

Functional Programming Concept with



Tutorial for

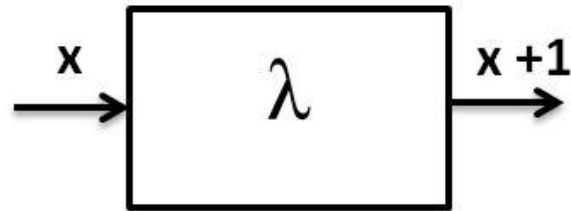
Programming Languages Laboratory (CS 431)

July – November 2018



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

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

Haskell - Features

➤ 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

- **facebook** → anti-spam programs

-  → a window manager for the X Window System
xmonad

-  **darcs** → revision control system

Some other FP Applications

-  → Scala
-  → 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-tutorial.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

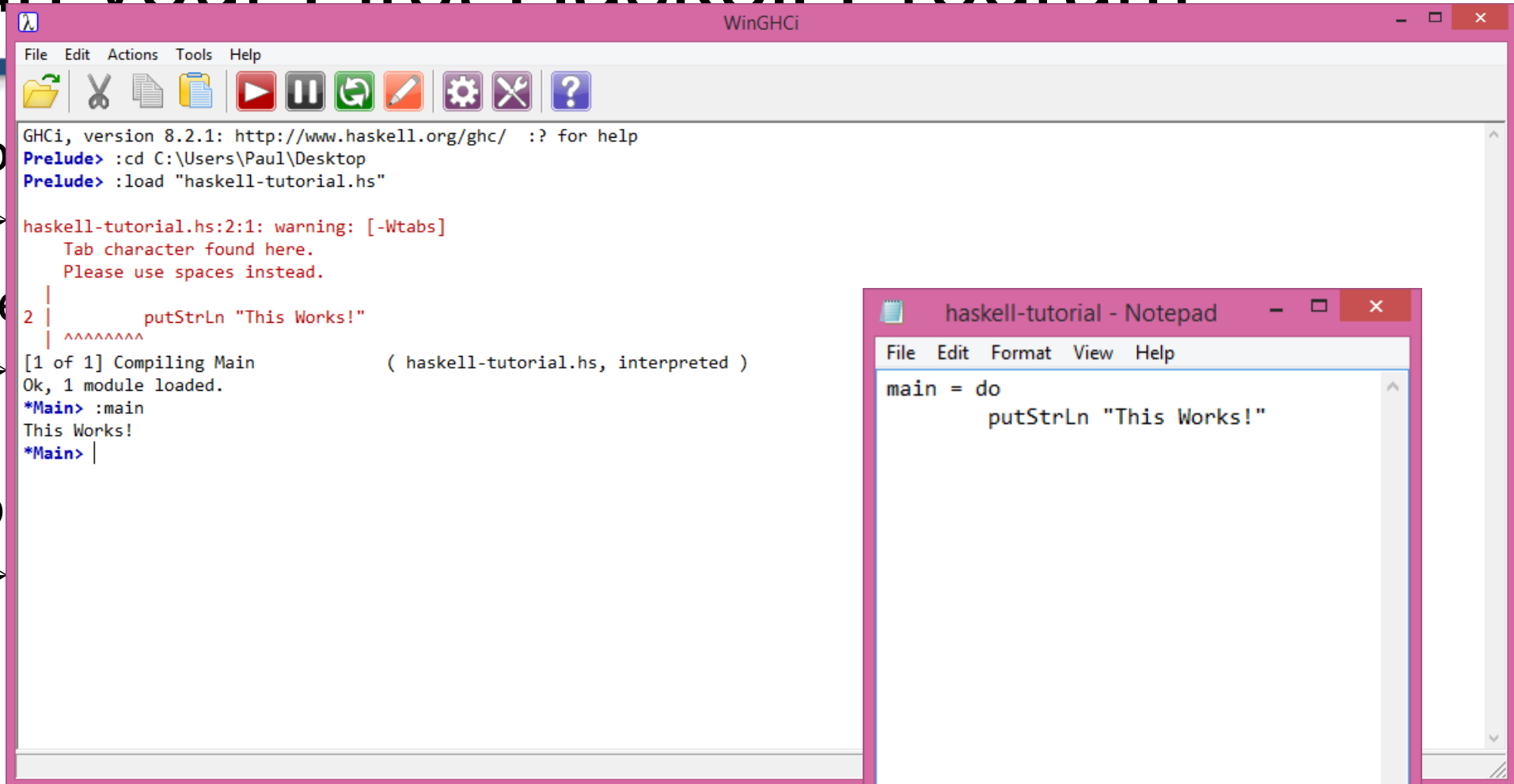
➤ Do



➤ File



➤ Co



The image shows two overlapping windows. The top window is titled 'WinGHCi' and contains the following text:

```
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> |
```

The bottom window is titled 'haskell-tutorial - Notepad' and contains the following code:

```
File Edit Format View Help
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-tutorial.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

➤ Download

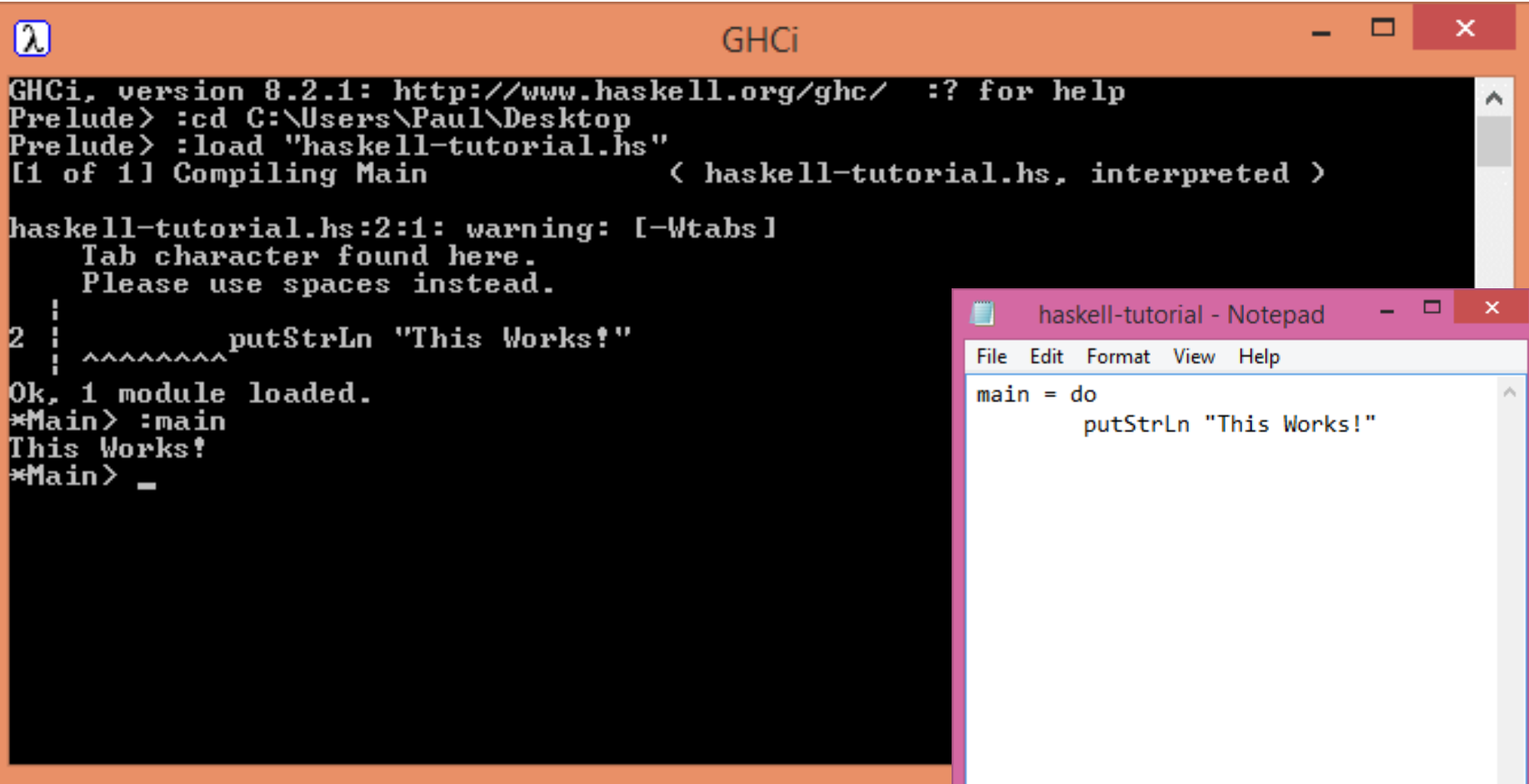
➤

➤ File

➤

➤ Compiler

➤



The image shows two windows. The top window is a Haskell GHCi terminal with an orange title bar. It displays the following text:

```
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> _
```

The bottom window is a Notepad editor with a pink title bar, titled "haskell-tutorial - Notepad". It shows the source code of the Haskell program:

```
File Edit Format View Help
main = do
    putStrLn "This Works!"
```

➤ Run the function you want

ion

p)

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
 - `always5 :: Int`
`always5 = 5` } permanent value for a variable
--Never Change

Expressions



```
WinGHCi
File Edit Actions Tools Help
[Icons]
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> |

C:\Users\Paul\Desktop\haskell-tutorial.hs - Notepad++ [Administrator]
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
[Icons]
haskell-tutorial.hs
1  import Data.List
2  import System.IO
3
4  sumOfNums = sum[1..100]
5
6  addExpr = 5+3
7  subExpr = 5-3
8  multExpr = 5*3
9  divExpr = 5/3
10
11 nextExpr = multExpr/subExpr
12
13
14
15

Haskell length: 163 lines: 15 Ln: 13 Col: 1 Sel: 0|0 Windows (CR LF) UTF-8 INS
12:37 AM 10/7/2017
```

Expressions

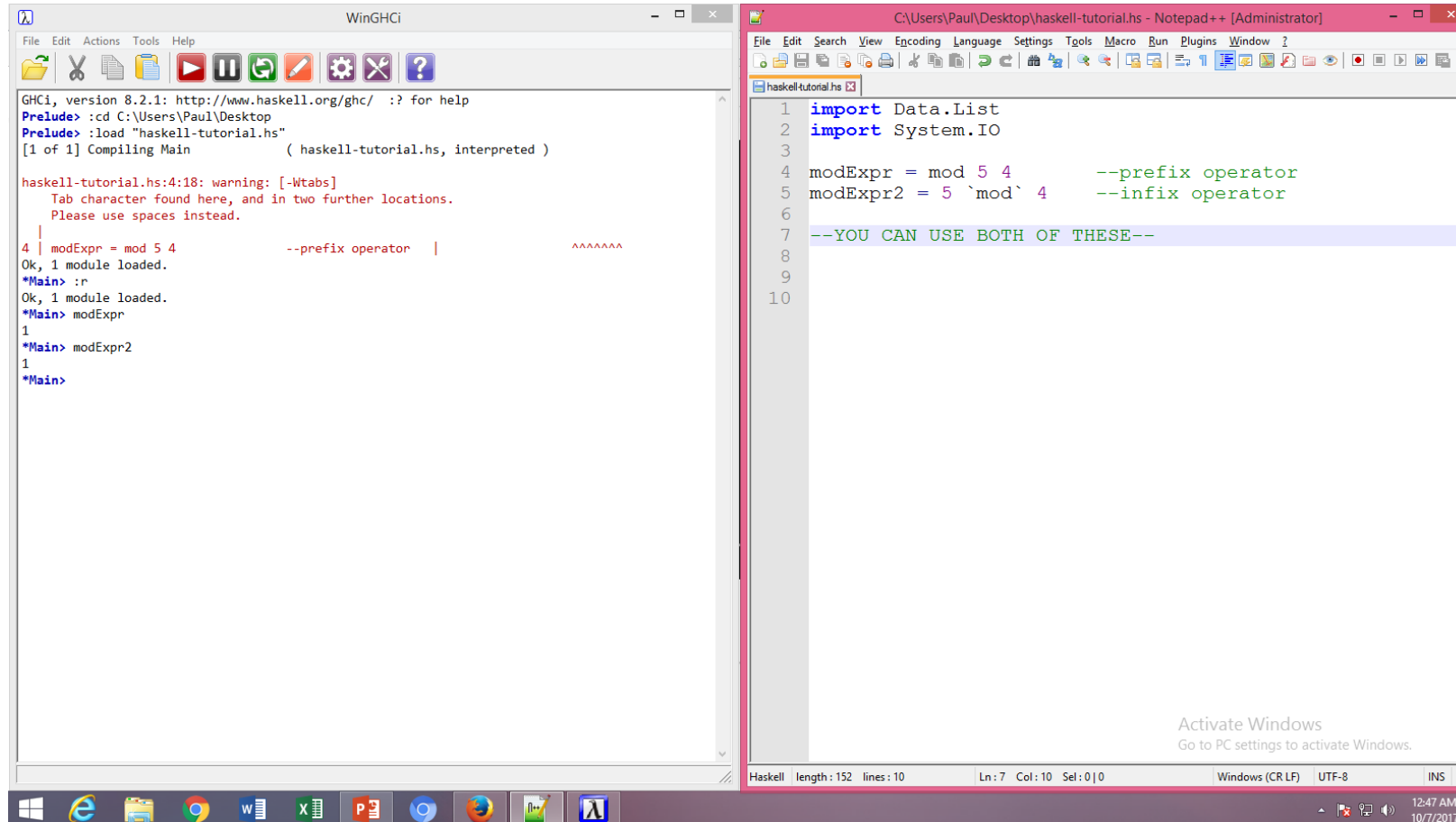


