

C/CPS 506

Assignment Project Exam Help

Comparative Programming Languages

<https://powcoder.com>

Prof. Alex Ufkes

Add WeChat powcoder

Topic 7: Pattern matching, custom types in Haskell

Notice!

**Obligatory copyright notice in the age of digital
delivery and online classrooms:**

Assignment Project Exam Help

<https://powcoder.com>

The copyright to this original work is held by Alex Ufkes. Students registered in course CCPS 505 can use this material for the purposes of this course but no other use is permitted, and there can be no sale or transfer or use of the work for any other purpose without explicit permission of Alex Ufkes.

Course Administration



<https://powcoder.com>

Add WeChat powcoder

It's important to start thinking about the assignment if you haven't already.

Any Questions?

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder



Let's Get Started!

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

Types in Haskell

Statically Typed:

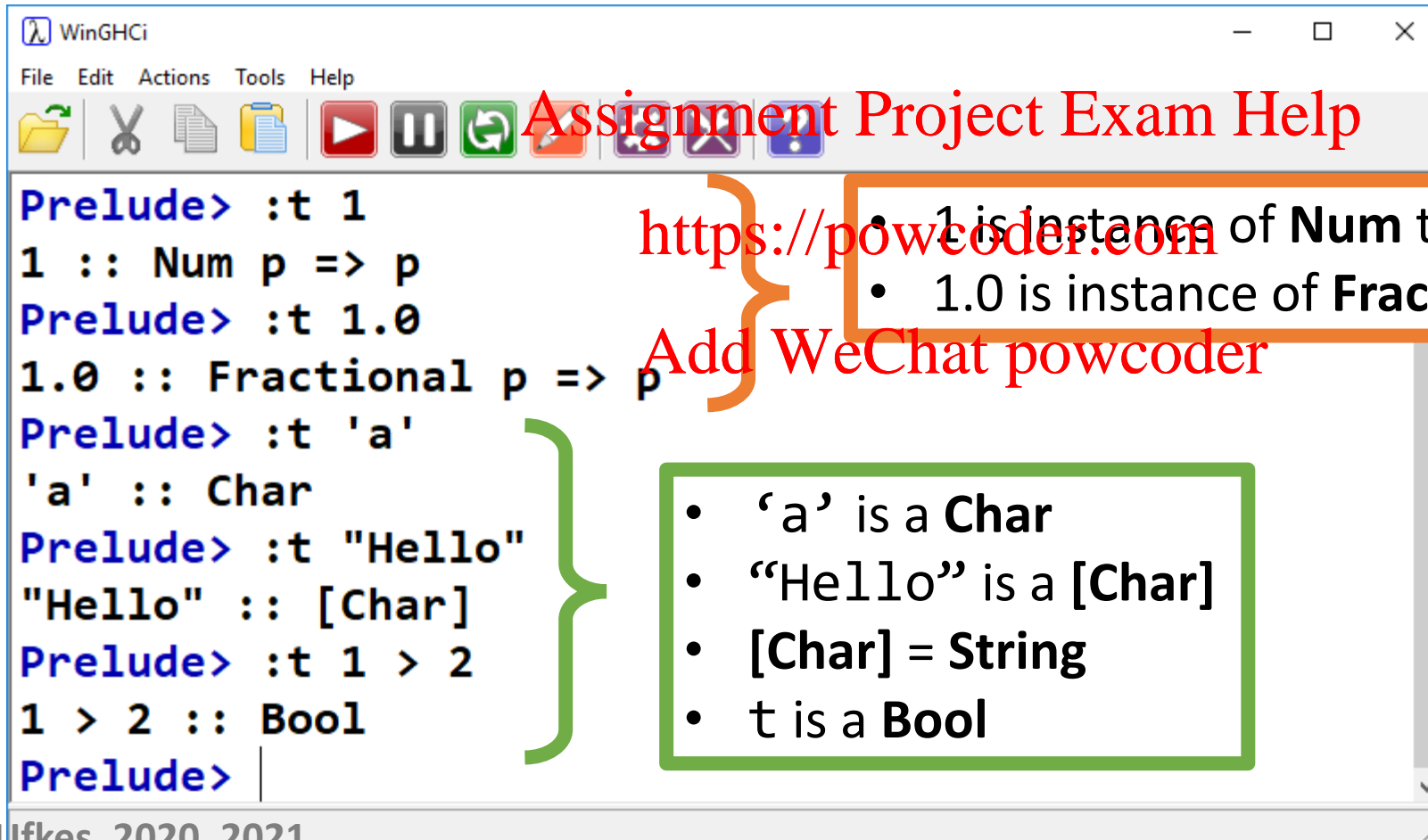
- Haskell uses static type checking.
- Every expression is assigned a type.
- If a function's arguments aren't the expected type, a compile error occurs.

Type Inference

- Like Python, and unlike Java, we need not specify type.
- It is inferred by the context: `X = "Hello"`, `X` is a string.
- However, we can explicitly specify types.
- Good practice when we know what types we want; compiler will give errors upon type mismatch.

Types in Haskell

`:t` can be used to reveal type:



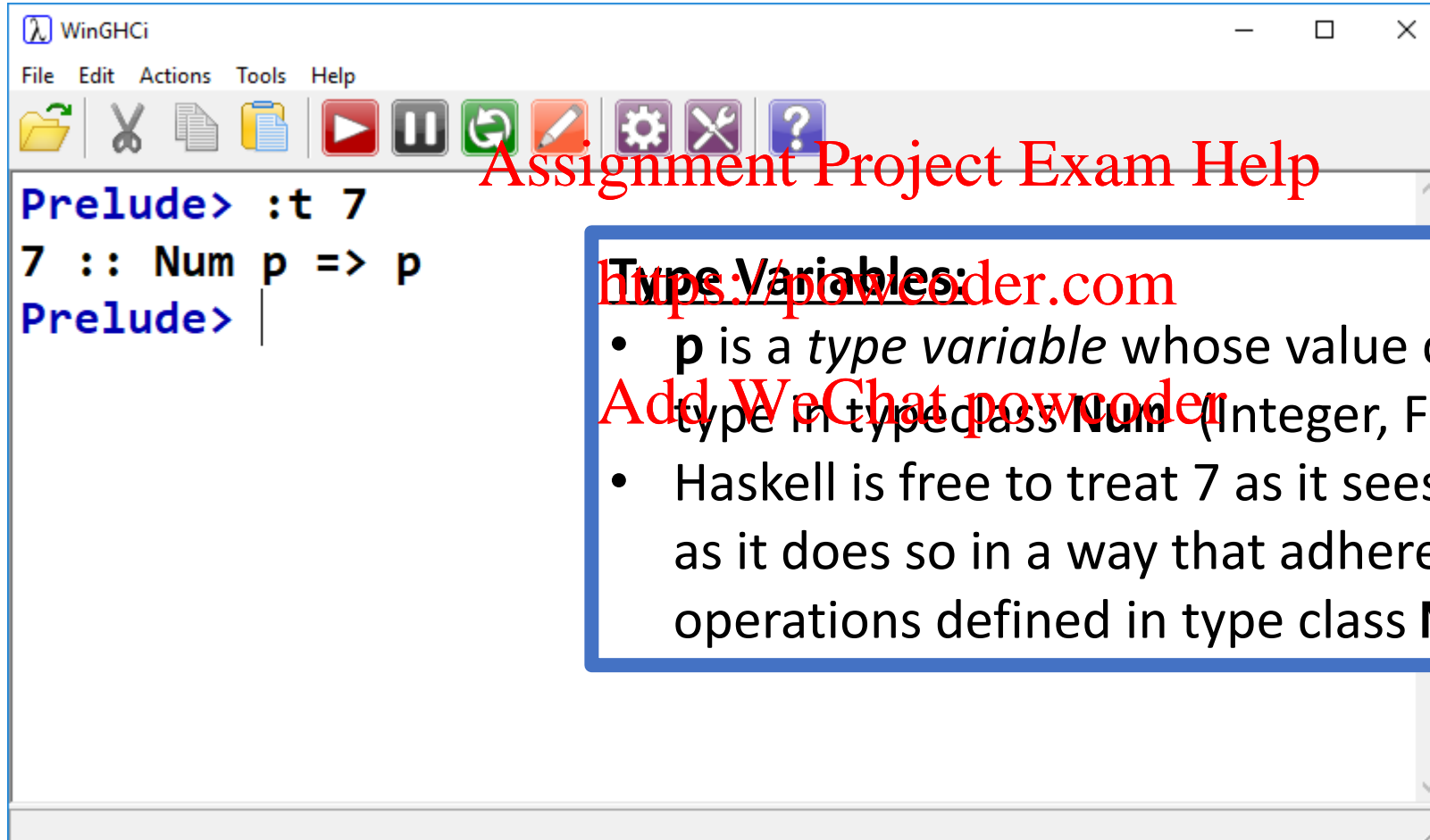
The screenshot shows the WinGHCi window with the following content:

```
WinGHCi
File Edit Actions Tools Help
[Icons]
Prelude> :t 1
1 :: Num p => p
Prelude> :t 1.0
1.0 :: Fractional p => p
Prelude> :t 'a'
'a' :: Char
Prelude> :t "Hello"
"Hello" :: [Char]
Prelude> :t 1 > 2
1 > 2 :: Bool
Prelude>
```

Annotations on the image:

- A red watermark "Assignment Project Exam Help" is overlaid on the top right of the window.
- A red bracket groups the first two queries, with a red text box containing:
 - <https://powcoder.com>
 - 1 is instance of **Num** type class.
 - 1.0 is instance of **Fractional** type class.
- A green bracket groups the last three queries, with a green text box containing:
 - 'a' is a **Char**
 - "Hello" is a **[Char]**
 - **[Char]** = **String**
 - `t` is a **Bool**

Num p => p ?

