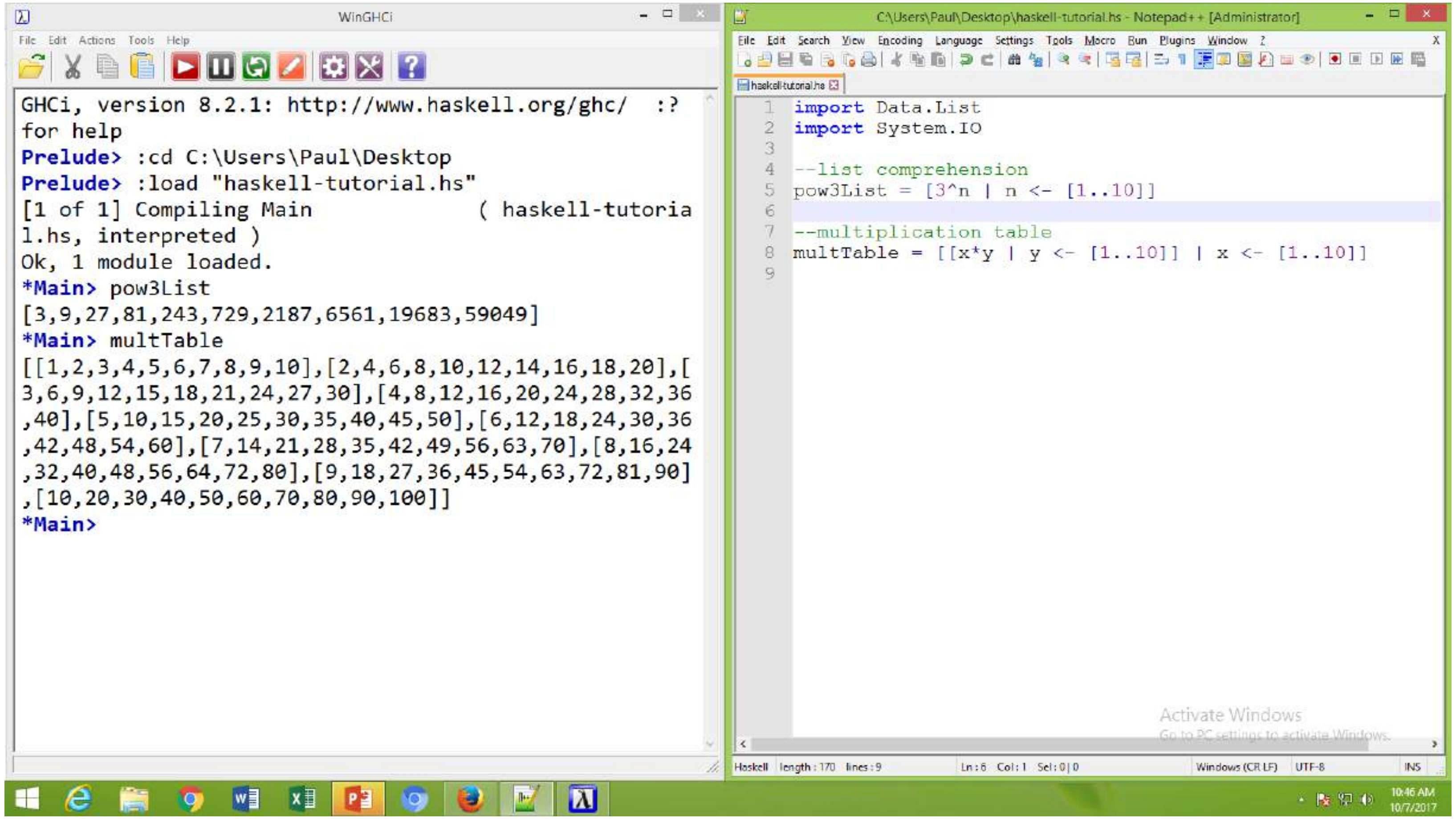
```haskell
import Data.List
import System.IO

primeNumber =[3,5,7,11]

--Concatenation
morePrime = primeNumber ++ [13,17,19,23,29]

--'cons' operator
favNums = 2:7:21:66:[]
morePrime2 = 2:morePrime

--'fold' - write to left and left to write operations
--'foldl' and 'foldr'

multOfList = foldl (*) 1 [2,3,4,5]
{-
in case of multiplication and addition, both foldl
and foldr are same; matter in case of subtraction
and division. Observe the output of the following.
-}
subOfList = foldl (-) 0 [1,2,3]
subOfList2 = foldr (-) 0 [1,2,3]
```

foldl & foldr

(((0-1)-2)-3)

1-(2-(3-0))

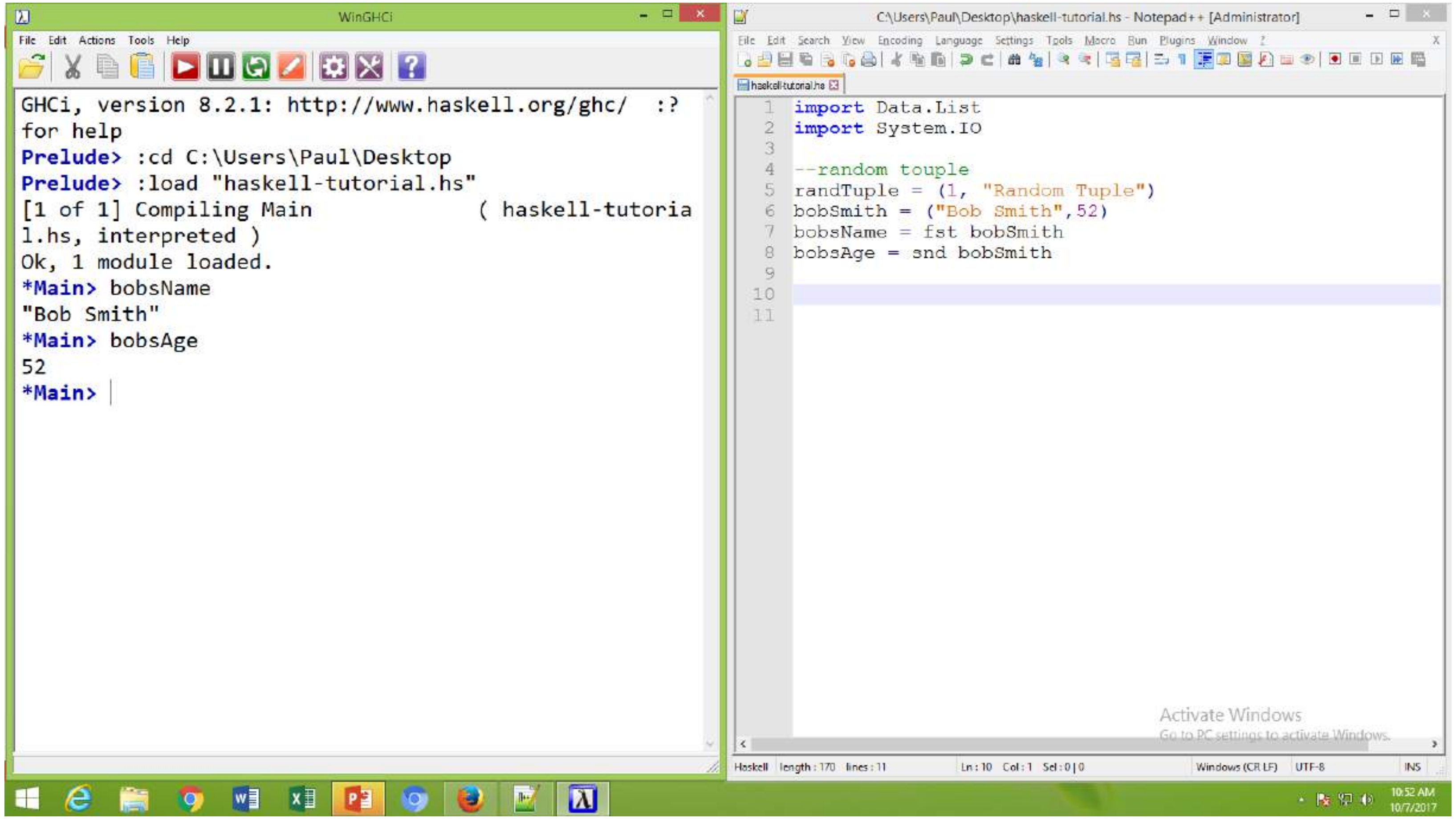
More Operations on List


WinGHCi
File Edit Actions Tools Help
C:\Users\Paul\Desktop\haskell-tutorial.hs - Notepad++ [Administrator]
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
haskell-tutorial.hs
Haskell length : 170 lines : 9 Ln : 6 Col : 1 Sel : 0 | 0 Windows (CR LF) UTF-8 INS
Activate Windows
Go to PC settings to activate Windows.
10:46 AM
10/7/2017

```
GHCi, version 8.2.1: http://www.haskell.org/ghc/  :?
for help
Prelude> :cd C:\Users\Paul\Desktop
Prelude> :load "haskell-tutorial.hs"
[1 of 1] Compiling Main             ( haskell-tutoria
l.hs, interpreted )
Ok, 1 module loaded.
*Main> pow3List
[3,9,27,81,243,729,2187,6561,19683,59049]
*Main> multTable
[[1,2,3,4,5,6,7,8,9,10],[2,4,6,8,10,12,14,16,18,20],[
3,6,9,12,15,18,21,24,27,30],[4,8,12,16,20,24,28,32,36
,40],[5,10,15,20,25,30,35,40,45,50],[6,12,18,24,30,36
,42,48,54,60],[7,14,21,28,35,42,49,56,63,70],[8,16,24
,32,40,48,56,64,72,80],[9,18,27,36,45,54,63,72,81,90]
,[10,20,30,40,50,60,70,80,90,100]]
*Main>
```

```haskell
import Data.List
import System.IO

--list comprehension
pow3List = [3^n | n <- [1..10]]

--multiplication table
multTable = [[x*y | y <- [1..10]] | x <- [1..10]]