```
WinGHCi
File Edit Actions Tools Help
[Icons]
GHCi, version 8.2.1: http://www.haskell.org/ghc/  :? for help
Prelude> :cd C:\Users\Paul\Desktop
Prelude> :load "haskell-tutorial.hs"
[1 of 1] Compiling Main          ( haskell-tutorial.hs, interpreted )
Ok, 1 module loaded.
*Main> addExpr
8
*Main> multExpr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

C:\Users\Paul\Desktop\haskell-tutorial.hs - Notepad++ [Administrator]
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
[Icons]
haskell-tutorial.hs
1  import Data.List
2  import System.IO
3
4  sumOfNums = sum[1..100]
5
6  addExpr = 5+3
7  subExpr = 5-3
8  multExpr = 5*3
9  divExpr = 5/3
10
11 nextExpr = multExpr/subExpr
12
13
14
15

Haskell length: 163 lines: 15 Ln: 13 Col: 1 Sel: 0|0 Windows (CR LF) UTF-8 INS
12:39 AM 10/7/2017
```

Infix and Prefix Operator



The image shows two windows side-by-side. The left window is WinGHCi, and the right window is Notepad++.

WinGHCi window:

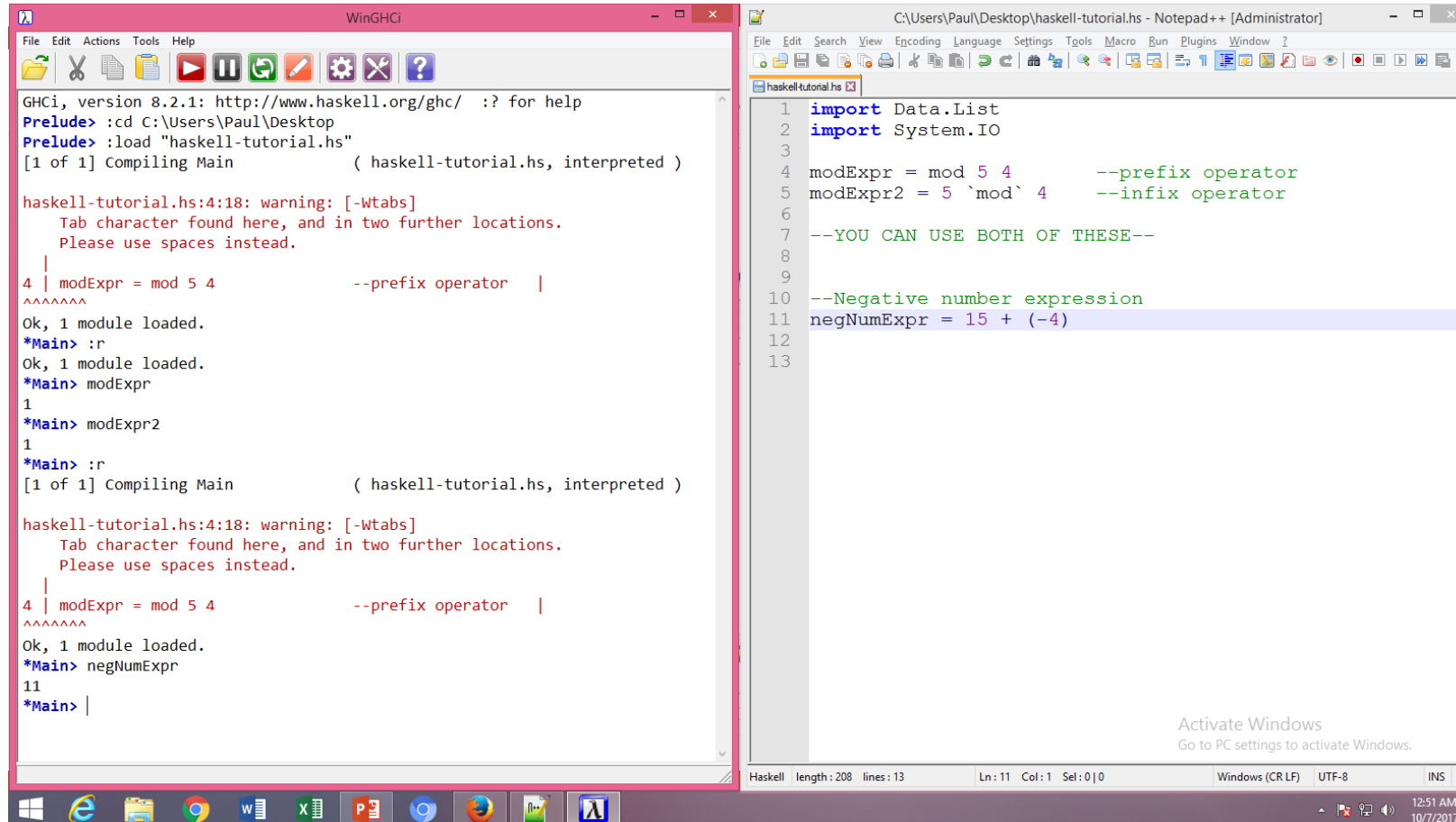
```
WinGHCi
File Edit Actions Tools Help
[Icons]
GHCi, version 8.2.1: http://www.haskell.org/ghc/  :? for help
Prelude> :cd C:\Users\Paul\Desktop
Prelude> :load "haskell-tutorial.hs"
[1 of 1] Compiling Main                ( haskell-tutorial.hs, interpreted )
haskell-tutorial.hs:4:18: warning: [-Wtabs]
    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>
```

Notepad++ window:

```
C:\Users\Paul\Desktop\haskell-tutorial.hs - Notepad++ [Administrator]
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
haskell-tutorial.hs
1 import Data.List
2 import System.IO
3
4 modExpr = mod 5 4          --prefix operator
5 modExpr2 = 5 `mod` 4      --infix operator
6
7 --YOU CAN USE BOTH OF THESE--
8
9
10
```

Windows taskbar at the bottom shows the time as 12:47 AM on 10/7/2017.

Negative Number Expression



The screenshot displays a Haskell development environment. On the left, the WinGHCi terminal window shows the execution of a Haskell program. The user has loaded 'haskell-tutorial.hs' and compiled it. The program defines two modulo expressions and a negative number expression. The terminal output shows the successful loading of the module and the evaluation of the expressions. On the right, the Notepad++ editor window shows the source code of 'haskell-tutorial.hs'. The code imports 'Data.List' and 'System.IO', defines 'modExpr' and 'modExpr2' using modulo operators, and defines 'negNumExpr' as '15 + (-4)'. The status bar at the bottom indicates the file is 'haskell-tutorial.hs' with a length of 208 bytes and 13 lines.

```
WinGHCi
File Edit Actions Tools Help
[Icons]
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>

C:\Users\Paul\Desktop\haskell-tutorial.hs - Notepad++ [Administrator]
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
haskell-tutorial.hs
1 import Data.List
2 import System.IO
3
4 modExpr = mod 5 4          --prefix operator
5 modExpr2 = 5 `mod` 4      --infix operator
6
7 --YOU CAN USE BOTH OF THESE--
8
9
10 --Negative number expression
11 negNumExpr = 15 + (-4)
12
13

Haskell length: 208 lines: 13 Ln: 11 Col: 1 Sel: 0|0 Windows (CR LF) UTF-8 INS
12:51 AM 10/7/2017
```

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, asin, acos, atan, sinh, cosh, tanh, asinh, acosh, atanh`

EXPLORE THESE

List - Concatenation



The image shows a Windows desktop with two windows open. The left window is WinGHCi, a Haskell interpreter, showing the execution of a Haskell script. The right window is Notepad++, showing the source code of the script. The code defines a list 'primeNumber' and then concatenates it with another list '[13,17,19,23,29]' to create 'morePrime'. Annotations with arrows point to the list definition and the concatenation operation.

```
WinGHCi
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> primeNumber
[3,5,7,11]
*Main> :r
[1 of 1] Compiling Main             ( haskell-tutorial.hs, interpreted )
Ok, 1 module loaded.
*Main> morePrime
[3,5,7,11,13,17,19,23,29]
*Main>

C:\Users\Paul\Desktop\haskell-tutorial.hs - Notepad++ [Administrator]
1  import Data.List
2  import System.IO
3
4  primeNumber =[3,5,7,11]
5
6  --Concatenation
7  morePrime = primeNumber ++ [13,17,19,23,29]
8
```

Define a list

Concatenation
of two lists

List – 'cons' operator

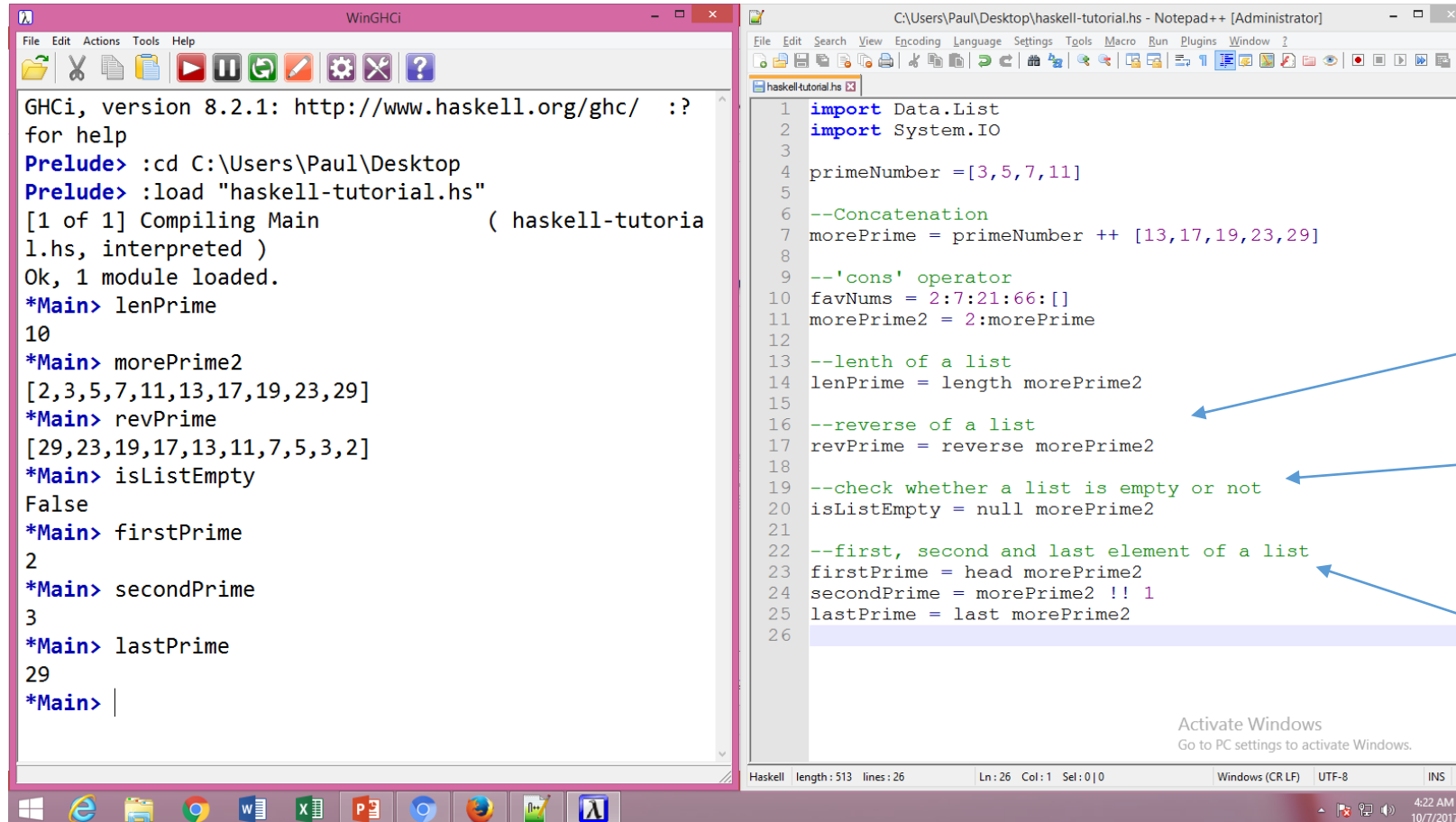


```
WinGHCi
GHCi, version 8.2.1: http://www.haskell.org/ghc/  :? for help
Prelude> :cd C:\Users\Paul\Desktop
Prelude> :load "haskell-tutorial.hs"
[1 of 1] Compiling Main             ( haskell-tutorial.hs, interpreted )
Ok, 1 module loaded.
*Main> primeNumber
[3,5,7,11]
*Main> :r
[1 of 1] Compiling Main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