A screenshot of the WinGHCi Haskell interpreter window. The window has a menu bar with 'File', 'Edit', 'Actions', 'Tools', and 'Help'. Below the menu is a toolbar with icons for file operations and execution. The main text area shows the following interaction:

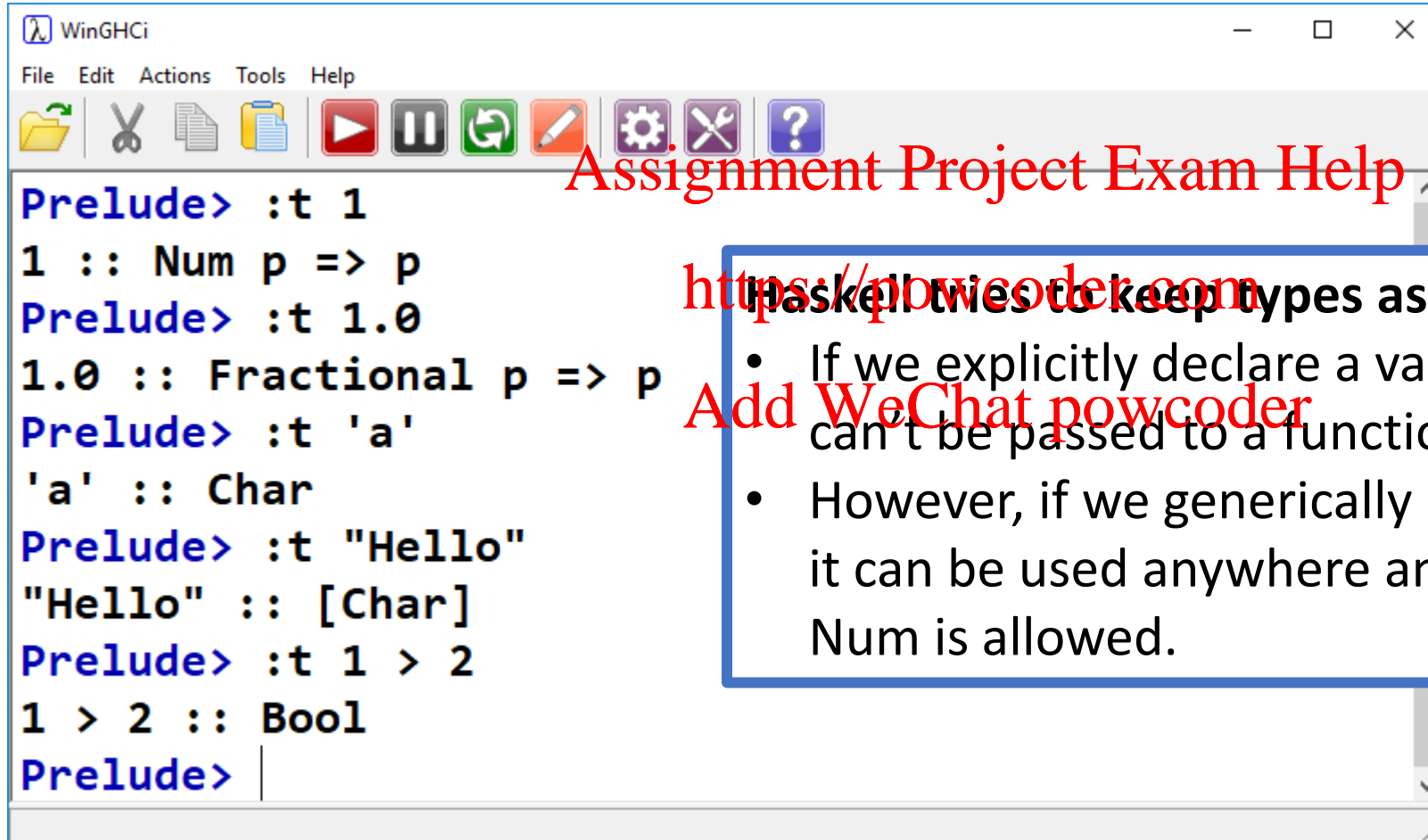
```
Prelude> :t 7
7 :: Num p => p
Prelude> |
```

Assignment Project Exam Help

Type Variables:


- **p** is a *type variable* whose value can be any type in typeclass **Num** (Integer, Float, etc.)
- Haskell is free to treat 7 as it sees fit, so long as it does so in a way that adheres to the operations defined in type class **Num**.

Typeclasses?



WinGHCi

File Edit Actions Tools Help



```
Prelude> :t 1
1 :: Num p => p
Prelude> :t 1.0
1.0 :: Fractional p => p
Prelude> :t 'a'
'a' :: Char
Prelude> :t "Hello"
"Hello" :: [Char]
Prelude> :t 1 > 2
1 > 2 :: Bool
Prelude> |
```

Assignment Project Exam Help

<https://powcoder.com>

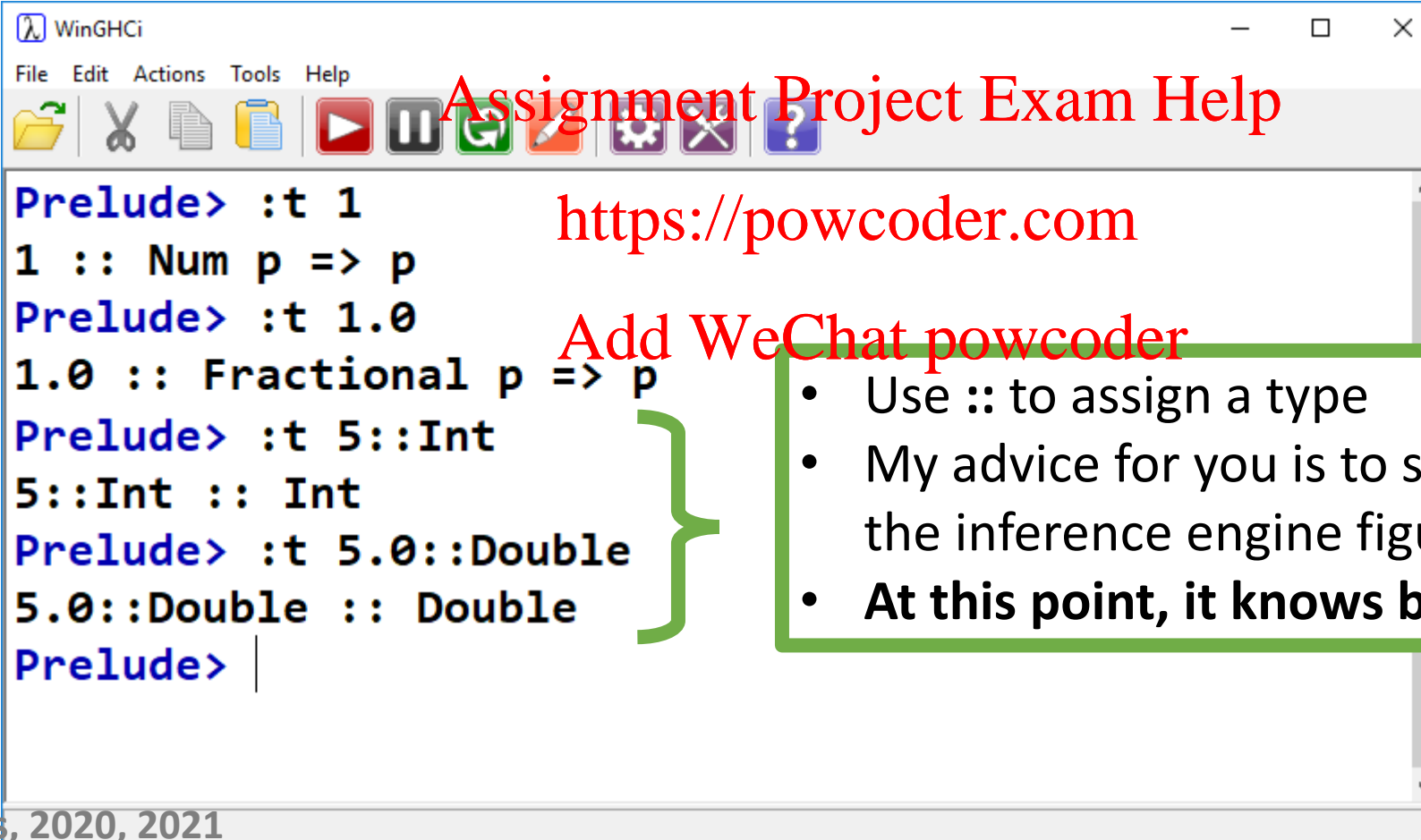
Add WeChat powcoder

Haskell tries to keep types as generic as possible

- If we explicitly declare a variable as integer, it can't be passed to a function requiring float.
- However, if we generically infer it to be a **Num**, it can be used anywhere any other member of Num is allowed.