```

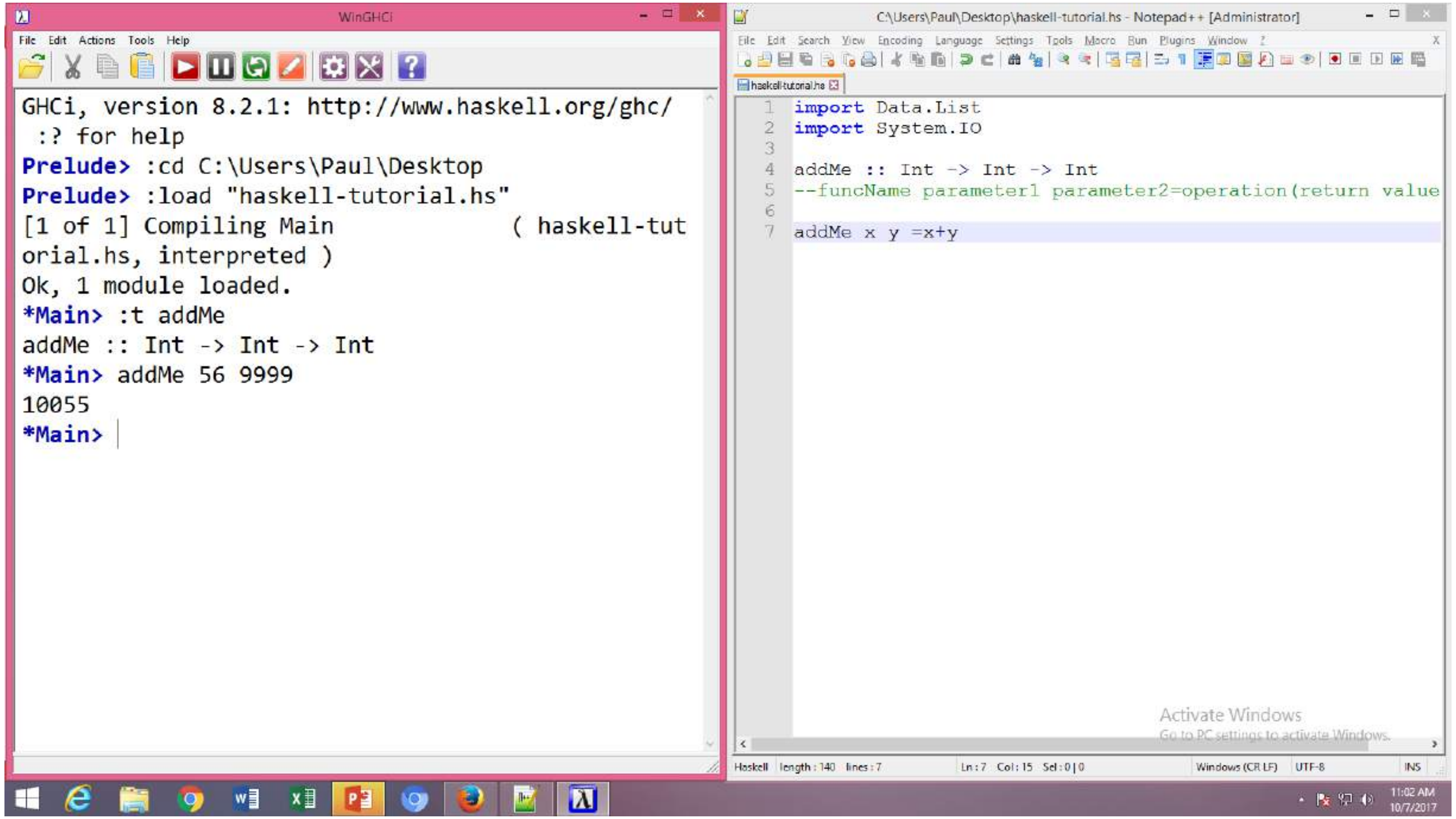
Multiple Data Type


WinGHCi
File Edit Actions Tools Help
C:\Users\Paul\Desktop\haskell-tutorial.hs - Notepad++ [Administrator]
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
haskell-tutorial.hs
Activate Windows
Go to PC settings to activate Windows.
Haskell length : 170 lines : 11 Ln : 10 Col : 1 Sel : 0 | 0 Windows (CR LF) UTF-8 INS
10:52 AM
10/7/2017

```
GHCi, version 8.2.1: http://www.haskell.org/ghc/  :?
for help
Prelude> :cd C:\Users\Paul\Desktop
Prelude> :load "haskell-tutorial.hs"
[1 of 1] Compiling Main             ( haskell-tutoria
l.hs, interpreted )
Ok, 1 module loaded.
*Main> bobsName
"Bob Smith"
*Main> bobsAge
52
*Main>
```

```haskell
import Data.List
import System.IO

--random touple
randTuple = (1, "Random Tuple")
bobSmith = ("Bob Smith",52)
bobsName = fst bobSmith
bobsAge = snd bobSmith
```