C:\Users\Paul\Desktop\haskell-tutorial.hs - Notepad++ [Administrator]
1  import Data.List
2  import System.IO
3
4  primeNumber =[3,5,7,11]
5
6  --Concatenation
7  morePrime = primeNumber ++ [13,17,19,23,29]
8
9  --'cons' operator
10 favNums = 2:7:21:66:[]
11 morePrime2 = 2:morePrime
12
13
14

Haskell length: 202 lines: 14 Ln: 13 Col: 1 Sel: 0 | 0 Windows (CR LF) UTF-8 INS
4:08 AM 10/7/2017
```

More Operations on List



The image shows two windows side-by-side. The left window is WinGHCi, and the right window is Notepad++.

WinGHCi window:

```
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> 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> isEmpty
False
*Main> firstPrime
2
*Main> secondPrime
3
*Main> lastPrime
29
*Main> |
```

Notepad++ window:

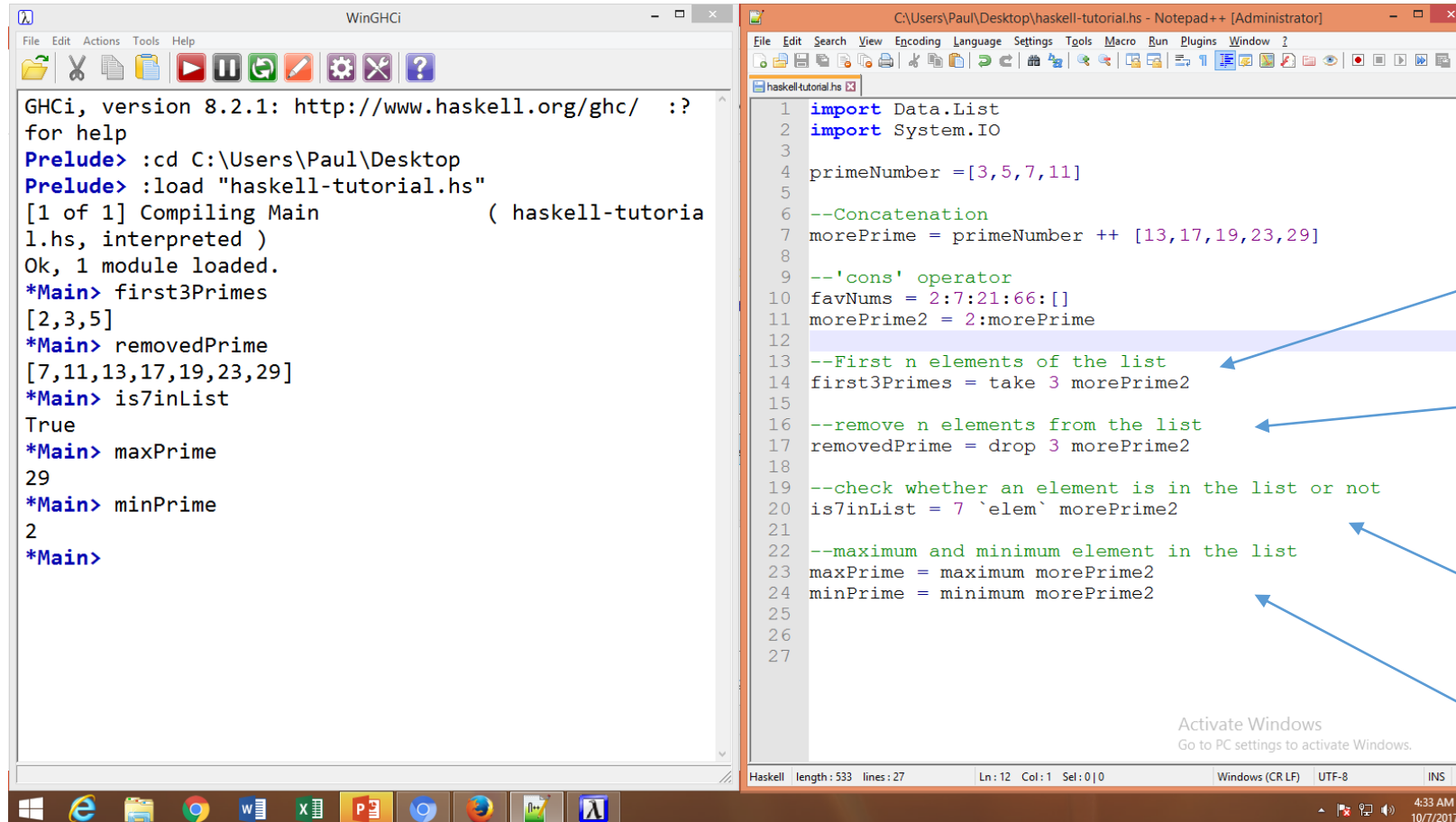
```
1 import Data.List
2 import System.IO
3
4 primeNumber =[3,5,7,11]
5
6 --Concatenation
7 morePrime = primeNumber ++ [13,17,19,23,29]
8
9 --'cons' operator
10 favNums = 2:7:21:66:[]
11 morePrime2 = 2:morePrime
12
13 --length of a list
14 lenPrime = length morePrime2
15
16 --reverse of a list
17 revPrime = reverse morePrime2
18
19 --check whether a list is empty or not
20 isEmpty = null morePrime2
21
22 --first, second and last element of a list
23 firstPrime = head morePrime2
24 secondPrime = morePrime2 !! 1
25 lastPrime = last morePrime2
26
```

reverse

List empty?

particular element

More Operations on List



The image shows two windows side-by-side. The left window is WinGHCi, a Haskell interpreter, showing the execution of a Haskell program. The right window is Notepad++, showing the source code of the program. The code defines a list of prime numbers and performs various operations on it, including concatenation, taking the first n elements, dropping the first n elements, checking if an element is in the list, and finding the maximum and minimum elements.

```
WinGHCi
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> first3Primes
[2,3,5]
*Main> removedPrime
[7,11,13,17,19,23,29]
*Main> is7inList
True
*Main> maxPrime
29
*Main> minPrime
2
*Main>
```

```
C:\Users\Paul\Desktop\haskell-tutorial.hs - Notepad++ [Administrator]
1 import Data.List
2 import System.IO
3
4 primeNumber =[3,5,7,11]
5
6 --Concatenation
7 morePrime = primeNumber ++ [13,17,19,23,29]
8
9 --'cons' operator
10 favNums = 2:7:21:66:[]
11 morePrime2 = 2:morePrime
12
13 --First n elements of the list
14 first3Primes = take 3 morePrime2
15
16 --remove n elements from the list
17 removedPrime = drop 3 morePrime2
18
19 --check whether an element is in the list or not
20 is7inList = 7 `elem` morePrime2
21
22 --maximum and minimum element in the list
23 maxPrime = maximum morePrime2
24 minPrime = minimum morePrime2
25
26
27
```

first n element

removing first
n element

finding an
element

max and min

More Operations on List



The screenshot shows a Windows desktop with two windows. The left window is WinGHCi, and the right window is Notepad++.

WinGHCi window:

```
WinGHCi
File Edit Actions Tools Help
[Icons]
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> prodList
60
*Main> :r
[1 of 1] Compiling Main             ( haskell-tutorial.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
"ACEGIKMQSUWY"
*Main> |
```

Notepad++ window:

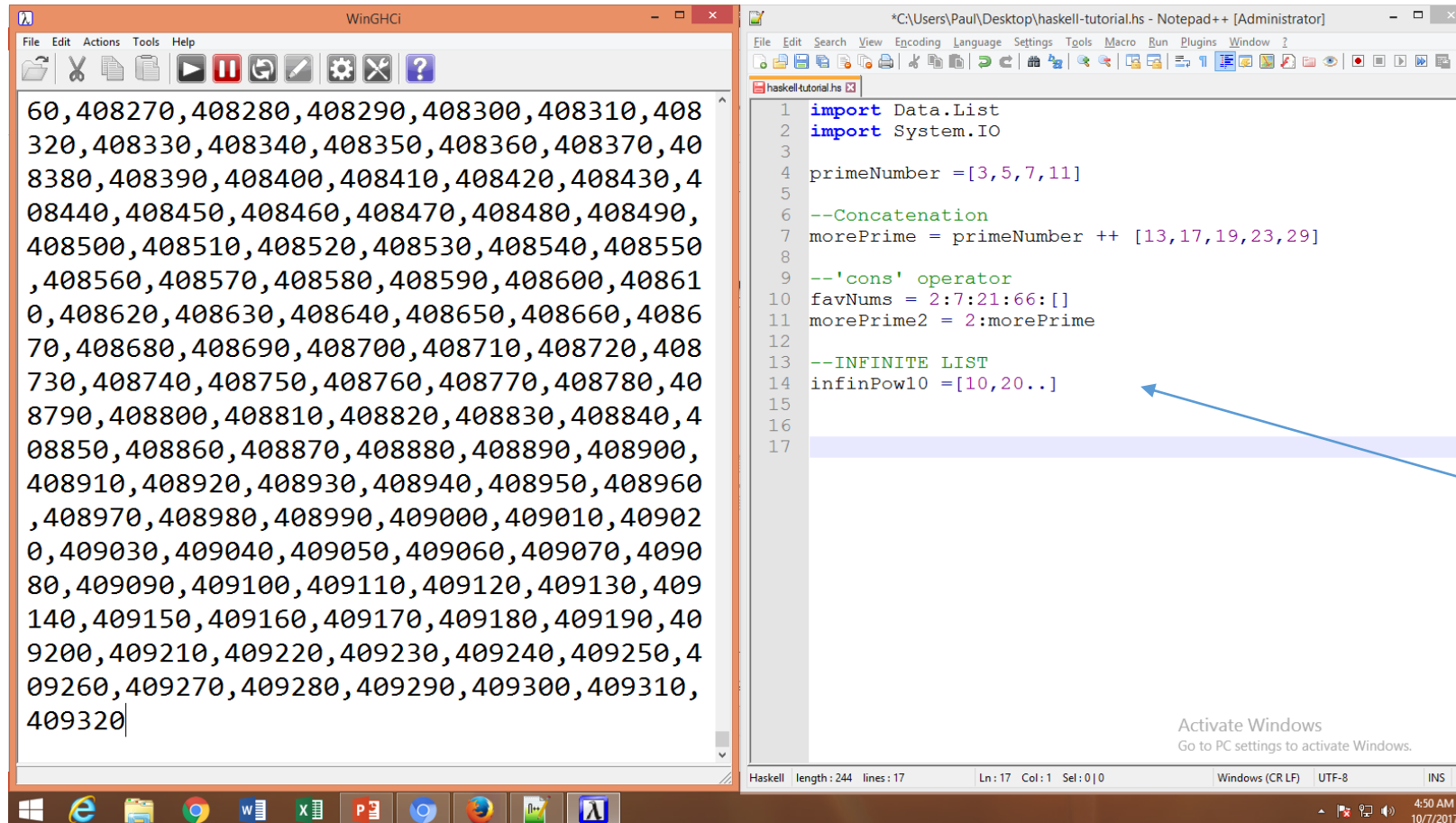
```
C:\Users\Paul\Desktop\haskell-tutorial.hs - Notepad++ [Administrator]
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
haskell-tutorial.hs
1 import Data.List
2 import System.IO
3
4 primeNumber =[3,5,7,11]
5
6 --Concatenation
7 morePrime = primeNumber ++ [13,17,19,23,29]
8
9 --'cons' operator
10 favNums = 2:7:21:66:[]
11 morePrime2 = 2:morePrime
12
13 --product of elements of a list
14 smallList = [3,4,5]
15 prodList = product smallList
16
17 --generating sequence of elements in list
18 zeroToTen = [0..10]
19 evenList = [2,4..10]
20 letterList = ['A','C'..'Z']
21
22
23
```

At the bottom of the Notepad++ window, there is a status bar with the text: "Haskell length: 405 lines: 23 Ln: 21 Col: 1 Sel: 0 | 0 Windows (CR LF) UTF-8 INS".

product

sequence

More Operations on List



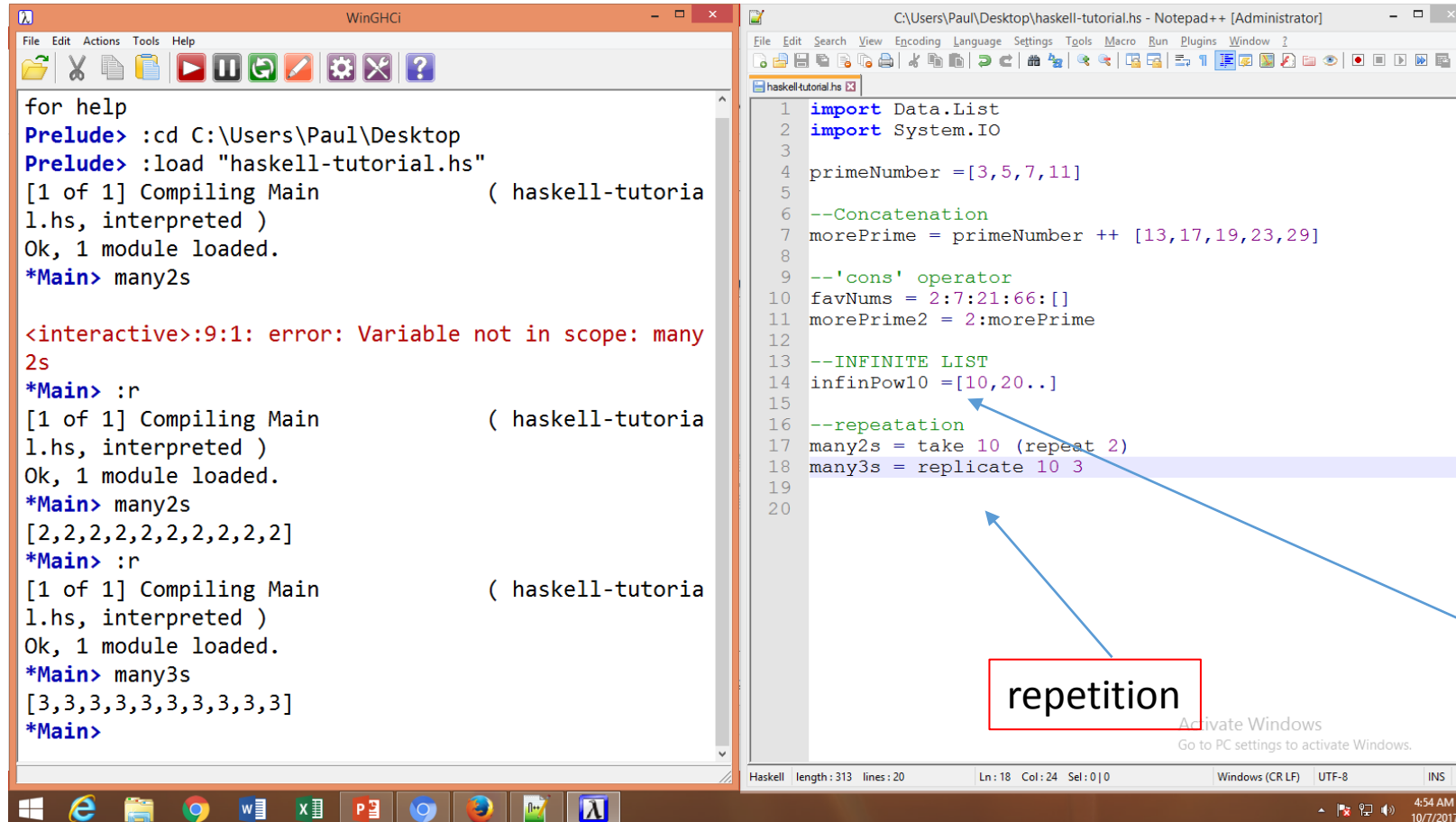
The image shows a screenshot of a computer screen with two windows. The left window, titled 'WinGHCi', displays a long list of numbers from 60 to 409320. The right window, titled '*C:\Users\Paul\Desktop\haskell-tutorial.hs - Notepad++ [Administrator]', shows Haskell code. A blue arrow points from a red-bordered box labeled 'infinite list' to the line 'infinPow10 = [10,20..]' in the code.