Types in Haskell

We can explicitly indicate types:



The screenshot shows the WinGHCi window with the following interactions:

```
Prelude> :t 1
1 :: Num p => p
Prelude> :t 1.0
1.0 :: Fractional p => p
Prelude> :t 5::Int
5::Int :: Int
Prelude> :t 5.0::Double
5.0::Double :: Double
Prelude> |
```

Red text overlays on the image include:

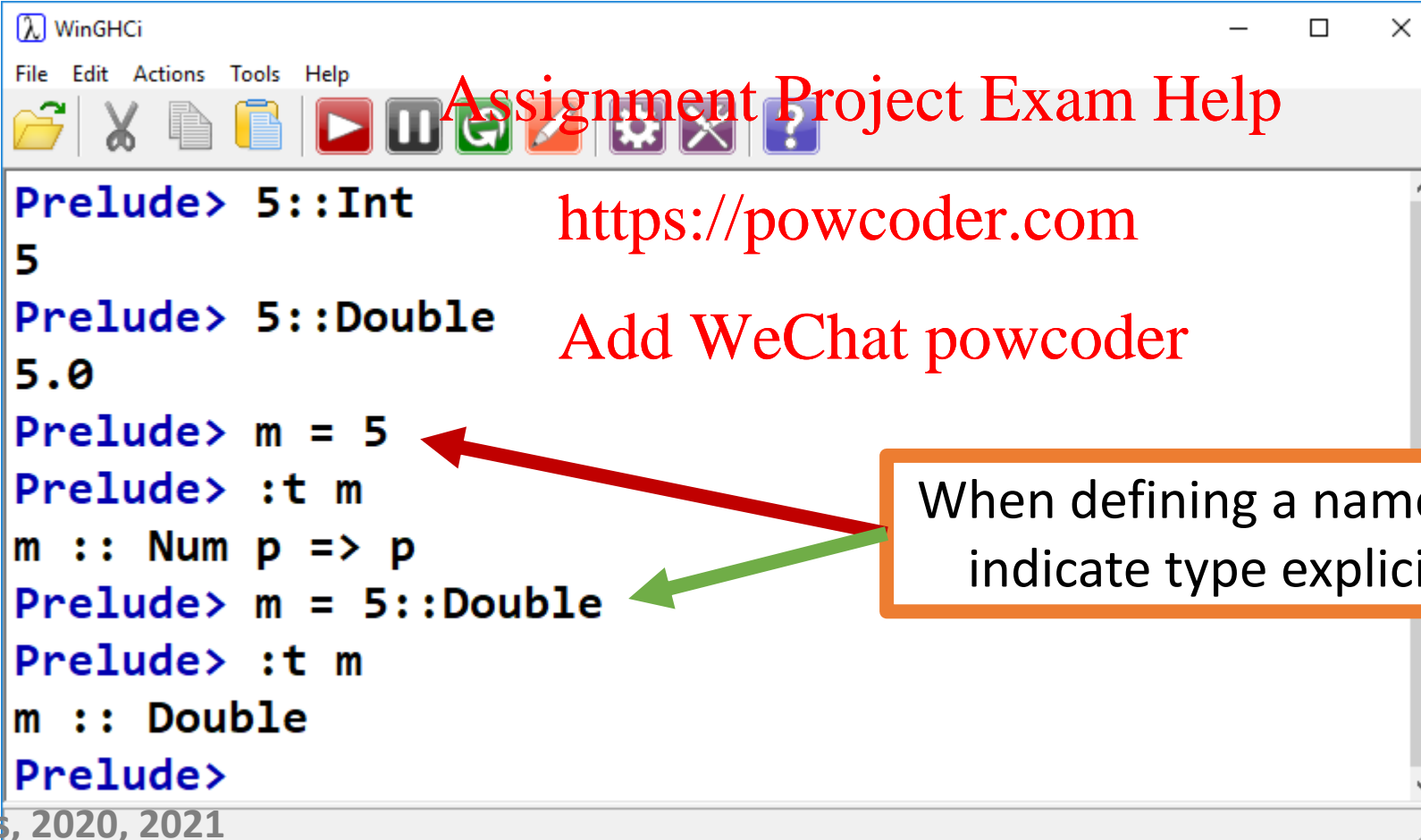
- Assignment Project Exam Help
- <https://powcoder.com>
- Add WeChat powcoder

A green box on the right contains the following advice:

- Use `::` to assign a type
- My advice for you is to start by letting the inference engine figure it out.
- **At this point, it knows better than you.**

Types in Haskell

We can explicitly indicate types:



The screenshot shows the WinGHCi window with the following interactions:

```
Prelude> 5::Int
5
Prelude> 5::Double
5.0
Prelude> m = 5
Prelude> :t m
m :: Num p => p
Prelude> m = 5::Double
Prelude> :t m
m :: Double
Prelude>
```

Annotations and links:

- Red text: <https://powcoder.com>
- Red text: Add WeChat powcoder
- Orange box: When defining a name, can indicate type explicitly:

Arrows point from the orange box to the lines `m = 5` and `m = 5::Double`.

Type Classes

Type polymorphism and type variables:

Recall: Overloading

- In languages like C++, the `==` operator is overloaded to work with many different types.
- Numeric type equality and string equality are performed differently.
- In general, if we want to compare two values of type α , we use an **α -compare**
- α is a *type variable*, because its value is a type.

Type Classes

Consider the equality (==) operator:

Takes two parameters, each of the same type (call it α), and returns a Boolean

<https://powcoder.com>

This operator may not be defined for *all* types, just some.

Add WeChat powcoder

Thus, we can associate == with a specific ***type class*** containing those types for which == is defined.

This type class is called **E_q** in Haskell.