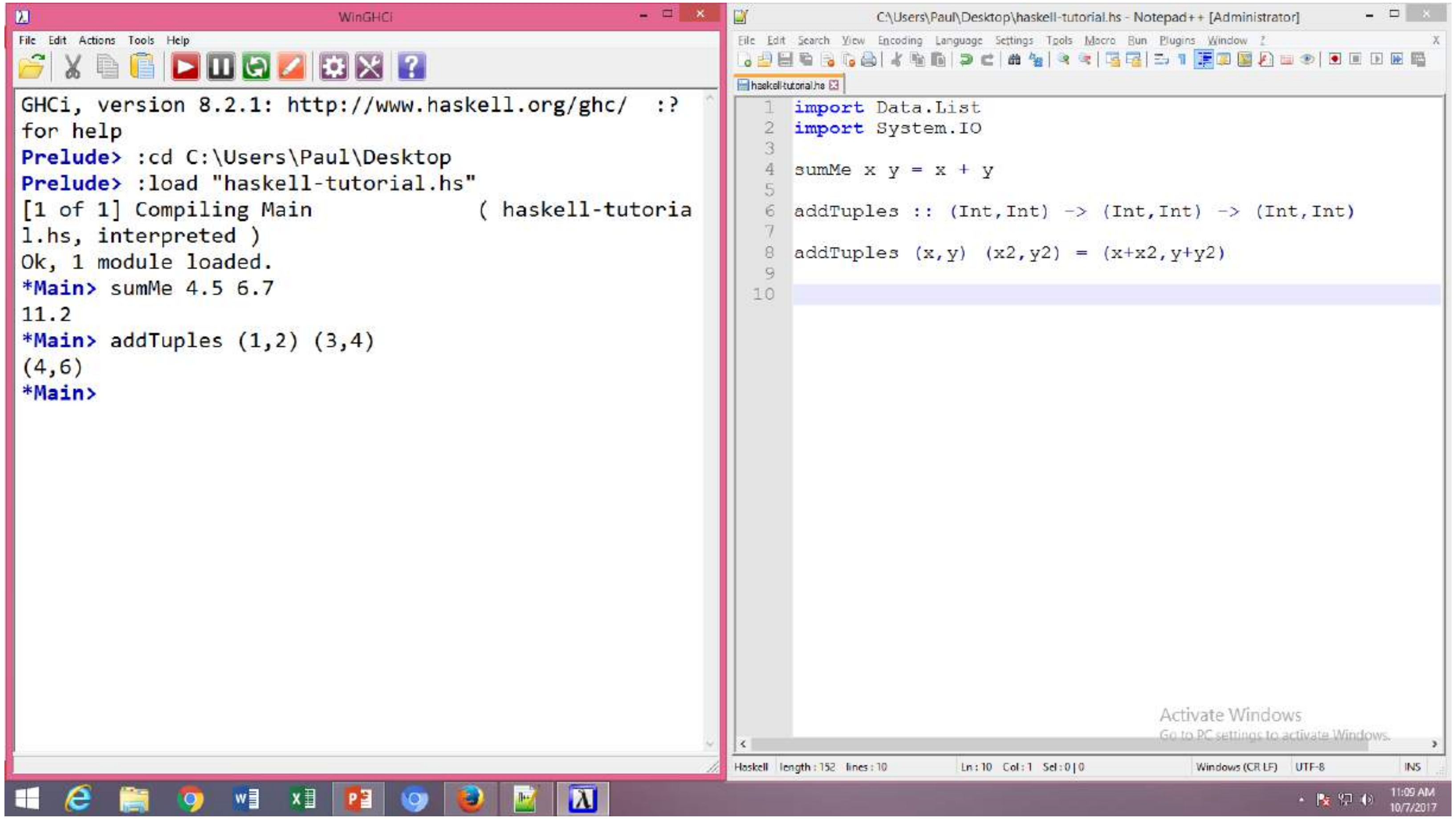
Function Declaration


WinGHCi
File Edit Actions Tools Help
C:\Users\Paul\Desktop\haskell-tutorial.hs - Notepad++ [Administrator]
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
haskell-tutorial.hs
Activate Windows
Go to PC settings to activate Windows.
Haskell length : 140 lines : 7 Ln : 7 Col : 15 Sel : 0 | 0 Windows (CR LF) UTF-8 INS
11:02 AM
10/7/2017

```
GHCi, version 8.2.1: http://www.haskell.org/ghc/
 :? for help
Prelude> :cd C:\Users\Paul\Desktop
Prelude> :load "haskell-tutorial.hs"
[1 of 1] Compiling Main             ( haskell-tut
orial.hs, interpreted )
Ok, 1 module loaded.
*Main> :t addMe
addMe :: Int -> Int -> Int
*Main> addMe 56 9999
10055
*Main>
```

```haskell
import Data.List
import System.IO

addMe :: Int -> Int -> Int
--funcName parameter1 parameter2=operation(return value

addMe x y =x+y
```

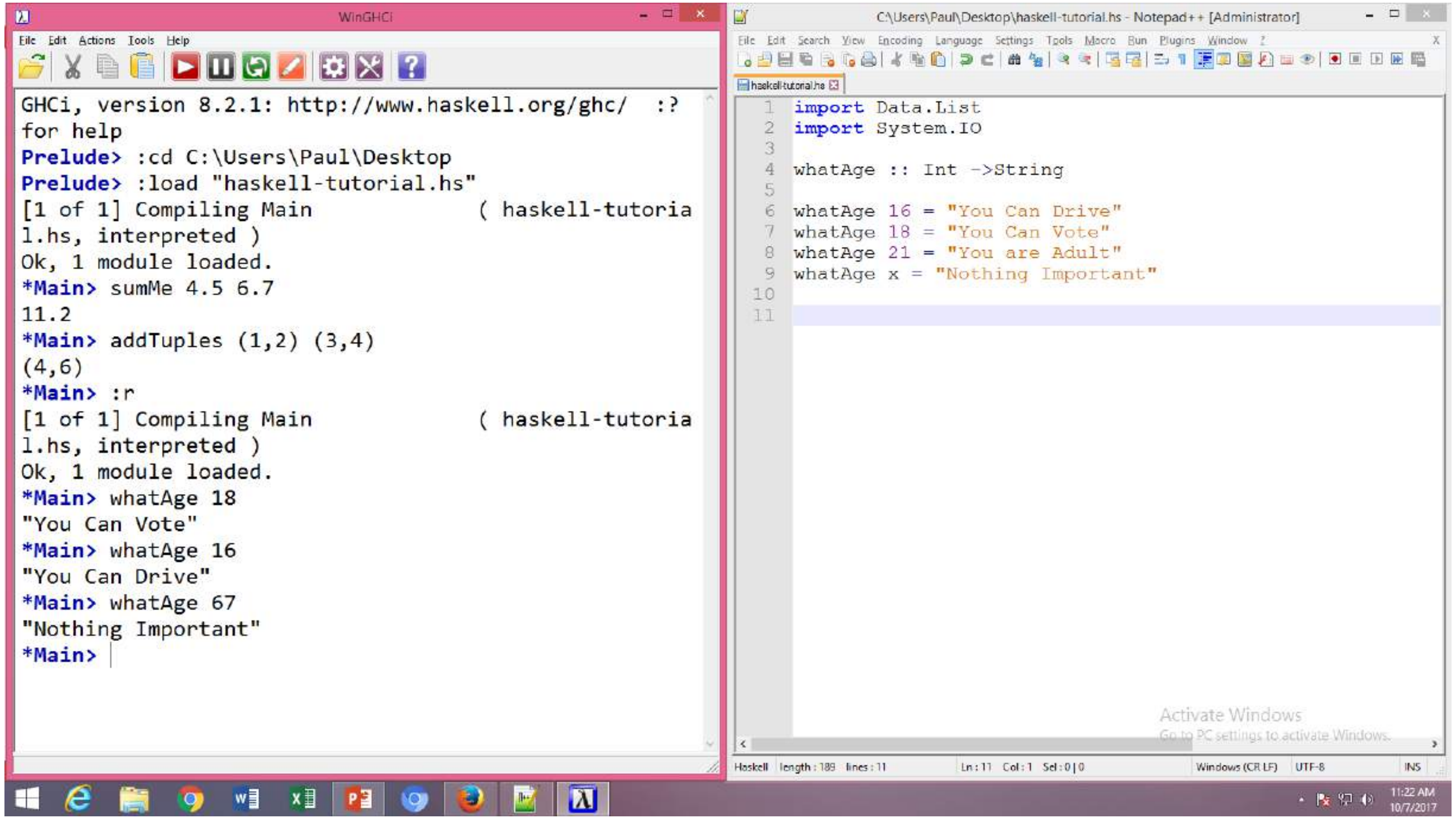
User Type Declaration


WinGHCi
File Edit Actions Tools Help
C:\Users\Paul\Desktop\haskell-tutorial.hs - Notepad++ [Administrator]
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
haskell-tutorial.hs
Activate Windows
Go to PC settings to activate Windows.
Haskell length : 152 lines : 10 Ln : 10 Col : 1 Sel : 0 | 0 Windows (CR LF) UTF-8 INS
11:09 AM
10/7/2017

```
GHCi, version 8.2.1: http://www.haskell.org/ghc/  :?
for help
Prelude> :cd C:\Users\Paul\Desktop
Prelude> :load "haskell-tutorial.hs"
[1 of 1] Compiling Main             ( haskell-tutoria
l.hs, interpreted )
Ok, 1 module loaded.
*Main> sumMe 4.5 6.7
11.2
*Main> addTuples (1,2) (3,4)
(4,6)
*Main>
```

```haskell
import Data.List
import System.IO

sumMe x y = x + y

addTuples :: (Int,Int) -> (Int,Int) -> (Int,Int)

addTuples (x,y) (x2,y2) = (x+x2,y+y2)