```
1 import Data.List
2 import System.IO
3
4 primeNumber = [3,5,7,11]
5
6 --Concatenation
7 morePrime = primeNumber ++ [13,17,19,23,29]
8
9 --'cons' operator
10 favNums = 2:7:21:66:[]
11 morePrime2 = 2:morePrime
12
13 --INFINITE LIST
14 infinPow10 = [10,20..]
15
16
17
```

WinGHCi output:

```
60,408270,408280,408290,408300,408310,408320,408330,408340,408350,408360,408370,408380,408390,408400,408410,408420,408430,408440,408450,408460,408470,408480,408490,408500,408510,408520,408530,408540,408550,408560,408570,408580,408590,408600,408610,408620,408630,408640,408650,408660,408670,408680,408690,408700,408710,408720,408730,408740,408750,408760,408770,408780,408790,408800,408810,408820,408830,408840,408850,408860,408870,408880,408890,408900,408910,408920,408930,408940,408950,408960,408970,408980,408990,409000,409010,409020,409030,409040,409050,409060,409070,409080,409090,409100,409110,409120,409130,409140,409150,409160,409170,409180,409190,409200,409210,409220,409230,409240,409250,409260,409270,409280,409290,409300,409310,409320
```

More Operations on List



The image shows a screenshot of a Haskell tutorial. On the left, the WinGHCi window displays the interactive session. On the right, the Notepad++ window shows the source code for 'haskell-tutorial.hs'.

WinGHCi Window:

```
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> many2s

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

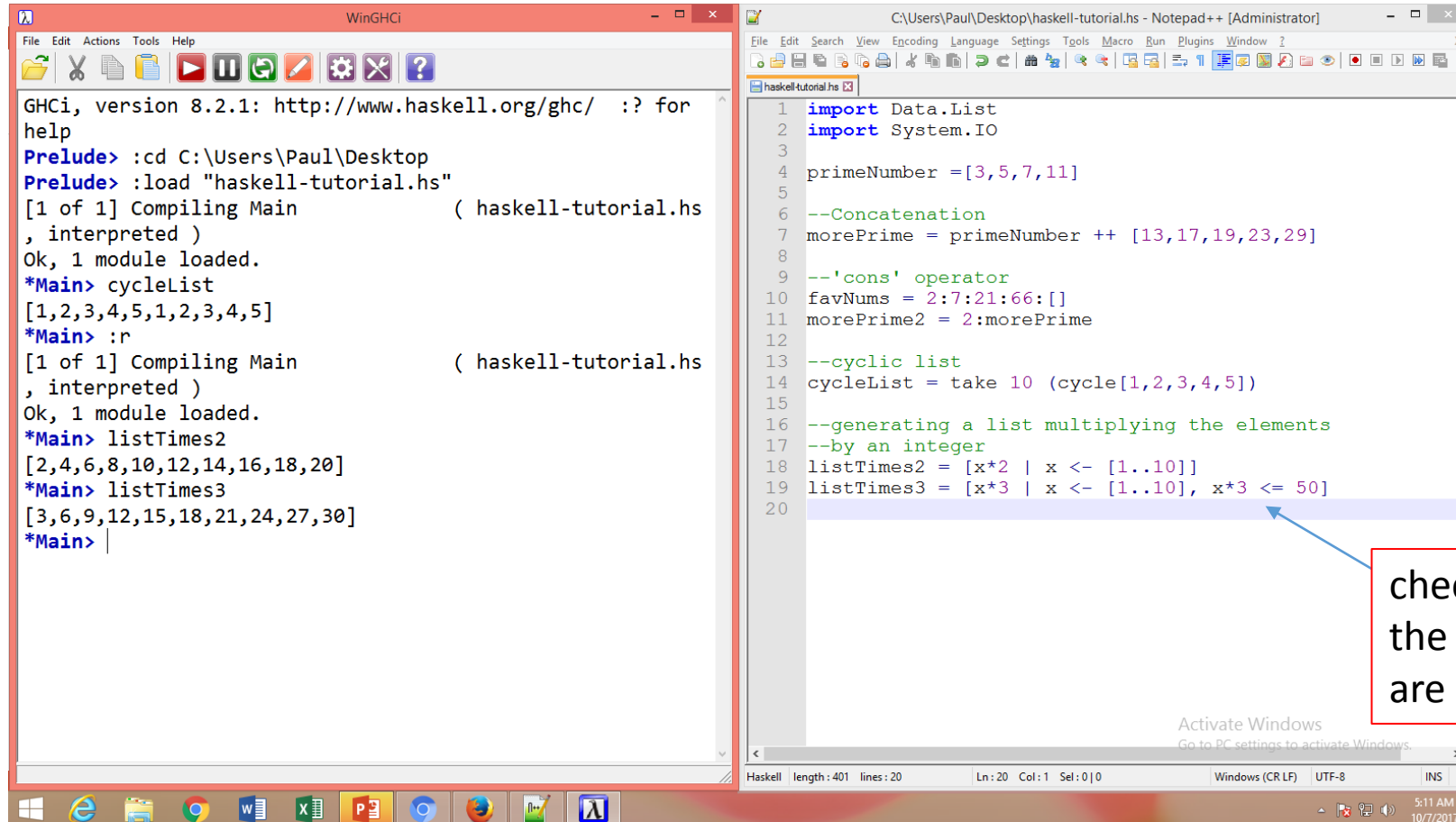
Notepad++ Window (haskell-tutorial.hs):

```
1 import Data.List
2 import System.IO
3
4 primeNumber =[3,5,7,11]
5
6 --Concatenation
7 morePrime = primeNumber ++ [13,17,19,23,29]
8
9 --'cons' operator
10 favNums = 2:7:21:66:[]
11 morePrime2 = 2:morePrime
12
13 --INFINITE LIST
14 infinPow10 =[10,20..]
15
16 --repeataction
17 many2s = take 10 (repeat 2)
18 many3s = replicate 10 3
19
20
```

Arrows point from the `repeat` and `replicate` functions in the code to the text box.

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



The image shows two windows side-by-side. The left window is WinGHCi, and the right window is Notepad++.

WinGHCi window:

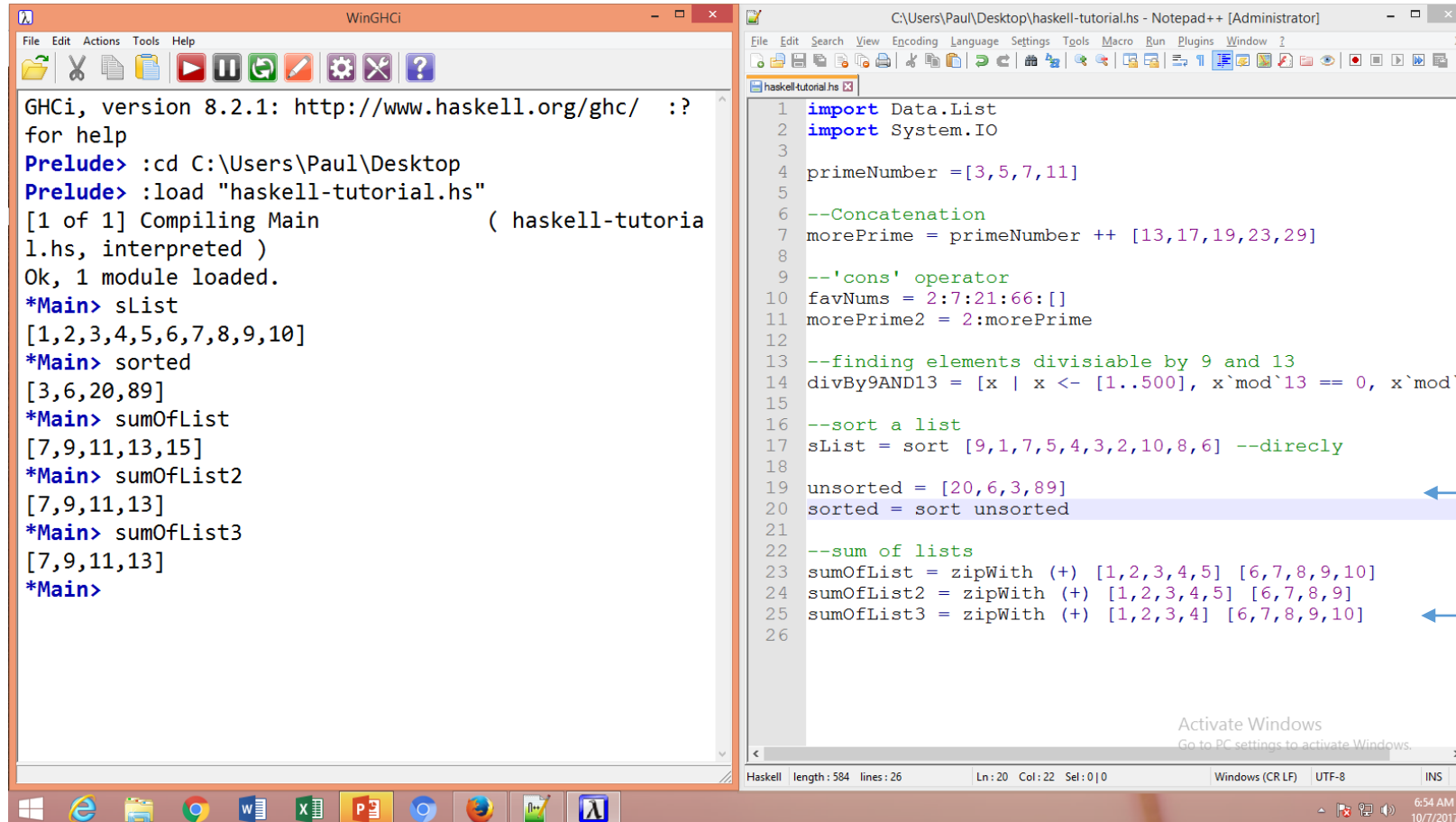
```
GHCi, version 8.2.1: http://www.haskell.org/ghc/  :? for help
Prelude> :cd C:\Users\Paul\Desktop
Prelude> :load "haskell-tutorial.hs"
[1 of 1] Compiling Main             ( haskell-tutorial.hs
, interpreted )
Ok, 1 module loaded.
*Main> cycleList
[1,2,3,4,5,1,2,3,4,5]
*Main> :r
[1 of 1] Compiling 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> |
```

Notepad++ window:

```
1 import Data.List
2 import System.IO
3
4 primeNumber =[3,5,7,11]
5
6 --Concatenation
7 morePrime = primeNumber ++ [13,17,19,23,29]
8
9 --'cons' operator
10 favNums = 2:7:21:66:[]
11 morePrime2 = 2:morePrime
12
13 --cyclic list
14 cycleList = take 10 (cycle[1,2,3,4,5])
15
16 --generating a list multiplying the elements
17 --by an integer
18 listTimes2 = [x*2 | x <- [1..10]]
19 listTimes3 = [x*3 | x <- [1..10], x*3 <= 50]
20
```

A red box with a blue arrow points to line 19 in the Notepad++ window. The text inside the box says: "check whether the element generated are less than 50 or not".