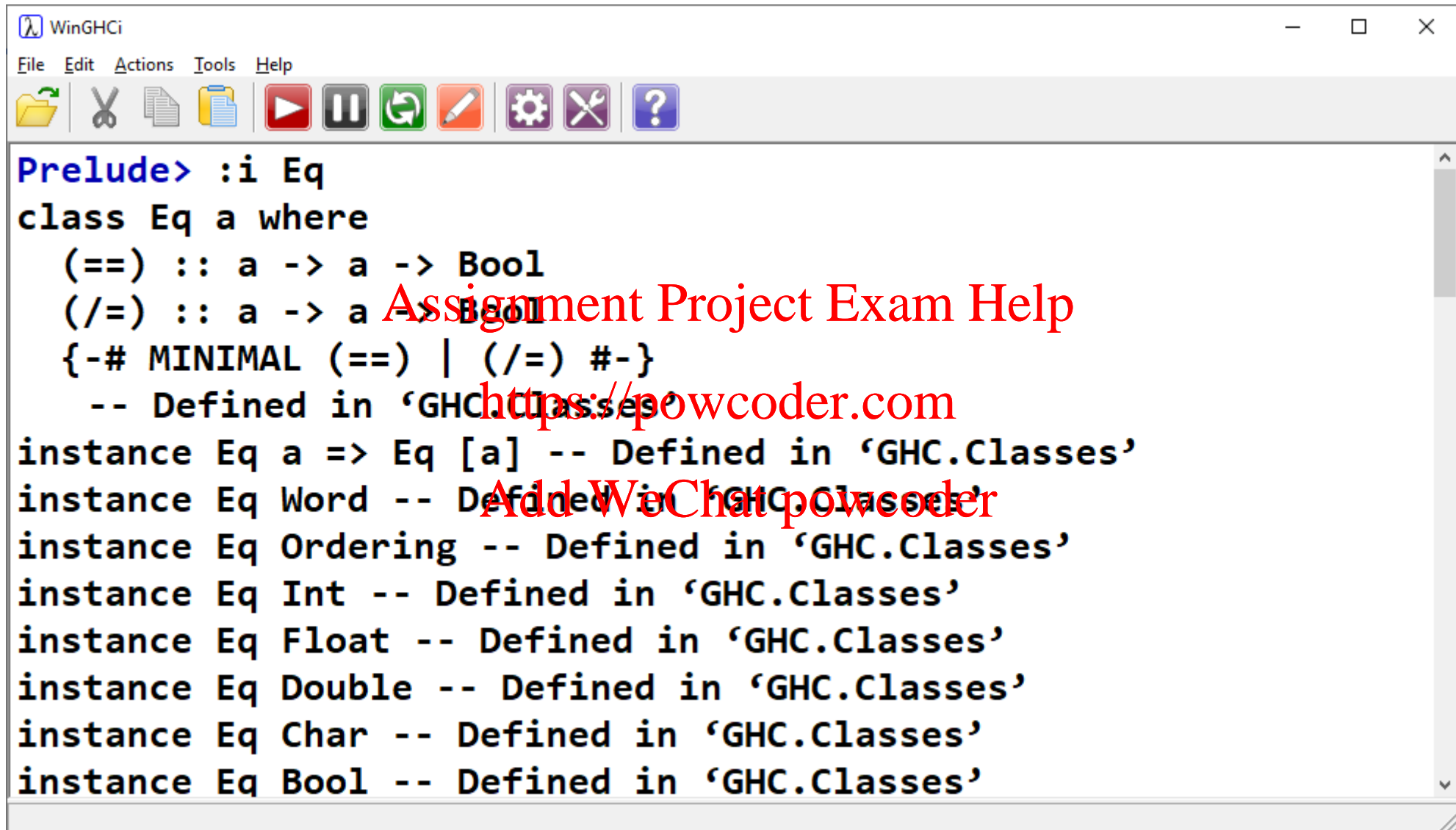
Eq Type Class

`(==)` is defined for types
in typeclass **Eq**

`(==) :: Eq a => a -> a -> Bool`

- `(==)` takes two args of type **a**, where **a** is a member of type class **Eq**
- It returns **Bool**

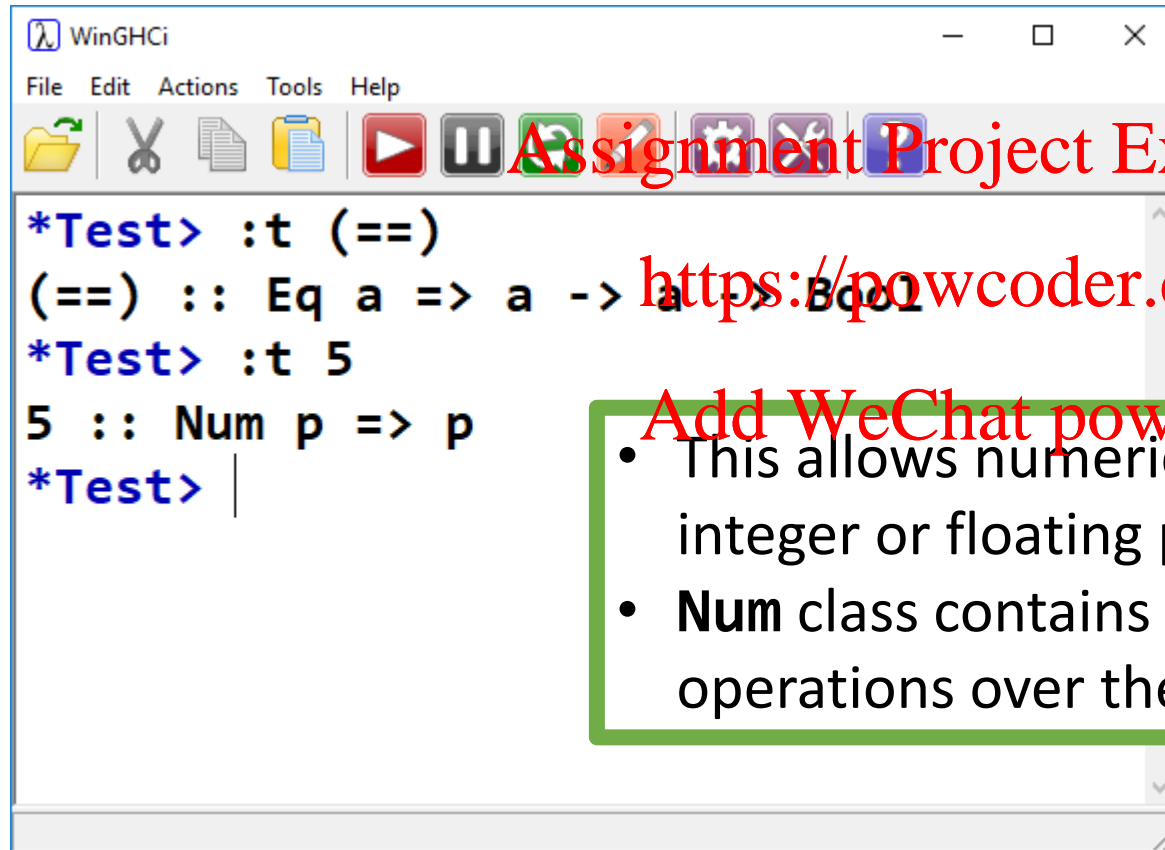
- If a concrete type, **a**, belongs to a certain type class, we say **a** is an *instance* of that type class.
- **Int** is an instance of **Eq**, for example.



The image shows a screenshot of the WinGHCi window. The title bar says 'WinGHCi'. The menu bar includes 'File', 'Edit', 'Actions', 'Tools', and 'Help'. The toolbar contains icons for file operations (copy, paste, save, undo, redo), execution (run, pause, stop), and settings (preferences, help). The main text area displays the following Haskell code:

```
Prelude> :i Eq
class Eq a where
  (==) :: a -> a -> Bool
  (/=) :: a -> a -> Bool
  {-# MINIMAL (==) | (/=) #-}
  -- Defined in 'GHC.Classes'
instance Eq a => Eq [a] -- Defined in 'GHC.Classes'
instance Eq Word -- Defined in 'GHC.Classes'
instance Eq Ordering -- Defined in 'GHC.Classes'
instance Eq Int -- Defined in 'GHC.Classes'
instance Eq Float -- Defined in 'GHC.Classes'
instance Eq Double -- Defined in 'GHC.Classes'
instance Eq Char -- Defined in 'GHC.Classes'
instance Eq Bool -- Defined in 'GHC.Classes'
```

Num Type Class



```
WinGHCi
File Edit Actions Tools Help
*Test> :t (==)
(==) :: Eq a => a -> a -> Bool
*Test> :t 5
5 :: Num p => p
*Test> |
```

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

- This allows numeric values freedom to be an integer or floating point as the compiler sees fit.
- **Num** class contains all numbers, and certain operations over them such as addition.