```

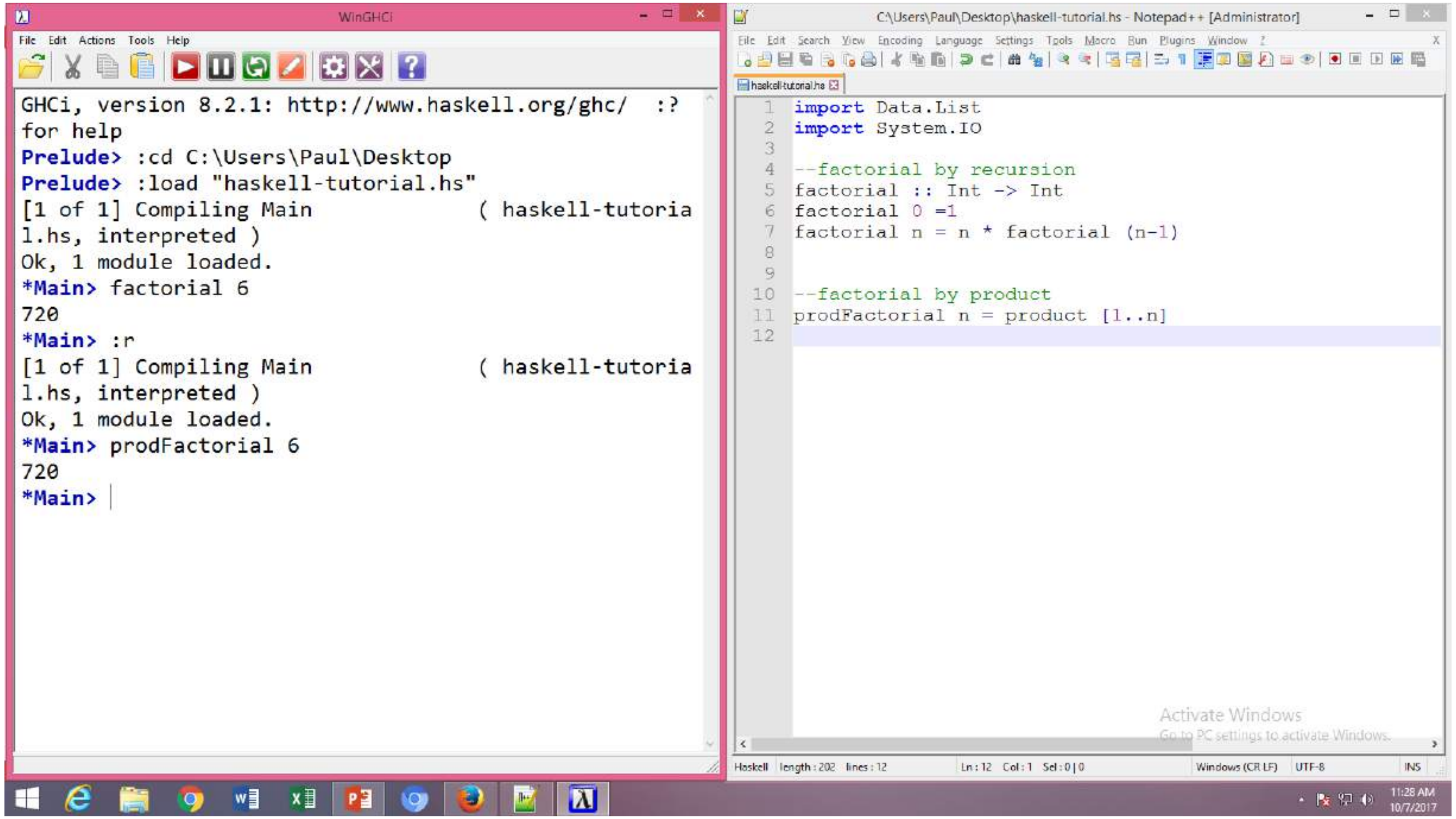
User Type Declaration


WinGHCi
File Edit Actions Tools Help
C:\Users\Paul\Desktop\haskell-tutorial.hs - Notepad++ [Administrator]
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
haskell-tutorial.hs
Haskell length : 189 lines : 11 Ln : 11 Col : 1 Sel : 0 | 0 Windows (CR LF) UTF-8 INS
11:22 AM
10/7/2017

```
GHCi, version 8.2.1: http://www.haskell.org/ghc/  :?
for help
Prelude> :cd C:\Users\Paul\Desktop
Prelude> :load "haskell-tutorial.hs"
[1 of 1] Compiling Main             ( haskell-tutoria
l.hs, interpreted )
Ok, 1 module loaded.
*Main> sumMe 4.5 6.7
11.2
*Main> addTuples (1,2) (3,4)
(4,6)
*Main> :r
[1 of 1] Compiling Main             ( haskell-tutoria
l.hs, interpreted )
Ok, 1 module loaded.
*Main> whatAge 18
"You Can Vote"
*Main> whatAge 16
"You Can Drive"
*Main> whatAge 67
"Nothing Important"
*Main>
```

```haskell
import Data.List
import System.IO

whatAge :: Int ->String

whatAge 16 = "You Can Drive"
whatAge 18 = "You Can Vote"
whatAge 21 = "You are Adult"
whatAge x = "Nothing Important"


```

Factorial (by recursion and by product)

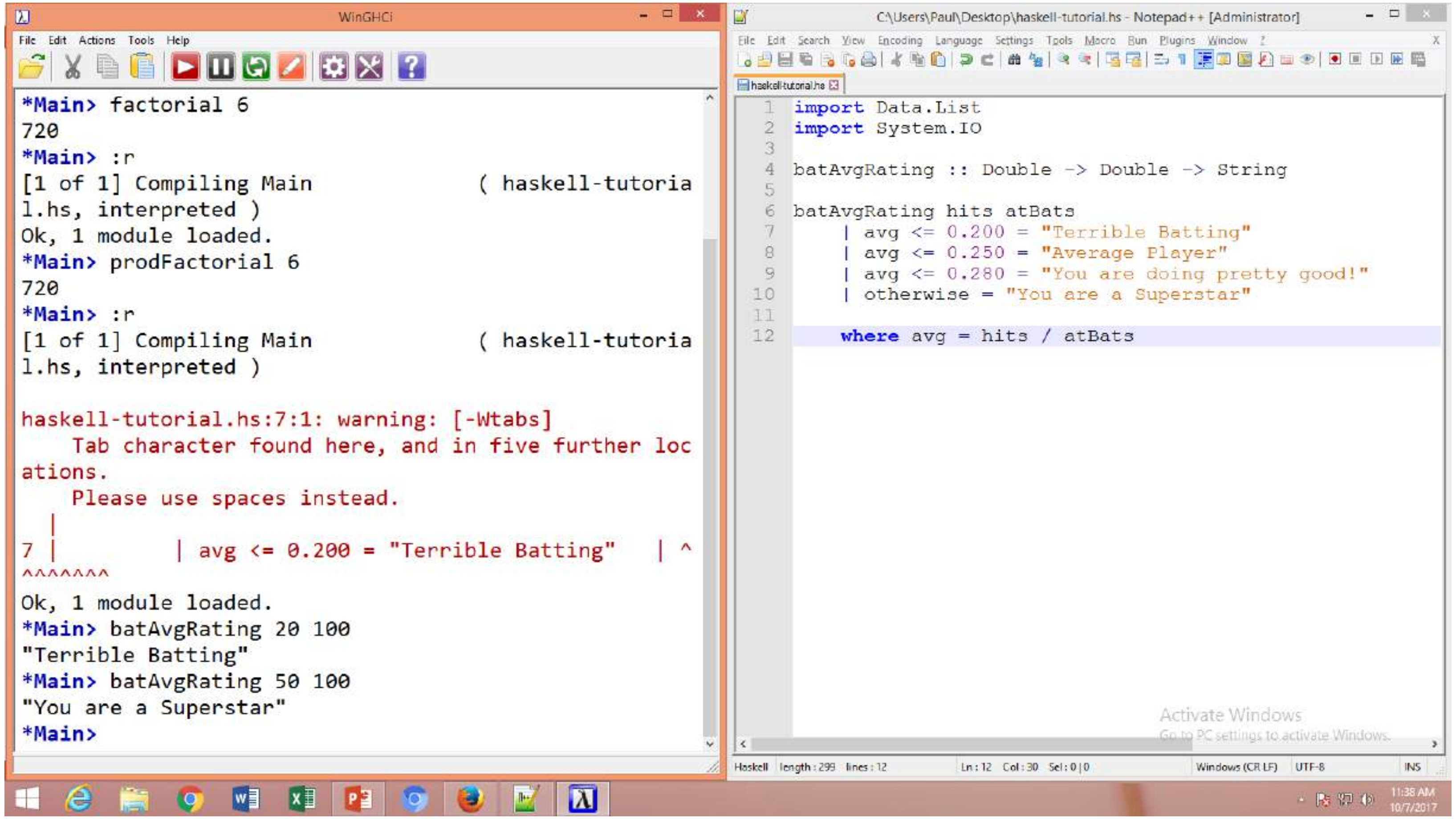


WinGHCi
File Edit Actions Tools Help
C:\Users\Paul\Desktop\haskell-tutorial.hs - Notepad++ [Administrator]
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
haskell-tutorial.hs
Haskell length : 202 lines : 12 Ln : 12 Col : 1 Sel : 0 | 0 Windows (CR LF) UTF-8 INS
Activate Windows
Go to PC settings to activate Windows.
11:28 AM
10/7/2017

```
GHCi, version 8.2.1: http://www.haskell.org/ghc/  :? for help
Prelude> :cd C:\Users\Paul\Desktop
Prelude> :load "haskell-tutorial.hs"
[1 of 1] Compiling Main             ( haskell-tutorial.hs, interpreted )
Ok, 1 module loaded.
*Main> factorial 6
720
*Main> :r
[1 of 1] Compiling Main             ( haskell-tutorial.hs, interpreted )
Ok, 1 module loaded.
*Main> prodFactorial 6
720
*Main>
```

```haskell
import Data.List
import System.IO

--factorial by recursion
factorial :: Int -> Int
factorial 0 =1
factorial n = n * factorial (n-1)


--factorial by product
prodFactorial n = product [1..n]