More Operations on List



The image shows two windows side-by-side. The left window is WinGHCi, and the right window is Notepad++.

WinGHCi window:

```
WinGHCi
File Edit Actions Tools Help
[Icons]
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> 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>
```

Notepad++ window:

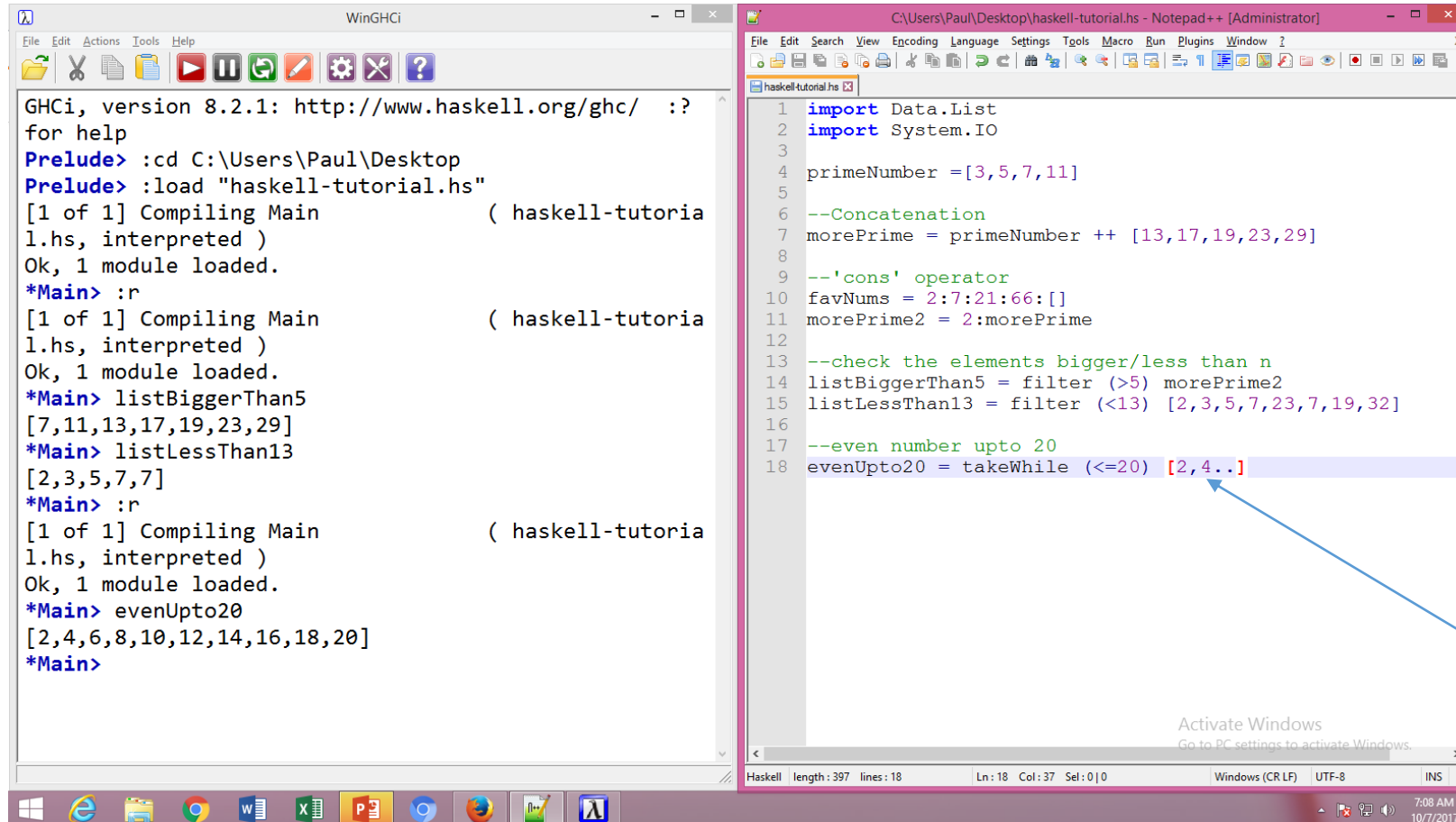
```
C:\Users\Paul\Desktop\haskell-tutorial.hs - Notepad++ [Administrator]
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
haskell-tutorial.hs
1 import Data.List
2 import System.IO
3
4 primeNumber =[3,5,7,11]
5
6 --Concatenation
7 morePrime = primeNumber ++ [13,17,19,23,29]
8
9 --'cons' operator
10 favNums = 2:7:21:66:[]
11 morePrime2 = 2:morePrime
12
13 --finding elements divisiable by 9 and 13
14 divBy9AND13 = [x | x <- [1..500], x`mod`13 == 0, x`mod`
15
16 --sort a list
17 sList = sort [9,1,7,5,4,3,2,10,8,6] --direcly
18
19 unsorted = [20,6,3,89]
20 sorted = sort unsorted
21
22 --sum of lists
23 sumOfList = zipWith (+) [1,2,3,4,5] [6,7,8,9,10]
24 sumOfList2 = zipWith (+) [1,2,3,4,5] [6,7,8,9]
25 sumOfList3 = zipWith (+) [1,2,3,4] [6,7,8,9,10]
26
```

Annotations:

- A red box labeled "sorting" with an arrow pointing to line 17: `sList = sort [9,1,7,5,4,3,2,10,8,6] --direcly`
- A red box labeled "summation" with an arrow pointing to line 23: `sumOfList = zipWith (+) [1,2,3,4,5] [6,7,8,9,10]`

Windows taskbar at the bottom shows the time as 6:54 AM on 10/7/2017.

More Operations on List



The image shows two windows side-by-side. The left window is WinGHCi, a Haskell interpreter. The right window is Notepad++ showing a Haskell file named 'haskell-tutorial.hs'.

WinGHCi Output:

```
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> :r
[1 of 1] Compiling Main             ( haskell-tutorial.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-tutorial.hs, interpreted )
Ok, 1 module loaded.
*Main> evenUpto20
[2,4,6,8,10,12,14,16,18,20]
*Main>
```

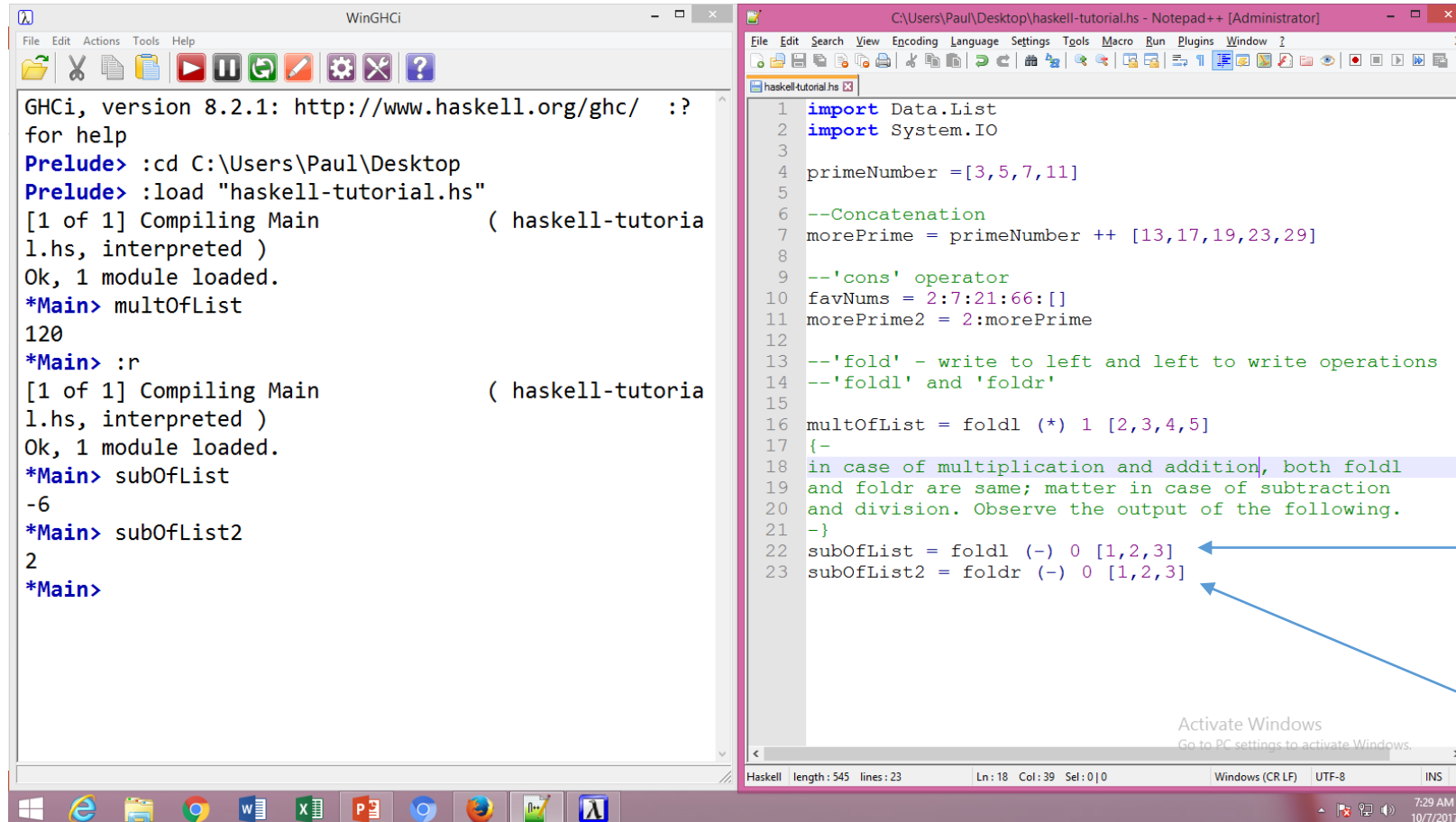
haskell-tutorial.hs Code:

```
1 import Data.List
2 import System.IO
3
4 primeNumber = [3,5,7,11]
5
6 --Concatenation
7 morePrime = primeNumber ++ [13,17,19,23,29]
8
9 --'cons' operator
10 favNums = 2:7:21:66:[]
11 morePrime2 = 2:morePrime
12
13 --check the elements bigger/less than n
14 listBiggerThan5 = filter (>5) morePrime2
15 listLessThan13 = filter (<13) [2,3,5,7,23,7,19,32]
16
17 --even number upto 20
18 evenUpto20 = takeWhile (<=20) [2,4..]
```

An arrow points from the text box on the right to the line 18 in the code.

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

More Operations on List



The image shows two windows side-by-side. The left window is WinGHCi, and the right window is Notepad++.