Num Type Class

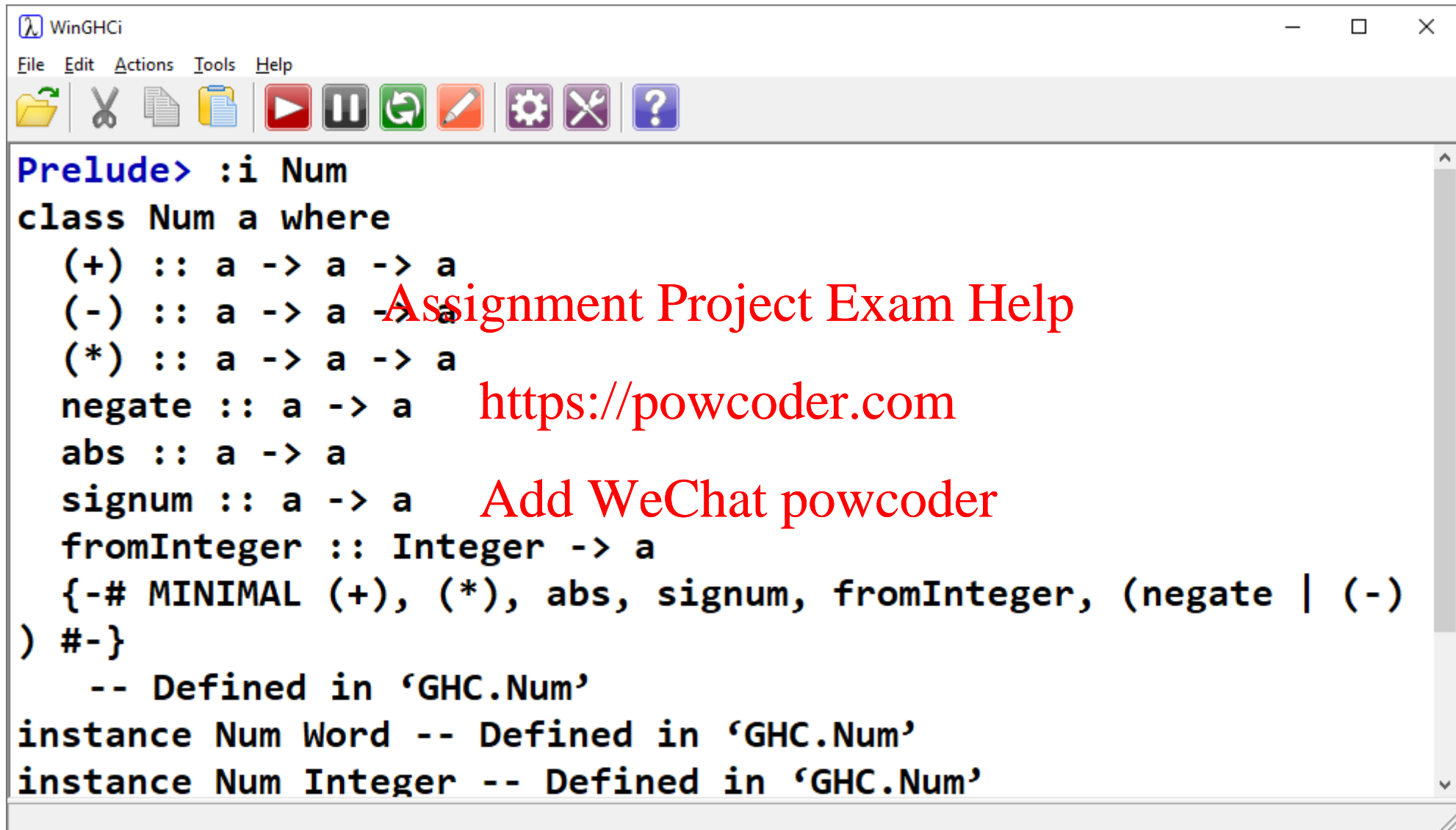
The screenshot shows the WinGHCi window with the following text:

```
*Test> :t (==)
(==) :: Eq a => a -> a -> Bool
*Test> :t 5
5 :: Num p => p
*Test>
```

Red text overlays the image: "Assignment Project Exam Help" and "https://powcoder.com".

Add WeChat powcoder

- **p** is a type variable
- The type of 5 is **p**, and **p** is a member of type class **Num**

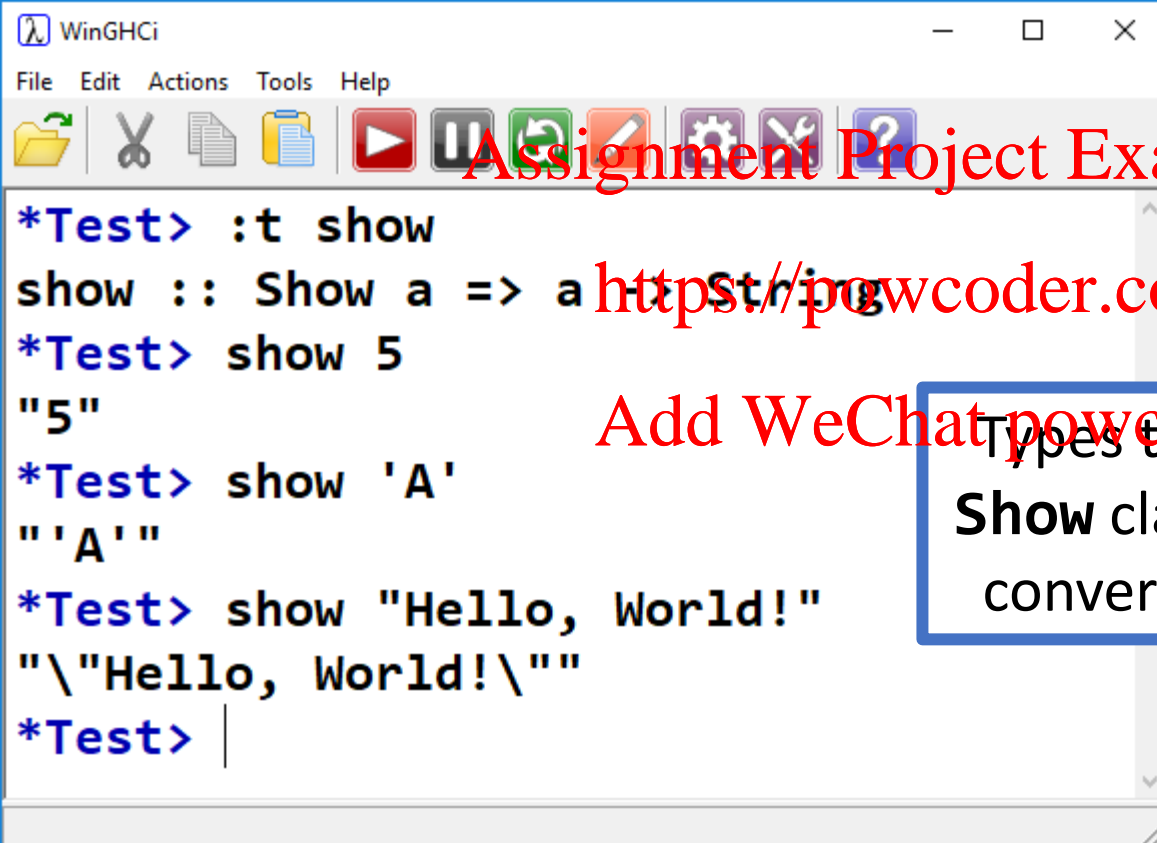


The image shows a screenshot of a Haskell interpreter window titled "WinGHCi". The window has a menu bar with "File", "Edit", "Actions", "Tools", and "Help". Below the menu bar is a toolbar with icons for file operations (copy, paste, save), execution (run, pause, step), and settings (gear, wrench, question mark). The main text area contains the following Haskell code:

```
Prelude> :i Num
class Num a where
  (+) :: a -> a -> a
  (-) :: a -> a -> a
  (*) :: a -> a -> a
  negate :: a -> a
  abs :: a -> a
  signum :: a -> a
  fromInteger :: Integer -> a
  {-# MINIMAL (+), (*), abs, signum, fromInteger, (negate | (-)
) #-}
  -- Defined in 'GHC.Num'
instance Num Word -- Defined in 'GHC.Num'
instance Num Integer -- Defined in 'GHC.Num'
```

Overlaid on the right side of the code is red text that reads: "Assignment Project Exam Help", "https://powcoder.com", and "Add WeChat powcoder".

Show Type Class



A screenshot of the WinGHCi terminal window. The window has a menu bar with 'File', 'Edit', 'Actions', 'Tools', and 'Help'. Below the menu bar is a toolbar with icons for file operations and execution. The terminal shows the following interactions:

```
*Test> :t show
show :: Show a => a -> String
*Test> show 5
"5"
*Test> show 'A'
"'A'"
*Test> show "Hello, World!"
 "\"Hello, World!\""
*Test> |
```