```

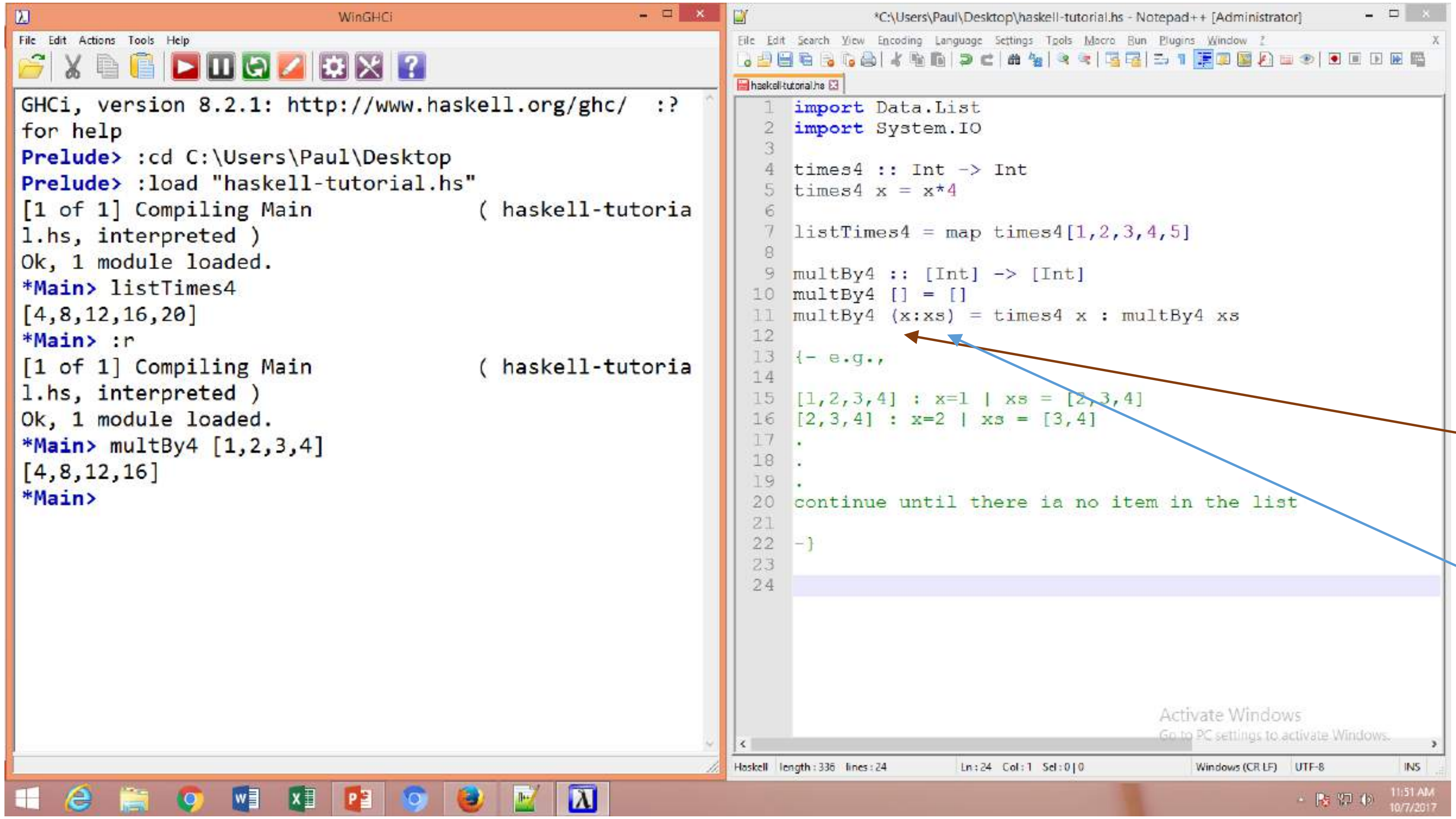
Guard (where clause)


WinGHCi
File Edit Actions Tools Help
C:\Users\Paul\Desktop\haskell-tutorial.hs - Notepad++ [Administrator]
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
haskell-tutorial.hs
Activate Windows
Go to PC settings to activate Windows.
Haskell length : 299 lines : 12 Ln : 12 Col : 30 Sel : 0 | 0 Windows (CR LF) UTF-8 INS
11:38 AM
10/7/2017

```
*Main> factorial 6
720
*Main> :r
[1 of 1] Compiling Main             ( haskell-tutoria
l.hs, interpreted )
Ok, 1 module loaded.
*Main> prodFactorial 6
720
*Main> :r
[1 of 1] Compiling Main             ( haskell-tutoria
l.hs, interpreted )

haskell-tutorial.hs:7:1: warning: [-Wtabs]
    Tab character found here, and in five further loc
ations.
    Please use spaces instead.
  |
7 |         | avg <= 0.200 = "Terrible Batting"    | ^
^^^^^^^
Ok, 1 module loaded.
*Main> batAvgRating 20 100
"Terrible Batting"
*Main> batAvgRating 50 100
"You are a Superstar"
*Main>
```

```haskell
import Data.List
import System.IO

batAvgRating :: Double -> Double -> String

batAvgRating hits atBats
      | avg <= 0.200 = "Terrible Batting"
      | avg <= 0.250 = "Average Player"
      | avg <= 0.280 = "You are doing pretty good!"
      | otherwise = "You are a Superstar"

    where avg = hits / atBats
```