WinGHCi window:

```
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> multOfList
120
*Main> :r
[1 of 1] Compiling Main             ( haskell-tutorial.hs, interpreted )
Ok, 1 module loaded.
*Main> subOfList
-6
*Main> subOfList2
2
*Main>
```

Notepad++ window:

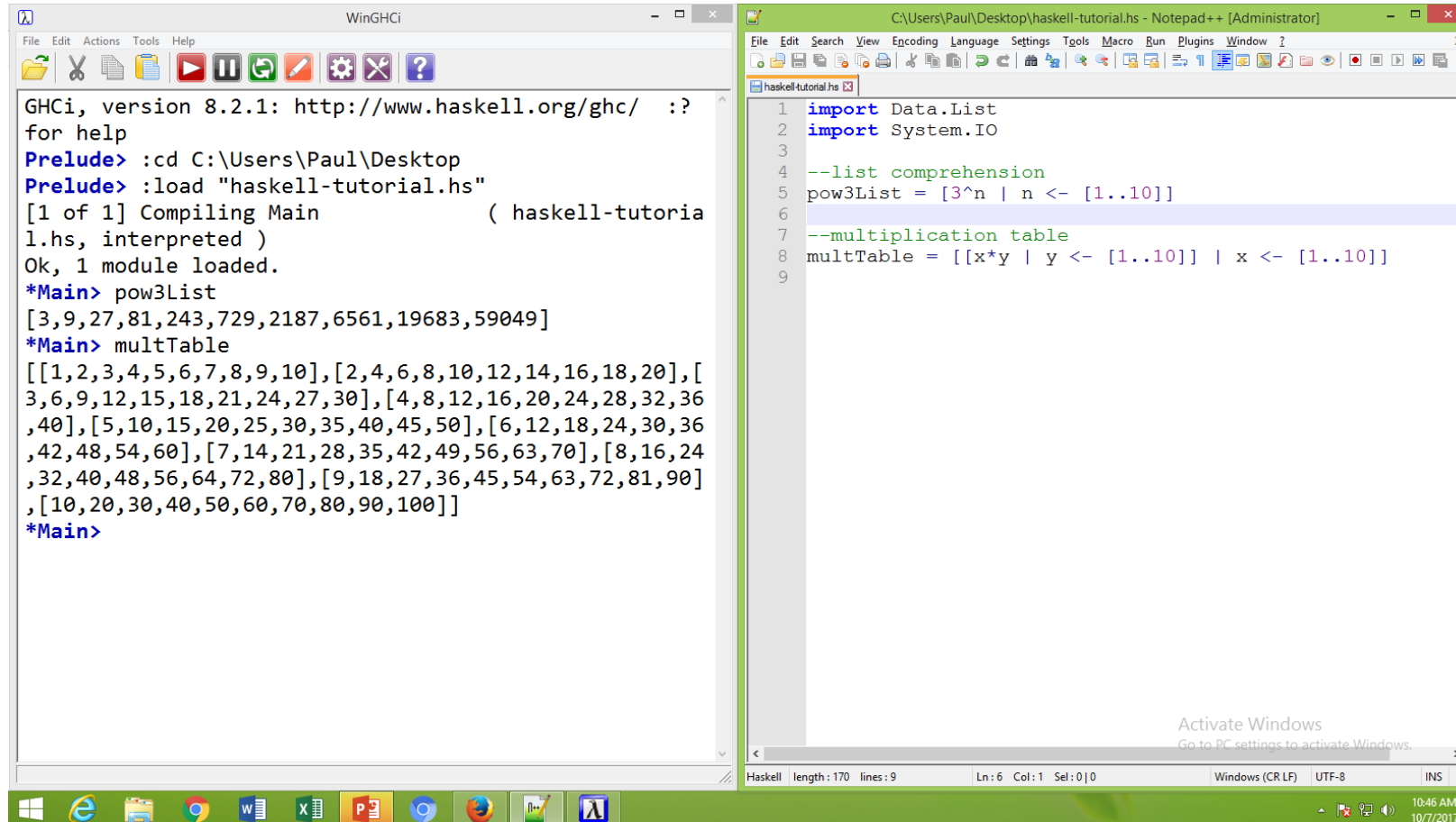
```
1 import Data.List
2 import System.IO
3
4 primeNumber =[3,5,7,11]
5
6 --Concatenation
7 morePrime = primeNumber ++ [13,17,19,23,29]
8
9 --'cons' operator
10 favNums = 2:7:21:66:[]
11 morePrime2 = 2:morePrime
12
13 --'fold' - write to left and left to write operations
14 --'foldl' and 'foldr'
15
16 multOfList = foldl (*) 1 [2,3,4,5]
17 {-
18 in case of multiplication and addition, both foldl
19 and foldr are same; matter in case of subtraction
20 and division. Observe the output of the following.
21 -}
22 subOfList = foldl (-) 0 [1,2,3]
23 subOfList2 = foldr (-) 0 [1,2,3]
```

foldl & foldr

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

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

More Operations on List



The screenshot displays two windows side-by-side. The left window, titled 'WinGHCi', shows the Haskell interpreter's command line. The right window, titled 'C:\Users\Paul\Desktop\haskell-tutorial.hs - Notepad++ [Administrator]', shows the source code for the tutorial.

```
WinGHCi
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> 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>
```

```
C:\Users\Paul\Desktop\haskell-tutorial.hs - Notepad++ [Administrator]
1 import Data.List
2 import System.IO
3
4 --list comprehension
5 pow3List = [3^n | n <- [1..10]]
6
7 --multiplication table
8 multTable = [[x*y | y <- [1..10]] | x <- [1..10]]
9
```

Windows taskbar at the bottom shows the time as 10:46 AM on 10/7/2017.

Multiple Data Type



The screenshot displays a Windows environment with two open windows. The left window, titled 'WinGHCi', shows the Haskell interpreter's command line. The right window, titled 'C:\Users\Paul\Desktop\haskell-tutorial.hs - Notepad++ [Administrator]', shows the source code of a Haskell program.

```
WinGHCi
File Edit Actions Tools Help
[Icons]
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> bobsName
"Bob Smith"
*Main> bobsAge
52
*Main> |

C:\Users\Paul\Desktop\haskell-tutorial.hs - Notepad++ [Administrator]
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
[Icons]
haskell-tutorial.hs
1 import Data.List
2 import System.IO
3
4 --random tuple
5 randTuple = (1, "Random Tuple")
6 bobSmith = ("Bob Smith",52)
7 bobsName = fst bobSmith
8 bobsAge = snd bobSmith
9
10
11
Haskell length: 170 lines: 11 Ln: 10 Col: 1 Sel: 0|0 Windows (CR LF) UTF-8 INS
10:52 AM 10/7/2017
```

Function Declaration



The screenshot displays two windows on a Windows 10 desktop. The left window is the WinGHCi terminal, showing the following commands and output:

```
WinGHCi
File Edit Actions Tools Help
[Icons]
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> :t addMe
addMe :: Int -> Int -> Int
*Main> addMe 56 9999
10055
*Main> |
```

The right window is a Notepad++ editor titled "C:\Users\Paul\Desktop\haskell-tutorial.hs - Notepad++ [Administrator]". It contains the following Haskell code:

```
C:\Users\Paul\Desktop\haskell-tutorial.hs - Notepad++ [Administrator]
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
haskell-tutorial.hs
1 import Data.List
2 import System.IO
3
4 addMe :: Int -> Int -> Int
5 --funcName parameter1 parameter2=operation(return value)
6
7 addMe x y =x+y
```

The status bar at the bottom of the Notepad++ window shows "Haskell length: 140 lines: 7 Ln: 7 Col: 15 Sel: 0|0 Windows (CR LF) UTF-8 INS". The Windows taskbar at the bottom shows the time as 11:02 AM on 10/7/2017.

User Type Declaration

A screenshot of a Windows desktop environment showing two windows. The left window is titled 'WinGHCi' and displays the Haskell interpreter's command line. The right window is titled 'C:\Users\Paul\Desktop\haskell-tutorial.hs - Notepad++ [Administrator]' and shows a Haskell source file. The taskbar at the bottom includes icons for Windows, Edge, File Explorer, Chrome, Word, Excel, PowerPoint, and other applications. The system clock in the bottom right corner shows the time as 11:09 AM on 10/7/2017.

```
WinGHCi
File Edit Actions Tools Help
[Icons]
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> sumMe 4.5 6.7
11.2
*Main> addTuples (1,2) (3,4)
(4,6)
*Main>

C:\Users\Paul\Desktop\haskell-tutorial.hs - Notepad++ [Administrator]
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
[Icons]
haskell-tutorial.hs
1  import Data.List
2  import System.IO
3
4  sumMe x y = x + y
5
6  addTuples :: (Int,Int) -> (Int,Int) -> (Int,Int)
7
8  addTuples (x,y) (x2,y2) = (x+x2,y+y2)
9
10
[Status Bar] Haskell length: 152 lines: 10 Ln: 10 Col: 1 Sel: 0|0 Windows (CR LF) UTF-8 INS
[Taskbar] Windows Edge File Explorer Chrome Word Excel PowerPoint
[System Tray] 11:09 AM 10/7/2017
```

User Type Declaration



```
WinGHCi
File Edit Actions Tools Help
[Icons]
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> sumMe 4.5 6.7
11.2
*Main> addTuples (1,2) (3,4)
(4,6)
*Main> :r
[1 of 1] Compiling Main             ( haskell-tutorial.hs, interpreted )
Ok, 1 module loaded.
*Main> whatAge 18
"You Can Vote"
*Main> whatAge 16
"You Can Drive"
*Main> whatAge 67
"Nothing Important"
*Main>

C:\Users\Paul\Desktop\haskell-tutorial.hs - Notepad++ [Administrator]
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
haskell-tutorial.hs
1 import Data.List
2 import System.IO
3
4 whatAge :: Int ->String
5
6 whatAge 16 = "You Can Drive"
7 whatAge 18 = "You Can Vote"
8 whatAge 21 = "You are Adult"
9 whatAge x = "Nothing Important"
10
11
Haskell length: 189 lines: 11 Ln: 11 Col: 1 Sel: 0|0 Windows (CR LF) UTF-8 INS
11:22 AM 10/7/2017
```

Factorial (by recursion and by product)



```
WinGHCi
File Edit Actions Tools Help
[Icons]
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> |

C:\Users\Paul\Desktop\haskell-tutorial.hs - Notepad++ [Administrator]
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
[Icons]
1 import Data.List
2 import System.IO
3
4 --factorial by recursion
5 factorial :: Int -> Int
6 factorial 0 = 1
7 factorial n = n * factorial (n-1)
8
9
10 --factorial by product
11 prodFactorial n = product [1..n]
12
```

Guard (where clause)



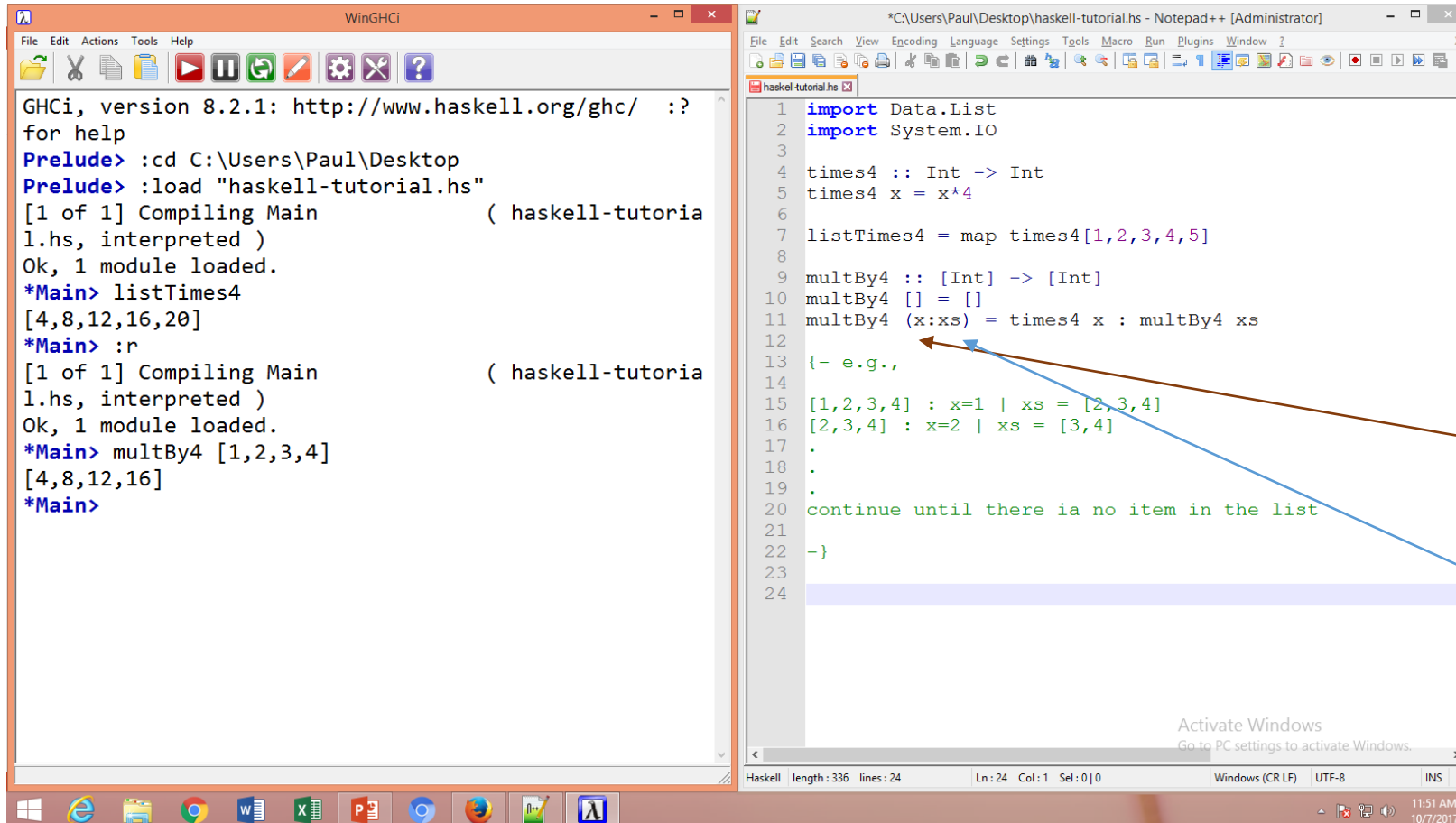
```
*Main> factorial 6
720
*Main> :r
[1 of 1] Compiling Main             ( haskell-tutorial.hs, interpreted )
Ok, 1 module loaded.
*Main> prodFactorial 6
720
*Main> :r
[1 of 1] Compiling Main             ( haskell-tutorial.hs, interpreted )

haskell-tutorial.hs:7:1: warning: [-Wtabs]
    Tab character found here, and in five further locations.
    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>
```

```
1 import Data.List
2 import System.IO
3
4 batAvgRating :: Double -> Double -> String
5
6 batAvgRating hits atBats
7   | avg <= 0.200 = "Terrible Batting"
8   | avg <= 0.250 = "Average Player"
9   | avg <= 0.280 = "You are doing pretty good!"
10  | otherwise = "You are a Superstar"
11
12 where avg = hits / atBats
```