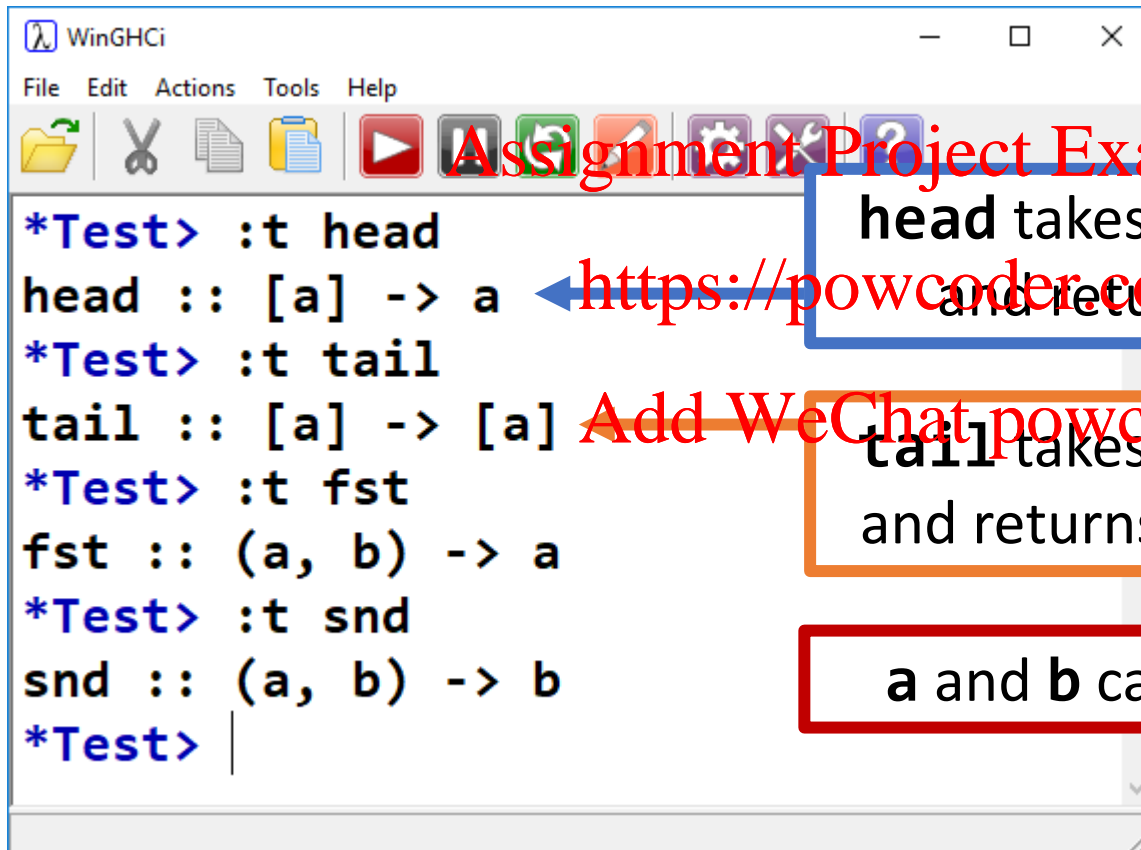
Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

Types that are members of the **Show** class have functions which convert their value to a String.

Function Types



```
WinGHCi
File Edit Actions Tools Help
*Test> :t head
head :: [a] -> a
*Test> :t tail
tail :: [a] -> [a]
*Test> :t fst
fst :: (a, b) -> a
*Test> :t snd
snd :: (a, b) -> b
*Test> |
```

Assignment Project Exam Help

<https://powcoder.com>

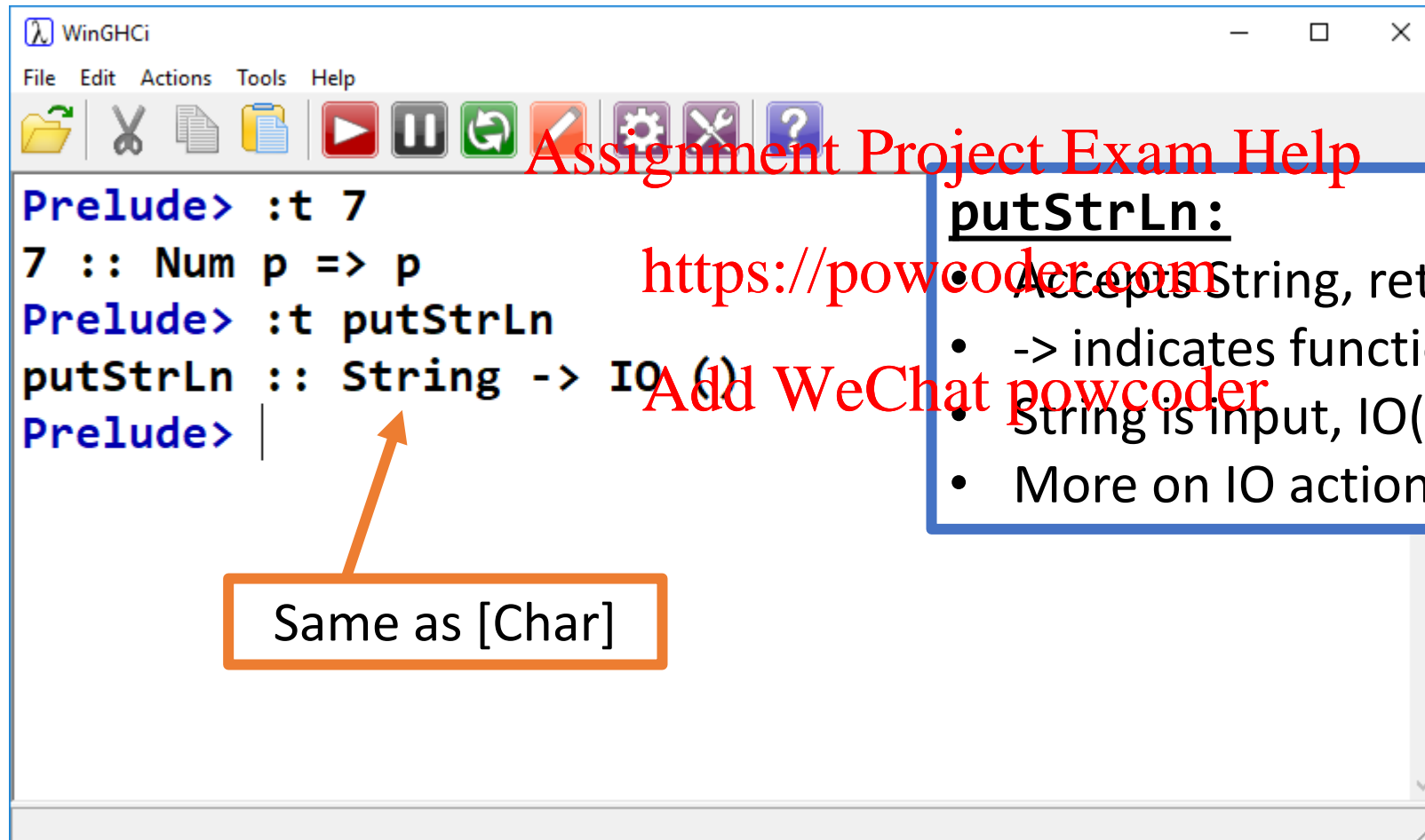
Add WeChat powcoder

head takes a list containing type **a**,
and returns a value of type **a**

tail takes a list containing type **a**,
and returns a list containing type **a**

a and **b** can be *literally any type*!