Higher Order Functions

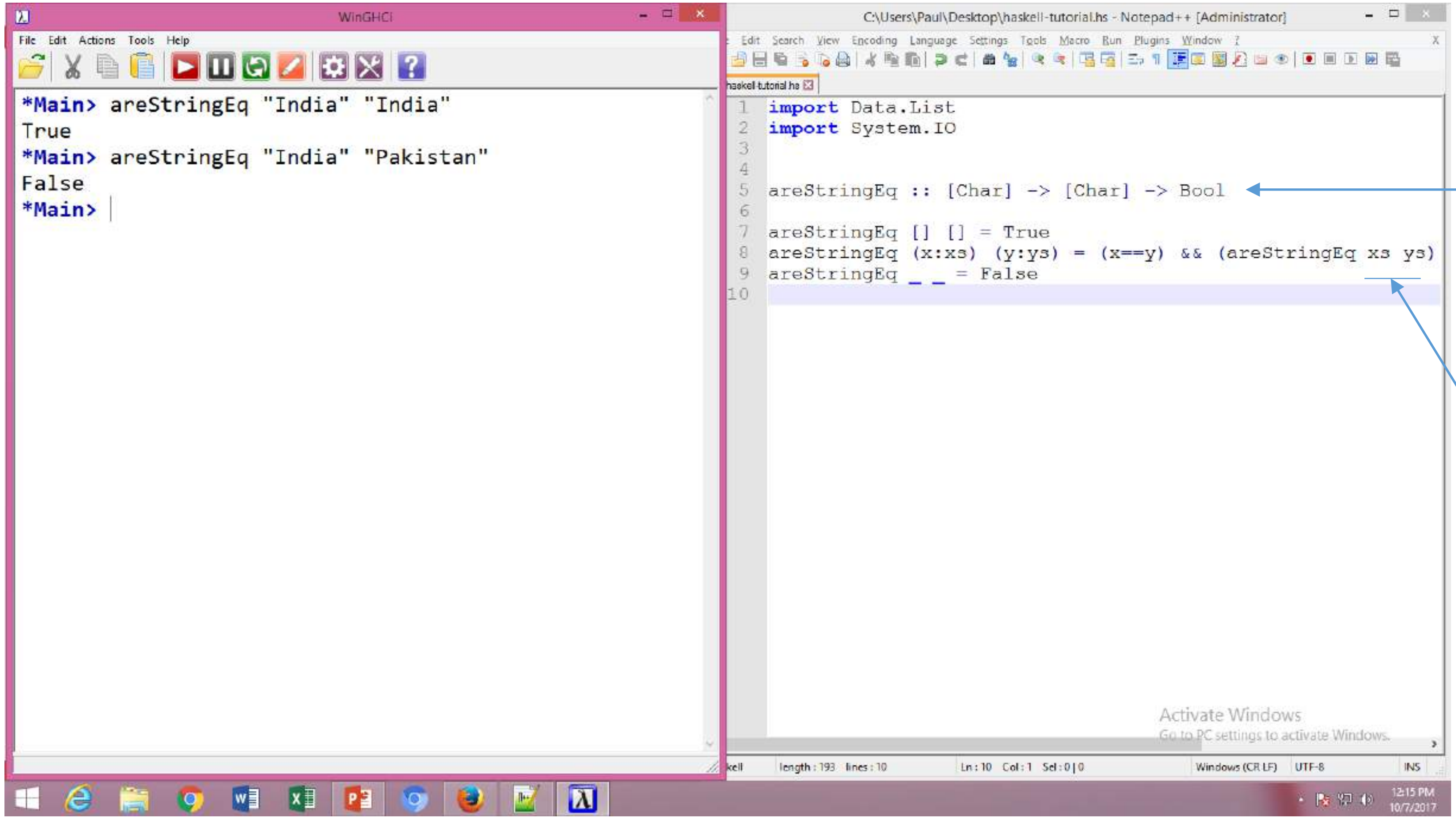

WinGHCi
File Edit Actions Tools Help
*C:\Users\Paul\Desktop\haskell-tutorial.hs - Notepad++ [Administrator]
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
haskell-tutorial.hs
Activate Windows
Go to PC settings to activate Windows.
Haskell length : 336 lines : 24 Ln : 24 Col : 1 Sel : 0 | 0 Windows (CR LF) UTF-8 INS
11:51 AM
10/7/2017

```
GHCi, version 8.2.1: http://www.haskell.org/ghc/  :?
for help
Prelude> :cd C:\Users\Paul\Desktop
Prelude> :load "haskell-tutorial.hs"
[1 of 1] Compiling Main             ( haskell-tutoria
l.hs, interpreted )
Ok, 1 module loaded.
*Main> listTimes4
[4,8,12,16,20]
*Main> :r
[1 of 1] Compiling Main             ( haskell-tutoria
l.hs, interpreted )
Ok, 1 module loaded.
*Main> multBy4 [1,2,3,4]
[4,8,12,16]
*Main>
```

```haskell
import Data.List
import System.IO

times4 :: Int -> Int
times4 x = x*4

listTimes4 = map times4[1,2,3,4,5]

multBy4 :: [Int] -> [Int]
multBy4 [] = []
multBy4 (x:xs) = times4 x : multBy4 xs

{- e.g.,

[1,2,3,4] : x=1 | xs = [2,3,4]
[2,3,4] : x=2 | xs = [3,4]
.
.
.
continue until there ia no item in the list

-}

```

you don't know how many items in the list Beforehand;
x represents first element in the list, and
xs represents remaining elements of the list

Higher Order Functions



WinGHCi
File Edit Actions Tools Help
C:\Users\Paul\Desktop\haskell-tutorial.hs - Notepad++ [Administrator]
haskell-tutorial.hs
Activate Windows
Go to PC settings to activate Windows.

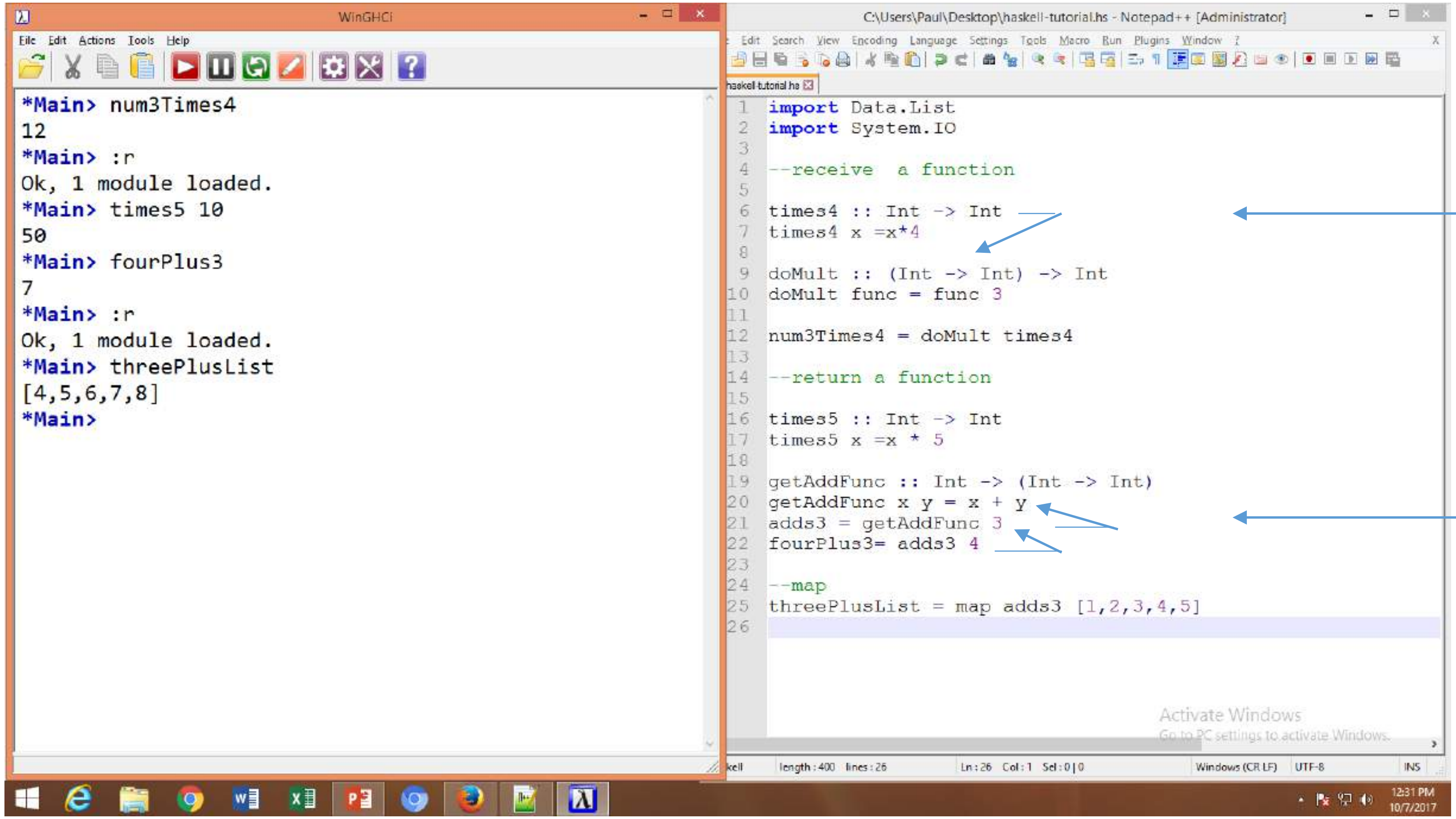
```
*Main> areStringEq "India" "India"
True
*Main> areStringEq "India" "Pakistan"
False
*Main>
```

```haskell
import Data.List
import System.IO


areStringEq :: [Char] -> [Char] -> Bool

areStringEq [] [] = True
areStringEq (x:xs) (y:ys) = (x==y) && (areStringEq xs ys)
areStringEq _ _ = False
```

return type

remaining

Receive and Return a Function


WinGHCi
File Edit Actions Tools Help
C:\Users\Paul\Desktop\haskell-tutorial.hs - Notepad++ [Administrator]
Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
haskell-tutorial.hs
Activate Windows
Go to PC settings to activate Windows.
length : 400 lines : 26 Ln : 26 Col : 1 Sel : 0 | 0 Windows (CR LF) UTF-8 INS
12:31 PM
10/7/2017

```
*Main> num3Times4
12
*Main> :r
Ok, 1 module loaded.
*Main> times5 10
50
*Main> fourPlus3
7
*Main> :r
Ok, 1 module loaded.
*Main> threePlusList
[4,5,6,7,8]
*Main>
```

```haskell
import Data.List
import System.IO

--receive  a function

times4 :: Int -> Int
times4 x =x*4

doMult :: (Int -> Int) -> Int
doMult func = func 3

num3Times4 = doMult times4

--return a function

times5 :: Int -> Int
times5 x =x * 5

getAddFunc :: Int -> (Int -> Int)
getAddFunc x y = x + y
adds3 = getAddFunc 3
fourPlus3= adds3 4

--map
threePlusList = map adds3 [1,2,3,4,5]
```