Windows (CR LF) UTF-8 INS
Haskell length: 299 lines: 12 Ln: 12 Col: 30 Sel: 0 | 0
11:38 AM 10/7/2017

Higher Order Functions

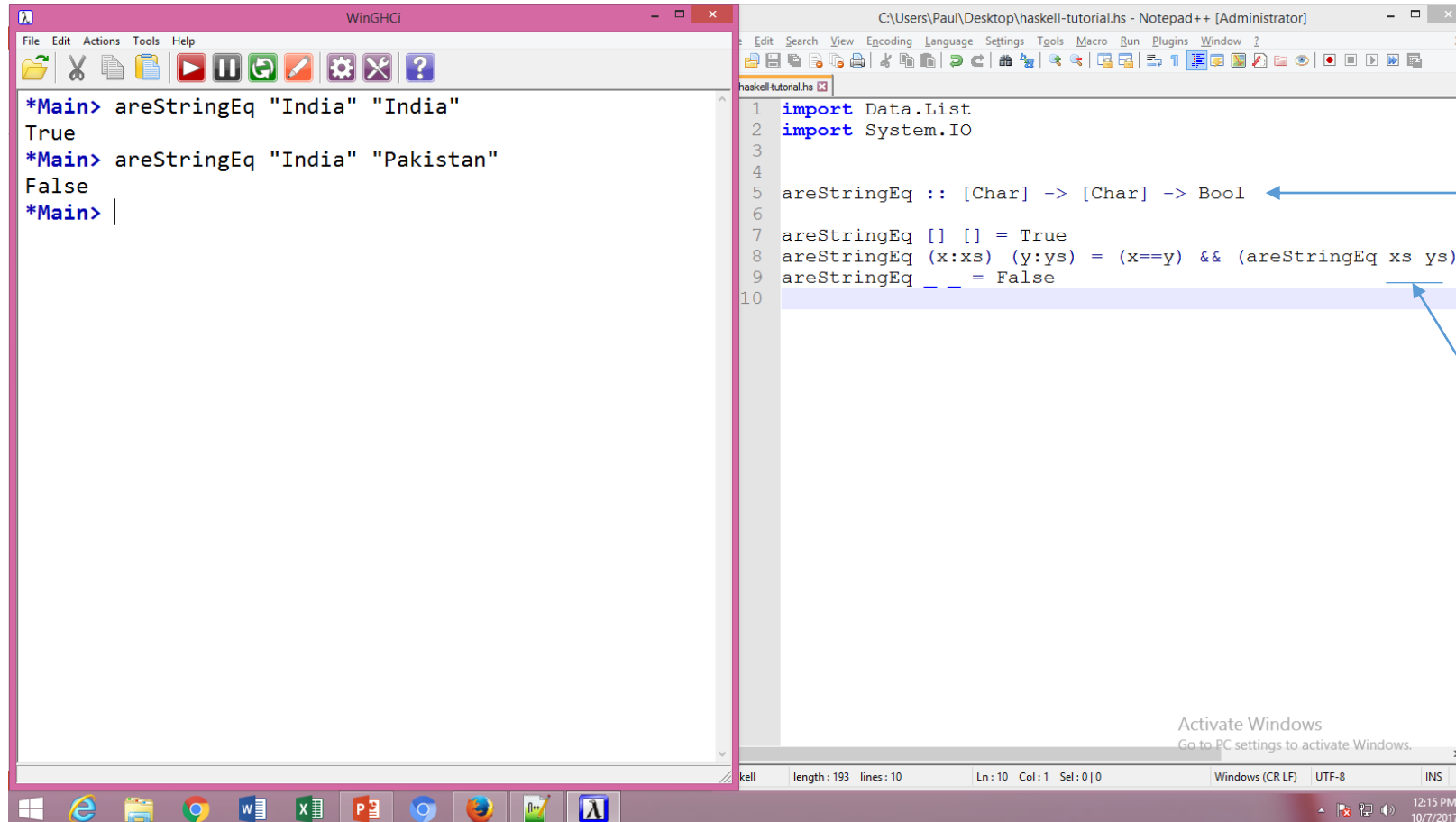


```
WinGHCi
File Edit Actions Tools Help
[Icons]
GHCi, version 8.2.1: http://www.haskell.org/ghc/  :?
for help
Prelude> :cd C:\Users\Paul\Desktop
Prelude> :load "haskell-tutorial.hs"
[1 of 1] Compiling Main             ( haskell-tutorial.hs, interpreted )
Ok, 1 module loaded.
*Main> listTimes4
[4,8,12,16,20]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

*C:\Users\Paul\Desktop\haskell-tutorial.hs - Notepad++ [Administrator]
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
haskell-tutorial.hs
1 import Data.List
2 import System.IO
3
4 times4 :: Int -> Int
5 times4 x = x*4
6
7 listTimes4 = map times4 [1,2,3,4,5]
8
9 multBy4 :: [Int] -> [Int]
10 multBy4 [] = []
11 multBy4 (x:xs) = times4 x : multBy4 xs
12
13 {- e.g.,
14 [1,2,3,4] : x=1 | xs = [2,3,4]
15 [2,3,4]  : x=2 | xs = [3,4]
16 [3,4]   : x=3 | xs = [4]
17 [4]     : x=4 | xs = []
18 []      : x=5 | xs = []
19 -}
20 continue until there is no item in the list
21 -}
22
23
24
Haskell length: 336 lines: 24 Ln: 24 Col: 1 Sel: 0|0 Windows (CR LF) UTF-8 INS
11:51 AM 10/7/2017
```

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



The image shows two windows side-by-side. The left window is WinGHCi, a Haskell interpreter, showing the execution of the `areStringEq` function. The right window is Notepad++ showing the source code of the `areStringEq` function. The function is defined as follows:

```
1 import Data.List
2 import System.IO
3
4 areStringEq :: [Char] -> [Char] -> Bool
5
6 areStringEq [] [] = True
7 areStringEq (x:xs) (y:ys) = (x==y) && (areStringEq xs ys)
8 areStringEq _ _ = False
9
10
```

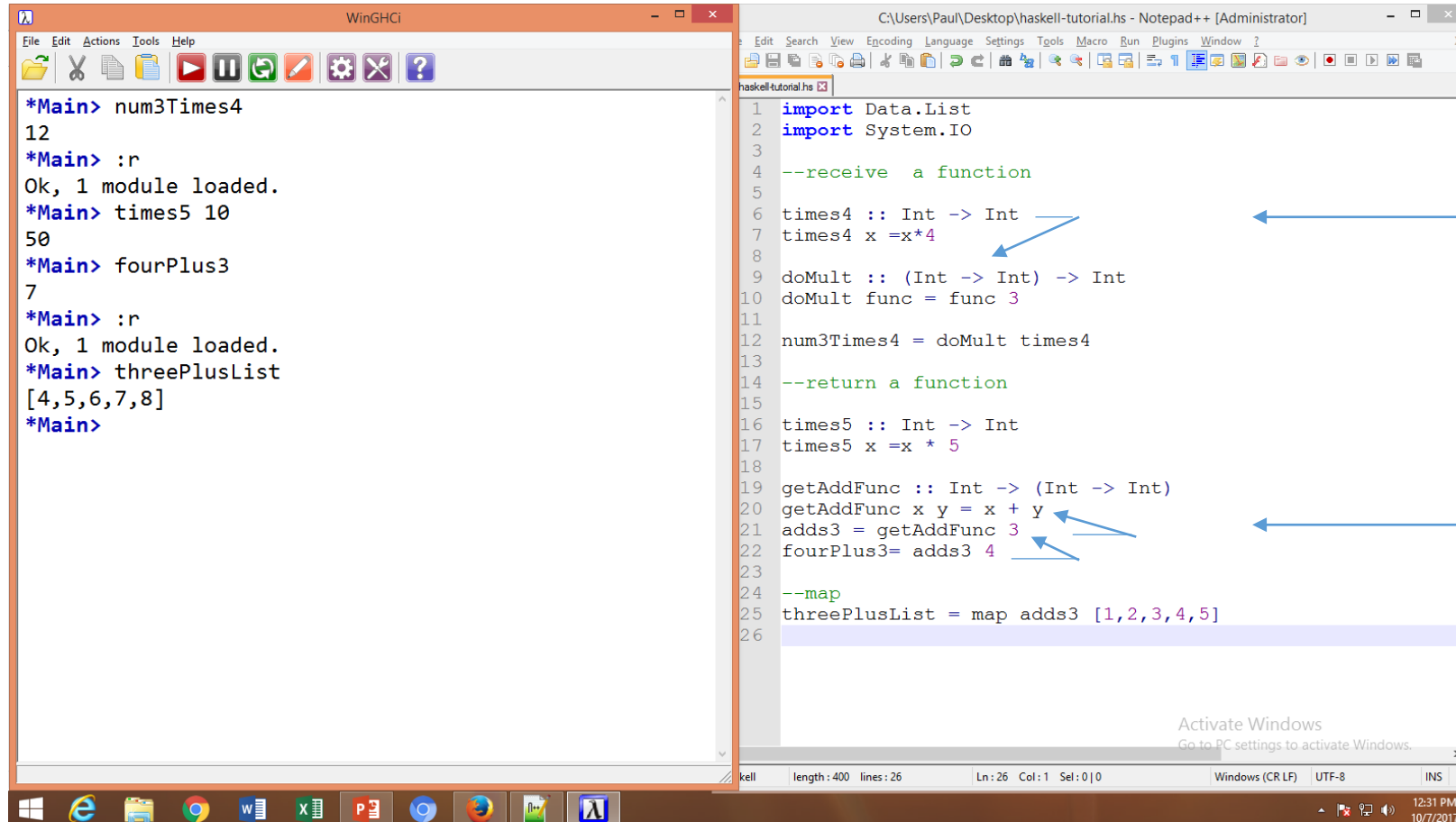
The WinGHCi window shows the following interactions:

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

return type

remaining

Receive and Return a Function



```
WinGHCi
File Edit Actions Tools Help
12
*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>

C:\Users\Paul\Desktop\haskell-tutorial.hs - Notepad++ [Administrator]
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
haskell-tutorial.hs
1 import Data.List
2 import System.IO
3
4 --receive a function
5
6 times4 :: Int -> Int
7 times4 x = x*4
8
9 doMult :: (Int -> Int) -> Int
10 doMult func = func 3
11
12 num3Times4 = doMult times4
13
14 --return a function
15
16 times5 :: Int -> Int
17 times5 x = x * 5
18
19 getAddFunc :: Int -> (Int -> Int)
20 getAddFunc x y = x + y
21 adds3 = getAddFunc 3
22 fourPlus3 = adds3 4
23
24 --map
25 threePlusList = map adds3 [1,2,3,4,5]
26

length: 400 lines: 26 Ln: 26 Col: 1 Sel: 0|0 Windows (CR LF) UTF-8 INS
12:31 PM 10/7/2017
```