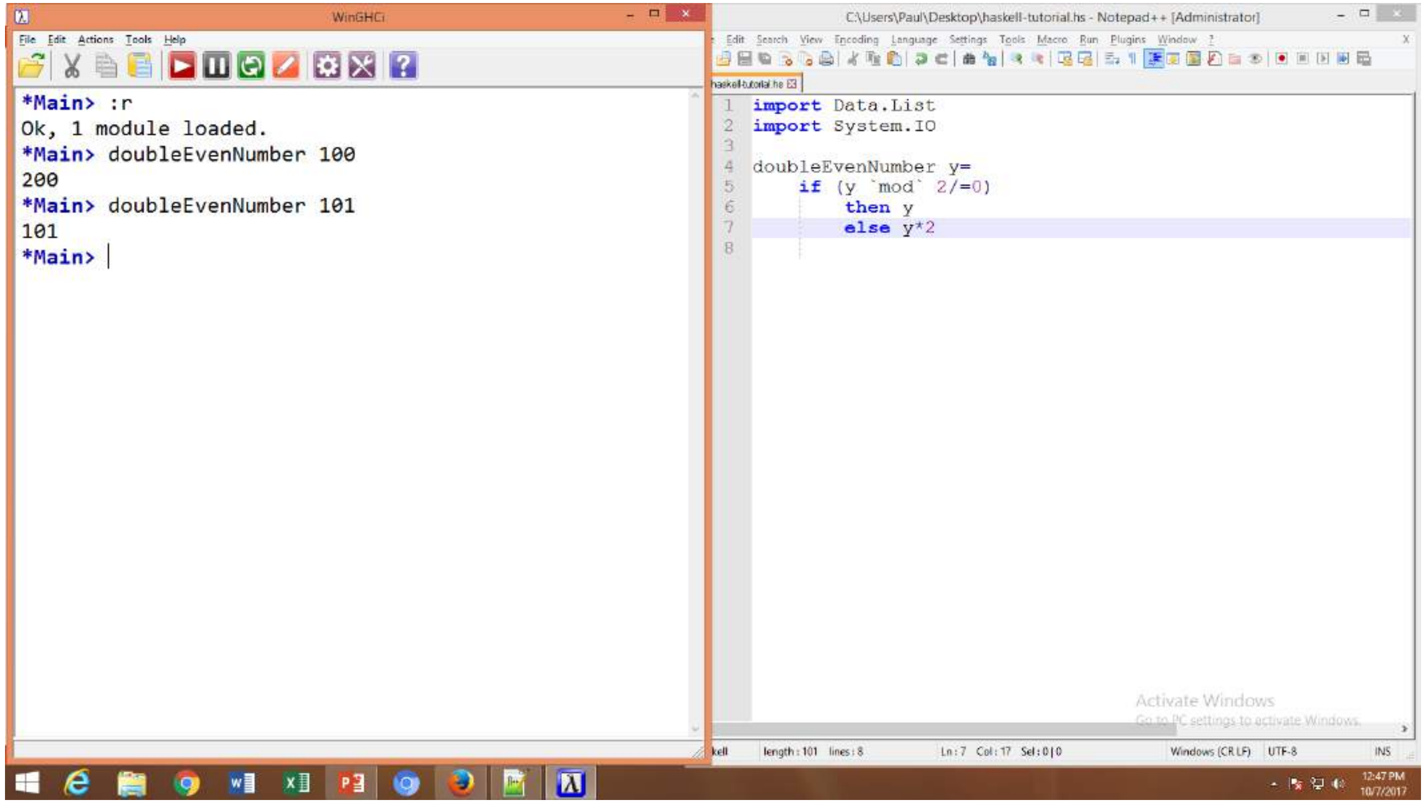
receive

return

Other Operators

- Comparison
  - < --less than
  - > --greater than
  - <= --less than equal to
  - >= --greater than equal to
  - == --equal to
- Logical
  - && --AND
  - || --OR
  - not --NOT

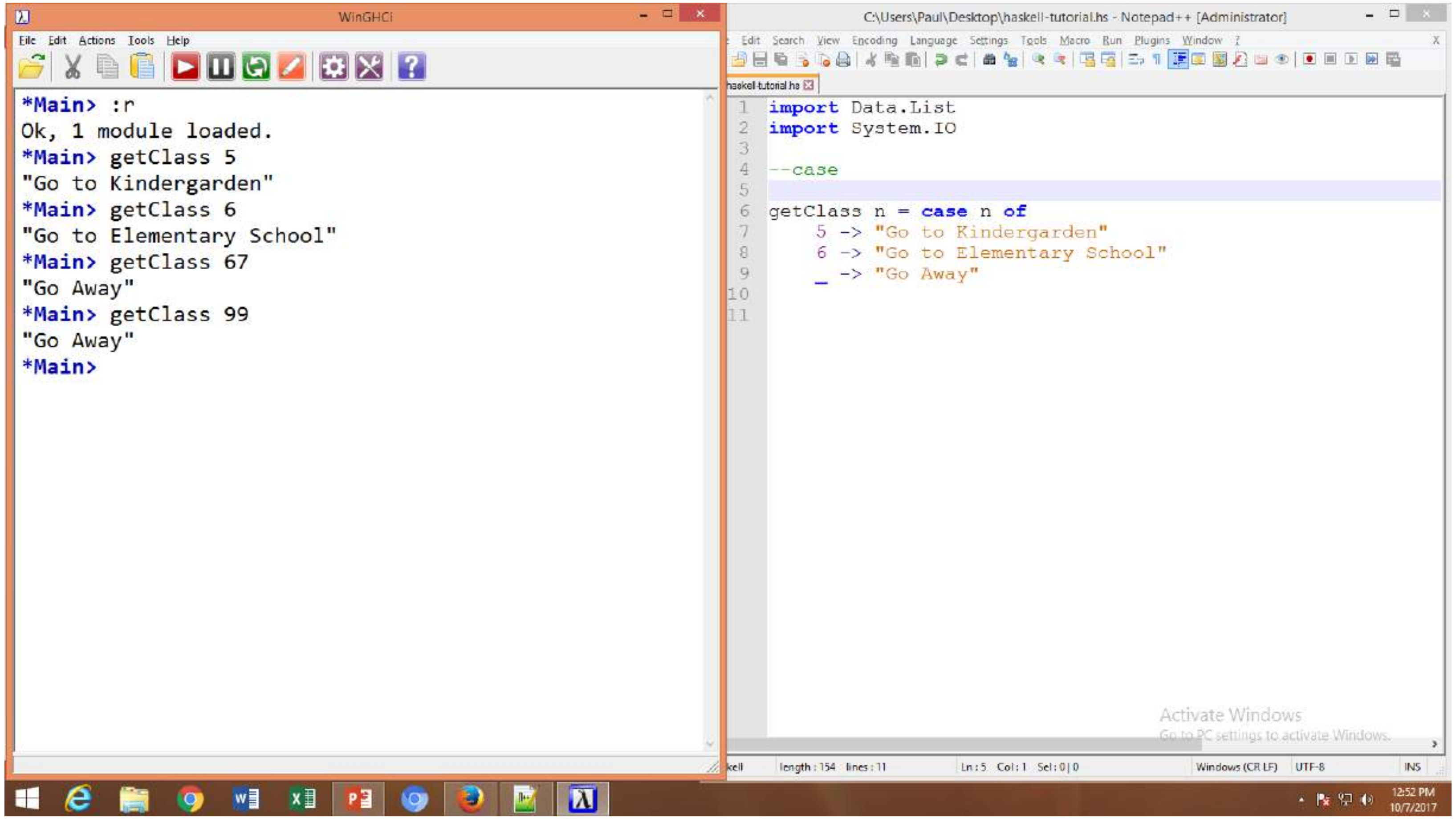
Example


WinGHCi
File Edit Actions Tools Help
C:\Users\Paul\Desktop\haskell-tutorial.hs - Notepad++ [Administrator]
Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
haskell-tutorial.hs

```
*Main> :r
Ok, 1 module loaded.
*Main> doubleEvenNumber 100
200
*Main> doubleEvenNumber 101
101
*Main> 
```

```haskell
import Data.List
import System.IO

doubleEvenNumber y=
    if (y `mod` 2/=0)
        then y
        else y*2
```