Other Operators

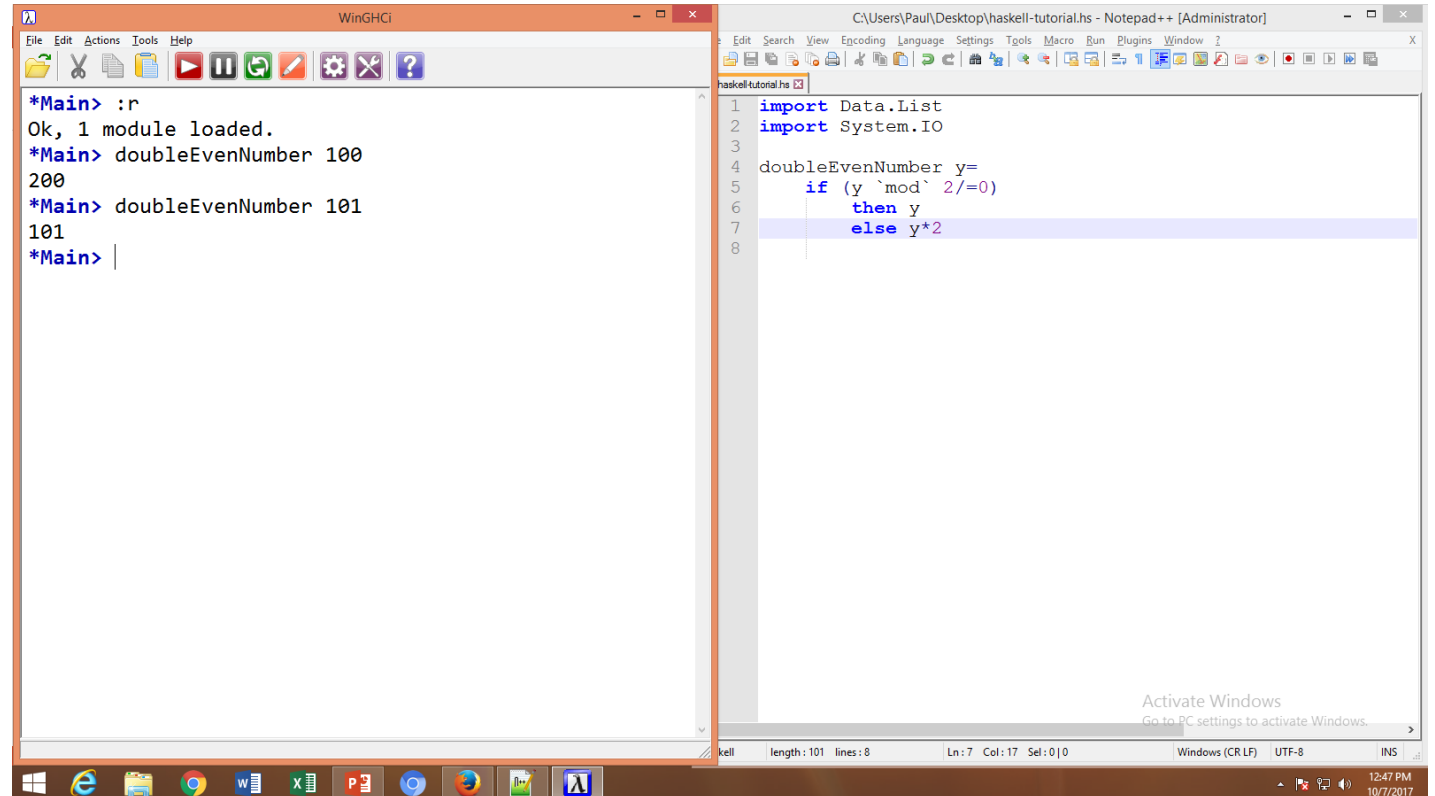
➤ Comparison

- < --less than
- > --greater than
- <= --less than equal to
- >= --greater than equal to
- == --equal to

➤ Logical

- && --AND
- || --OR
- not --NOT

Example

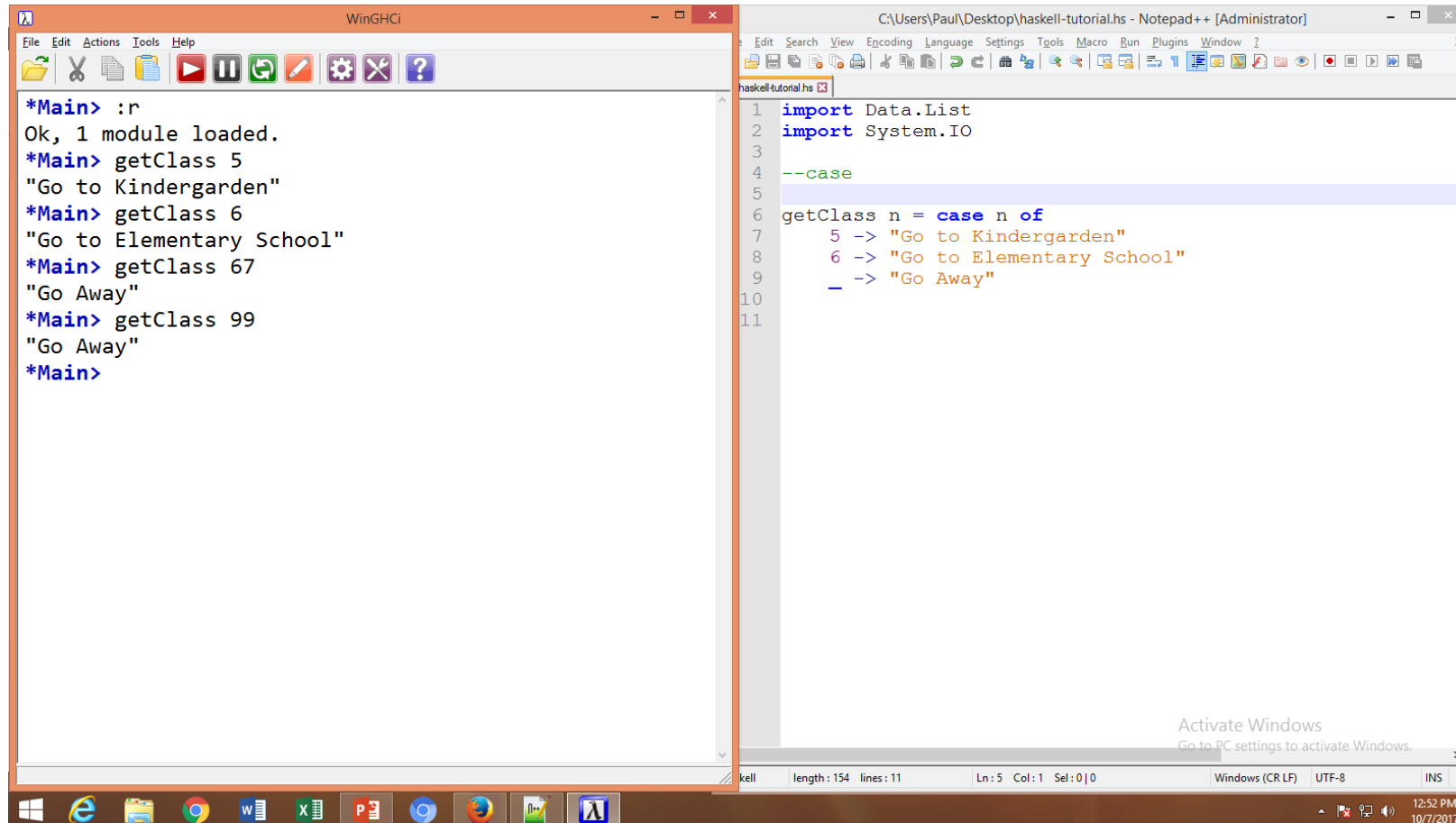


The screenshot shows two windows. The left window is WinGHC, a Haskell interpreter, showing the execution of a program. The right window is Notepad++, showing the source code of the program.

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

```
1 import Data.List
2 import System.IO
3
4 doubleEvenNumber y=
5     if (y `mod` 2/=0)
6     then y
7     else y*2
8
```

Case



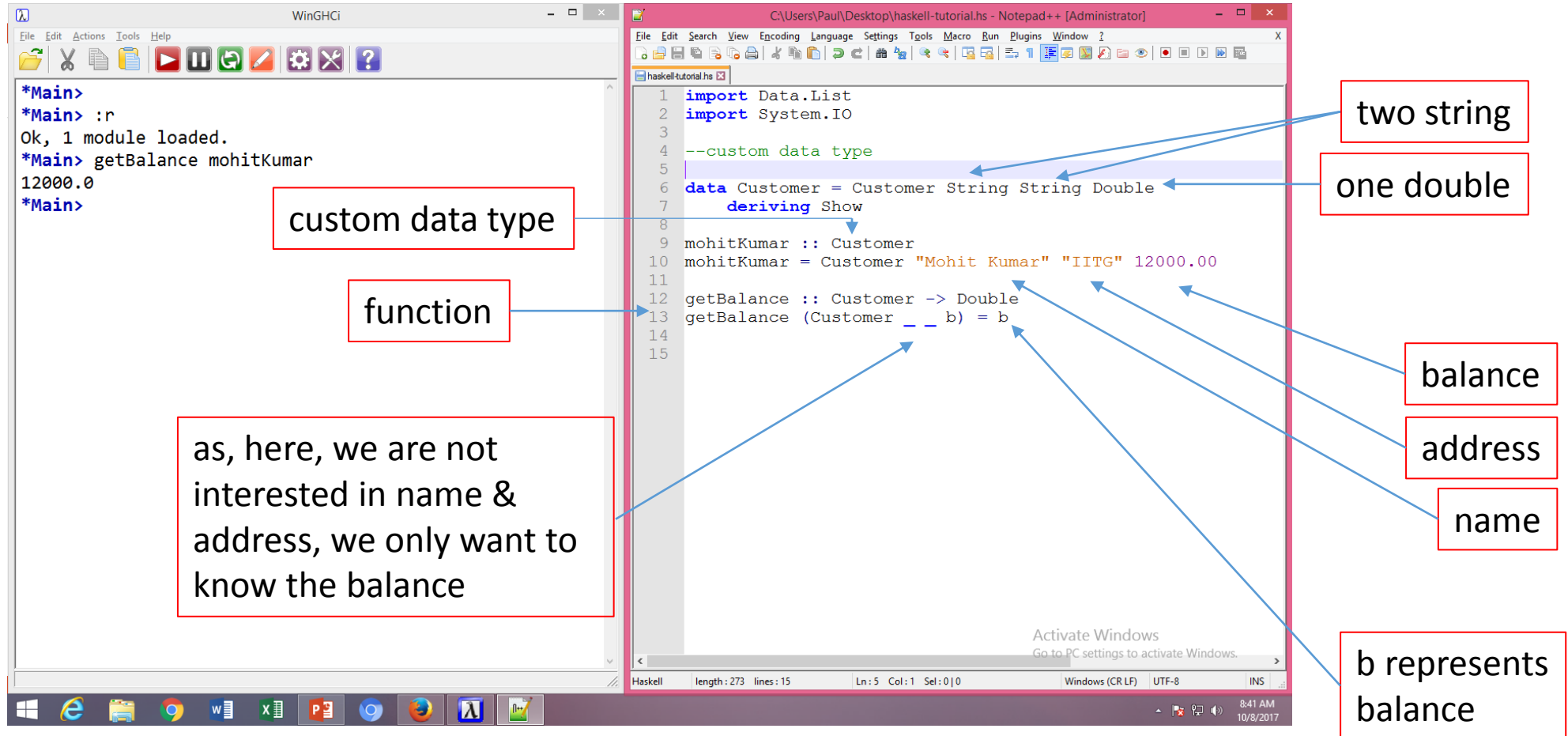
The screenshot displays two windows side-by-side. The left window, titled 'WinGHCi', shows the Haskell interpreter's prompt and the execution of a program. The right window, titled 'C:\Users\Paul\Desktop\haskell-tutorial.hs - Notepad++ [Administrator]', shows the source code of the program being executed.

```
*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>
```

```
1 import Data.List
2 import System.IO
3
4 --case
5
6 getClass n = case n of
7     5 -> "Go to Kindergarden"
8     6 -> "Go to Elementary School"
9     _ -> "Go Away"
10
11
```

At the bottom of the Notepad++ window, a status bar shows: 'haskell length: 154 lines: 11 Ln: 5 Col: 1 Sel: 0|0 Windows (CR LF) UTF-8 INS'. A watermark 'Activate Windows Go to PC settings to activate Windows.' is visible in the bottom right corner of the Notepad++ window.

Custom Data Type



The image shows a screenshot of a Haskell tutorial. On the left, the WinGHCi window displays the following commands and output:

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

Annotations for the WinGHCi window:

- custom data type**: Points to the `data Customer` definition in the Notepad++ window.
- function**: Points to the `getBalance` function definition in the Notepad++ window.
- as, here, we are not interested in name & address, we only want to know the balance**: Points to the `getBalance` function definition in the Notepad++ window.

On the right, the Notepad++ window shows the Haskell code:

```
1 import Data.List
2 import System.IO
3
4 --custom data type
5
6 data Customer = Customer String String Double
7   deriving Show
8
9 mohitKumar :: Customer
10 mohitKumar = Customer "Mohit Kumar" "IITG" 12000.00
11
12 getBalance :: Customer -> Double
13 getBalance (Customer _ _ b) = b
14
15
```

Annotations for the Notepad++ window:

- two string**: Points to the first two `String` arguments in the `Customer` constructor.
- one double**: Points to the `Double` argument in the `Customer` constructor.
- balance**: Points to the `b` parameter in the `getBalance` function signature.
- address**: Points to the second `String` argument in the `Customer` constructor.
- name**: Points to the first `String` argument in the `Customer` constructor.
- b represents balance**: Points to the `b` parameter in the `getBalance` function body.

Type Classes



```
*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>
```

```
1 import Data.List
2 import System.IO
3
4 --custom data type
5
6 data Employee = Employee {name :: String,
7                             position :: String,
8                             idNum :: Int
9                             } deriving (Eq, Show)
10
11 ankitKumar = Employee{name="Ankit Kumar", position="Manager", idNum=1001}
12 darshitPatel = Employee{name="Darshit Patel", position="Sales", idNum=3001}
13
14 --check whether Ankit and Darshit are same or not
15 isAnkitDarshit = ankitKumar == darshitPatel
16
17 --shows the data of employee
18 ankitKumarData = show ankitKumar
19 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