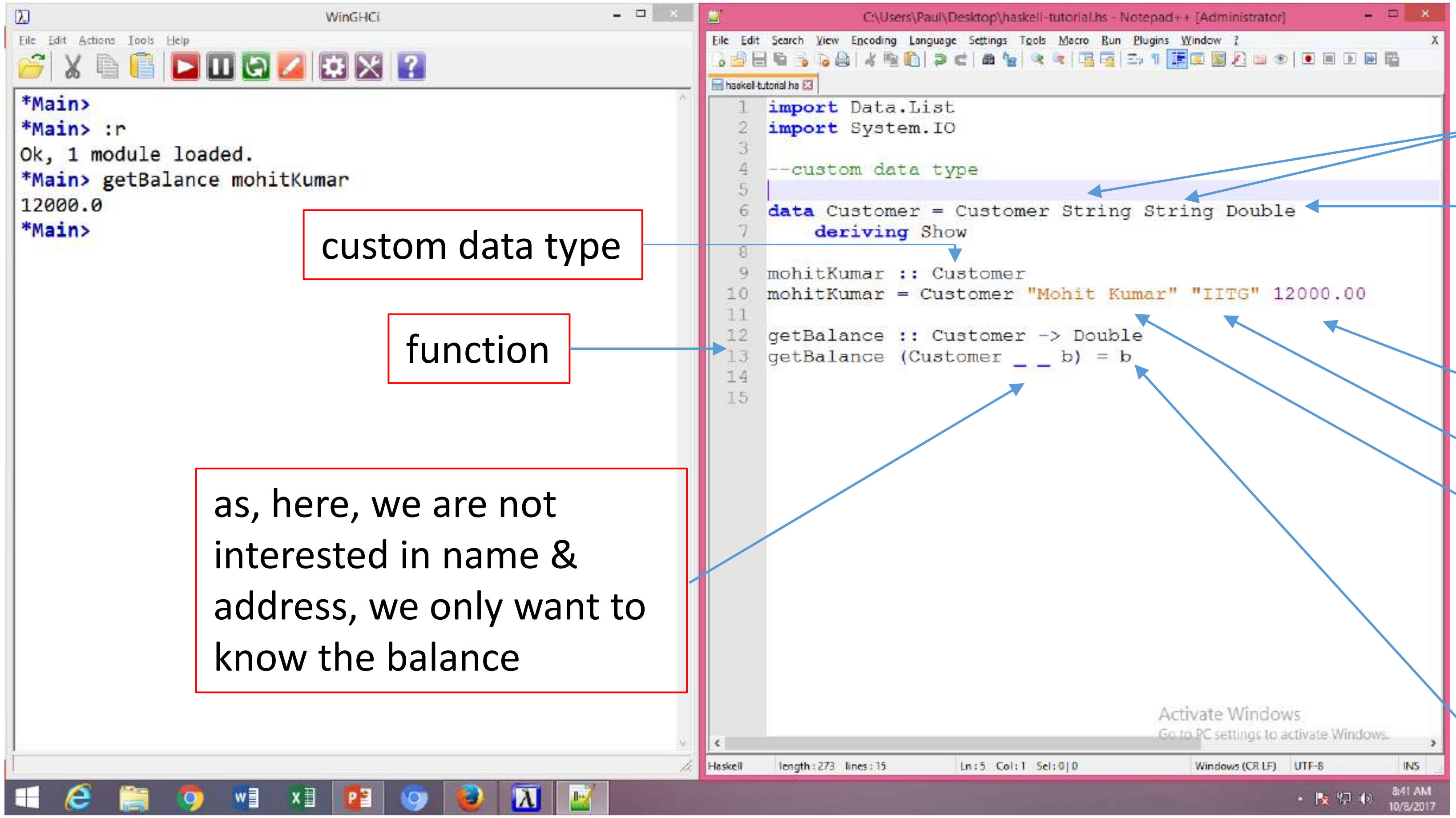
Case


WinGHCi
File Edit Actions Tools Help
C:\Users\Paul\Desktop\haskell-tutorial.hs - Notepad++ [Administrator]
Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
haskell-tutorial.hs
length : 154 lines : 11
Ln : 5 Col : 1 Sel : 0 | 0
Windows (CR LF) UTF-8 INS
Activate Windows
Go to PC settings to activate Windows.
12:52 PM
10/7/2017

```
*Main> :r
Ok, 1 module loaded.
*Main> getClass 5
"Go to Kindergarden"
*Main> getClass 6
"Go to Elementary School"
*Main> getClass 67
"Go Away"
*Main> getClass 99
"Go Away"
*Main>
```

```haskell
import Data.List
import System.IO

--case

getClass n = case n of
    5 -> "Go to Kindergarden"
    6 -> "Go to Elementary School"
    _ -> "Go Away"
```

Custom Data Type



WinGHCi
File Edit Actions Tools Help
C:\Users\Paul\Desktop\haskell-tutorial.hs - Notepad++ [Administrator]
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
haskell-tutorial.hs
Activate Windows
Go to PC settings to activate Windows.
Haskell length : 273 lines : 15 Ln : 5 Col : 1 Sel : 0 | 0 Windows (CR LF) UTF-8 INS
8:41 AM
10/8/2017

```
*Main>
*Main> :r
Ok, 1 module loaded.
*Main> getBalance mohitKumar
12000.0
*Main>
```

```haskell
import Data.List
import System.IO

--custom data type

data Customer = Customer String String Double
    deriving Show

mohitKumar :: Customer
mohitKumar = Customer "Mohit Kumar" "IITG" 12000.00

getBalance :: Customer -> Double
getBalance (Customer _ _ b) = b
```

two string

one double

custom data type

function

balance

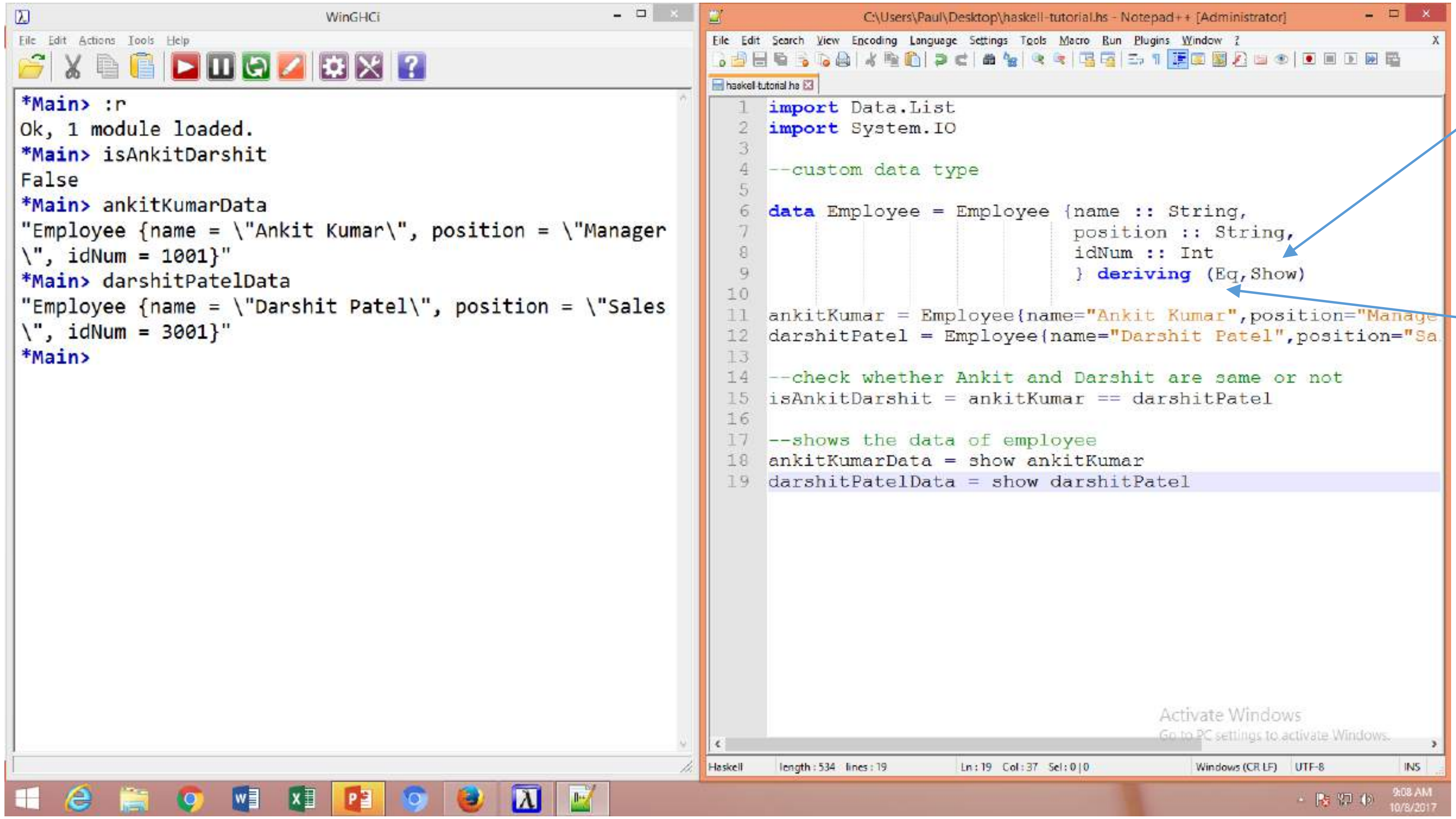
address

name

as, here, we are not interested in name & address, we only want to know the balance

b represents balance

Type Classes



WinGHCi
File Edit Actions Tools Help
C:\Users\Paul\Desktop\haskell-tutorial.hs - Notepad++ [Administrator]
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
haskell-tutorial.hs
Activate Windows
Go to PC settings to activate Windows.
Haskell length : 534 lines : 19 Ln : 19 Col : 37 Sel : 0 | 0 Windows (CR LF) UTF-8 INS
9:08 AM
10/8/2017

```
*Main> :r
Ok, 1 module loaded.
*Main> isAnkitDarshit
False
*Main> ankitKumarData
"Employee {name = \"Ankit Kumar\", position = \"Manager
\", idNum = 1001}"
*Main> darshitPatelData
"Employee {name = \"Darshit Patel\", position = \"Sales
\", idNum = 3001}"
*Main>
```

```
import Data.List
import System.IO

--custom data type

data Employee = Employee {name :: String,
                          position :: String,
                          idNum :: Int
                          } deriving (Eq,Show)

ankitKumar = Employee{name="Ankit Kumar",position="Manage
darshitPatel = Employee{name="Darshit Patel",position="Sa

--check whether Ankit and Darshit are same or not
isAnkitDarshit = ankitKumar == darshitPatel

--shows the data of employee
ankitKumarData = show ankitKumar
darshitPatelData = show darshitPatel
```

able to show the employee details

able to check for the equality

END OF TUTORIAL

YOU MAY EXPLORE

http://www.learnyouahaskell.com

FOR MORE DETAIL