Function Types



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

Prelude> :t 7
7 :: Num p => p
Prelude> :t putStrLn
putStrLn :: String -> IO ()
Prelude> |
```

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

putStrLn:

- Accepts String, returns **IO action**.
- -> indicates function
- String is input, IO() is output.
- More on IO actions later.

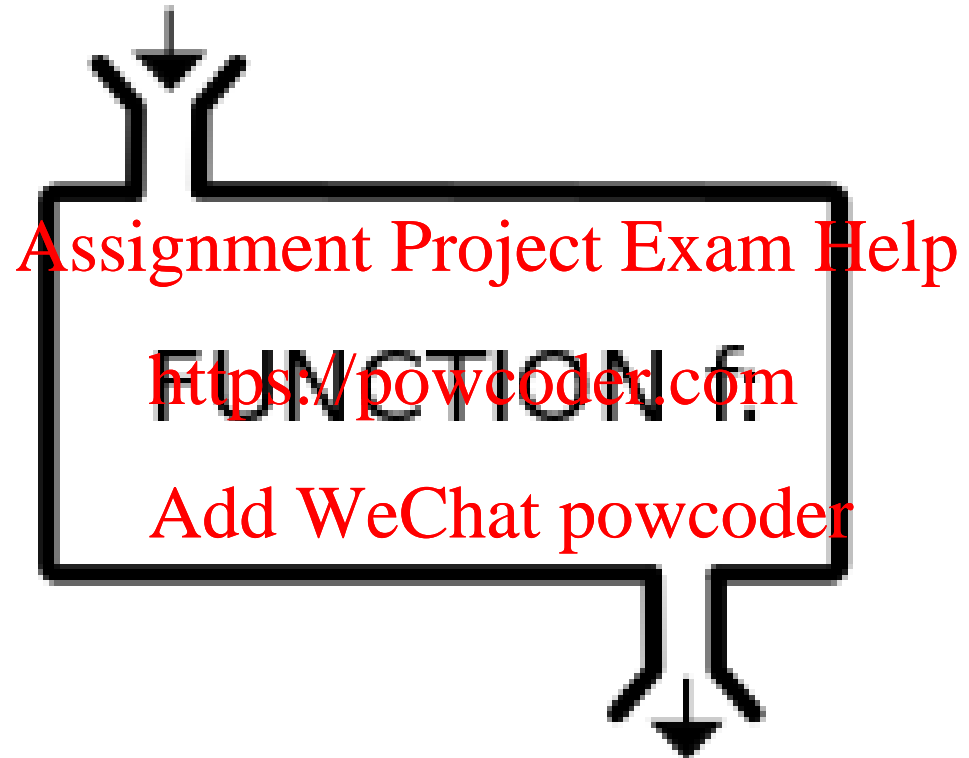
Same as [Char]

We'll create our own types soon, and see how to
add them to existing type classes.

<https://powcoder.com>

Add WeChat powcoder

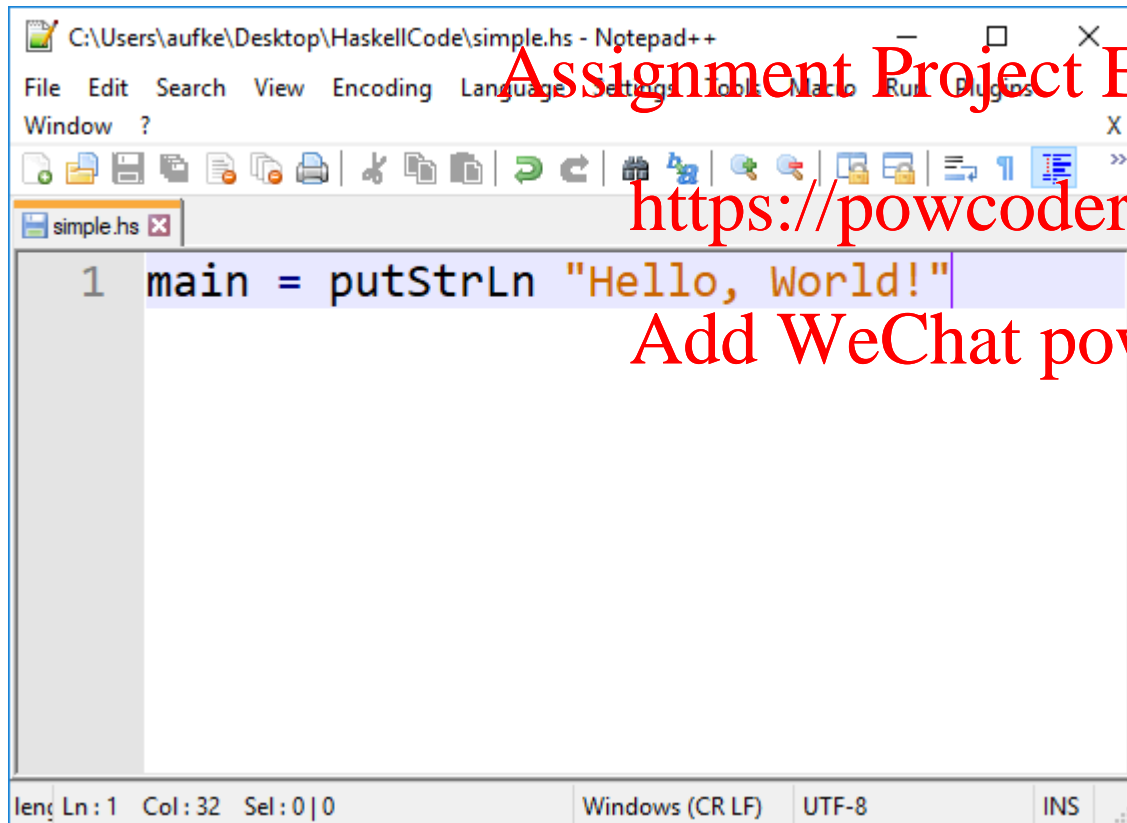
INPUT x



OUTPUT $f(x)$

Functions in Haskell

As expected of a pure functional language, functions are central in Haskell



```
1 main = putStrLn "Hello, World!"
```

Assignment Project Exam Help

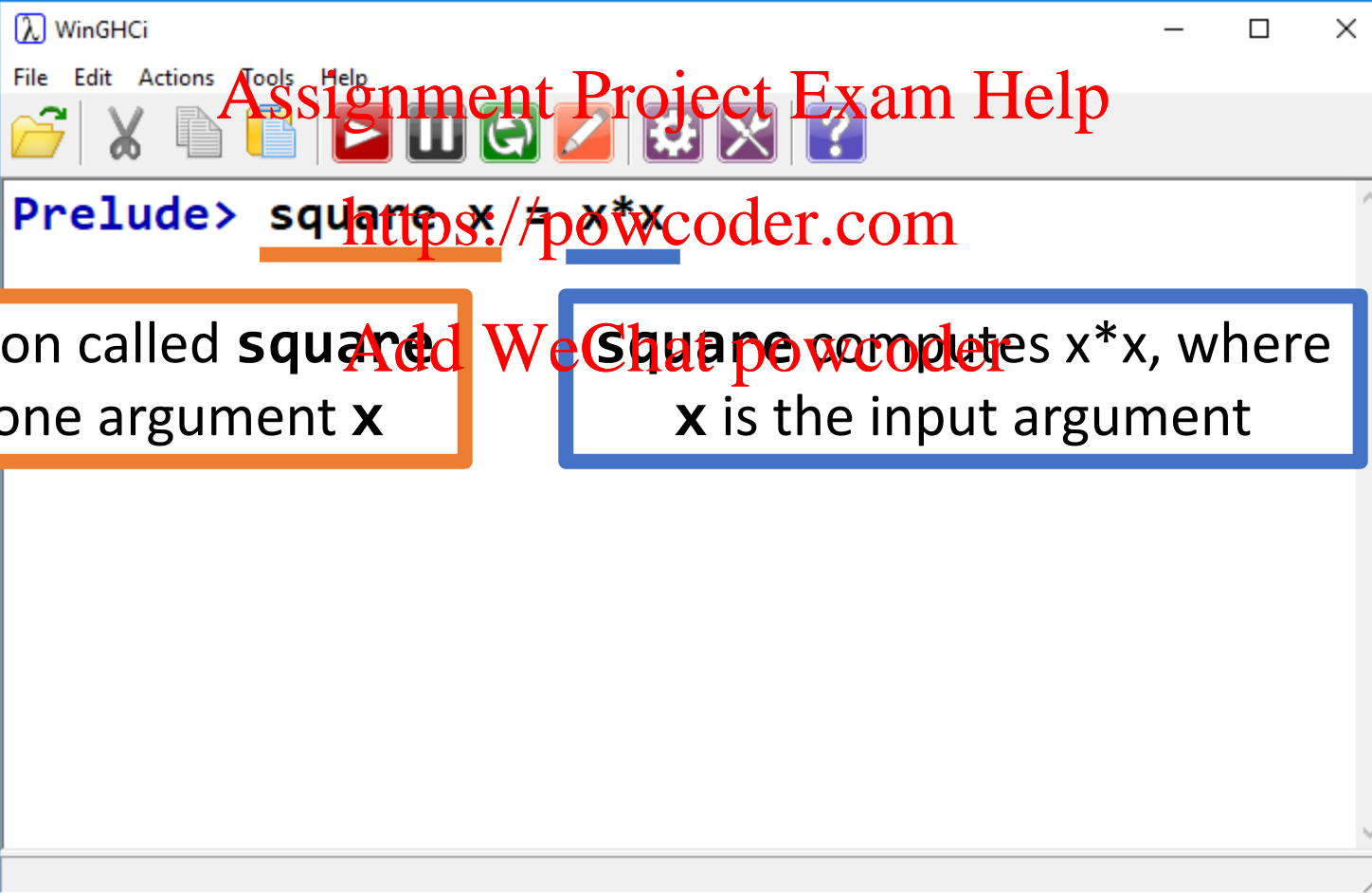
<https://powcoder.com>

Add WeChat powcoder

- If we're compiling our code into an executable, we need a main.
- If we're using the GHCi shell, we don't.

Functions in Haskell

Let's start simple:



The image shows a screenshot of the WinGHCi Haskell interpreter window. The window has a menu bar with 'File', 'Edit', 'Actions', 'Tools', and 'Help'. Below the menu bar is a toolbar with various icons. The main text area contains the following code:

```
Prelude> square x = x*x
```

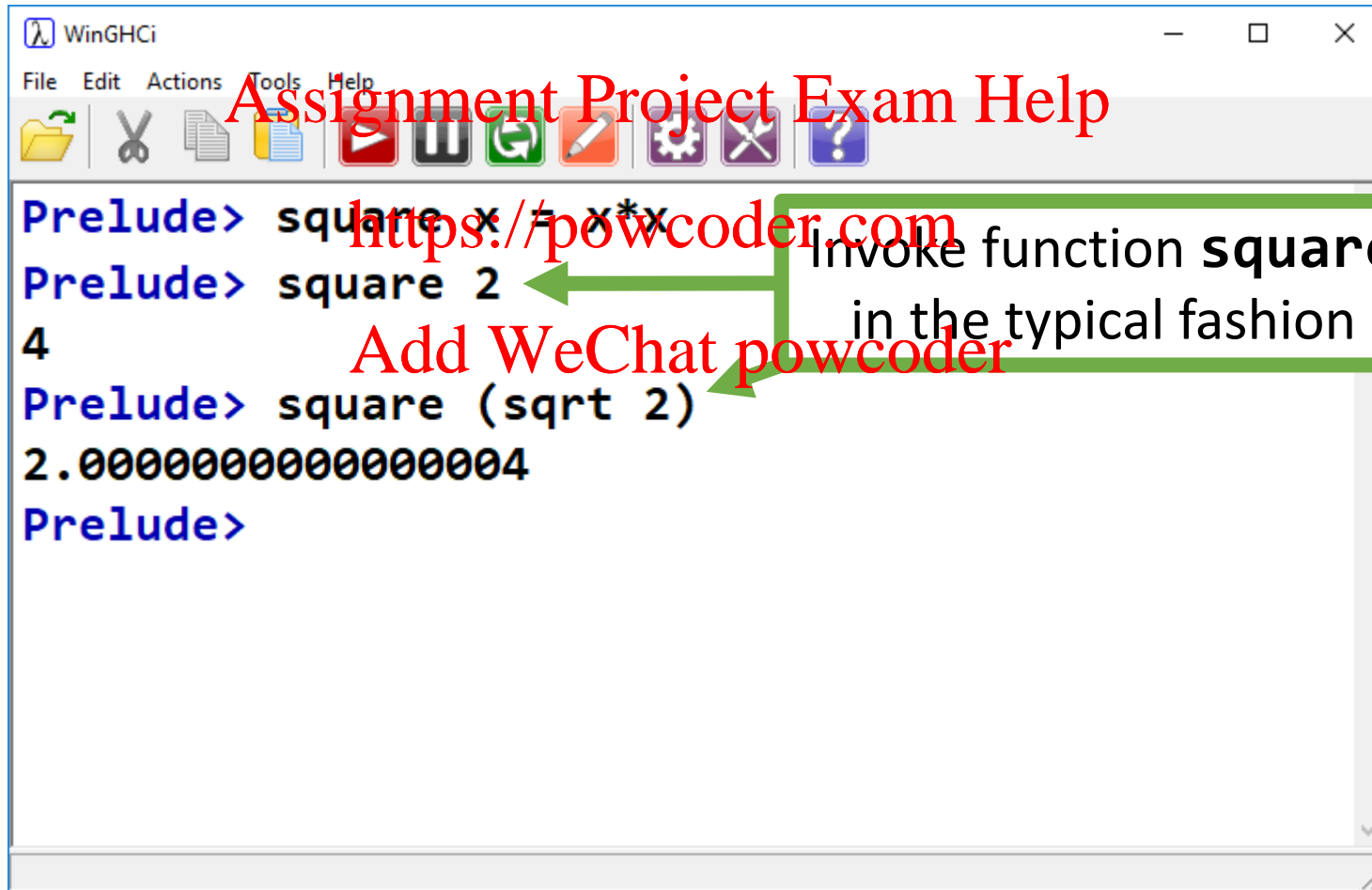
There are two annotations overlaid on the image:

- An orange-bordered box on the left contains the text: "Define function called **square** that takes one argument **x**".
- A blue-bordered box on the right contains the text: "Square computes $x*x$, where **x** is the input argument".

Red text is overlaid across the center of the image, reading: "Assignment Project Exam Help" and "https://powcoder.com".

Functions in Haskell

Let's start simple:



The screenshot shows the WinGHCi window with a menu bar (File, Edit, Actions, Tools, Help) and a toolbar. The command line contains the following text:

```
Prelude> square x = x*x
Prelude> square 2
4
Prelude> square (sqrt 2)
2.0000000000000004
Prelude>
```

Annotations on the image include:

- A red watermark "Assignment Project Exam Help" and the URL "https://powcoder.com" are overlaid on the top half of the window.
- A green box on the right contains the text "Invoke function **square** in the typical fashion".
- Two green arrows point from this box to the `square 2` and `square (sqrt 2)` lines in the code.
- Red text "Add WeChat powcoder" is located below the first arrow.

Functions in Haskell

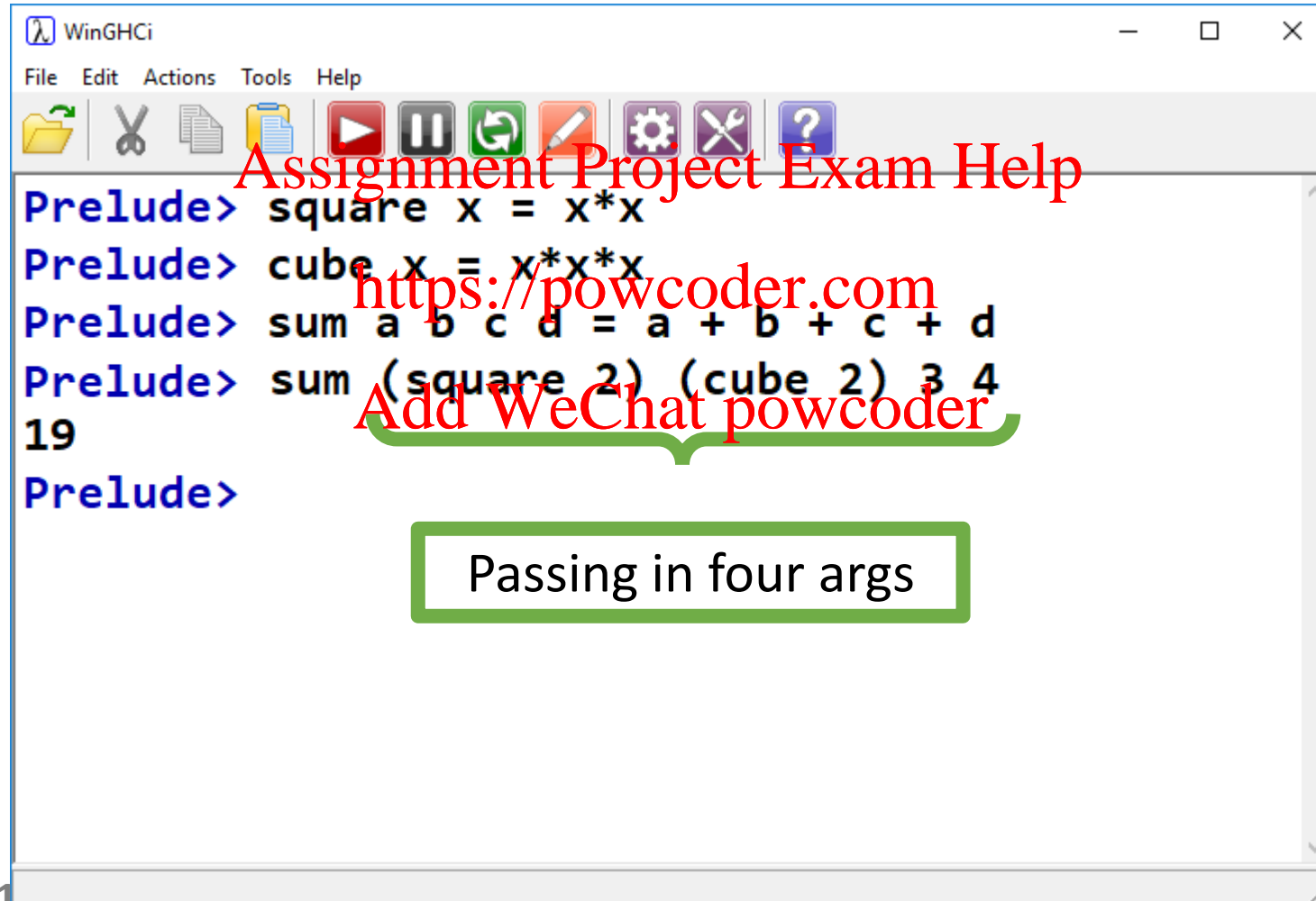
The image shows a screenshot of the WinGHCi Haskell interpreter window. The window has a menu bar (File, Edit, Actions, Tools, Help) and a toolbar with icons for file operations and execution. The main text area contains the following code:

```
Prelude> square x = x*x
Prelude> cube x = x*x*x
Prelude> sum a b c d = a + b + c + d
Prelude>
```

Annotations are present on the code:

- A red watermark "Assignment Project Exam Help" is at the top.
- A red watermark "https://powcoder.com" is in the middle.
- A red watermark "Add WeChat powcoder" is at the bottom.
- A blue box labeled "Function named **sum**" has an arrow pointing to the `sum` keyword.
- A green box labeled "Parameter list" is underlined under `a b c d`.
- An orange box labeled "Expression to evaluate" is underlined under `= a + b + c + d`.

Functions in Haskell



The screenshot shows the WinGHCi window with the following content:

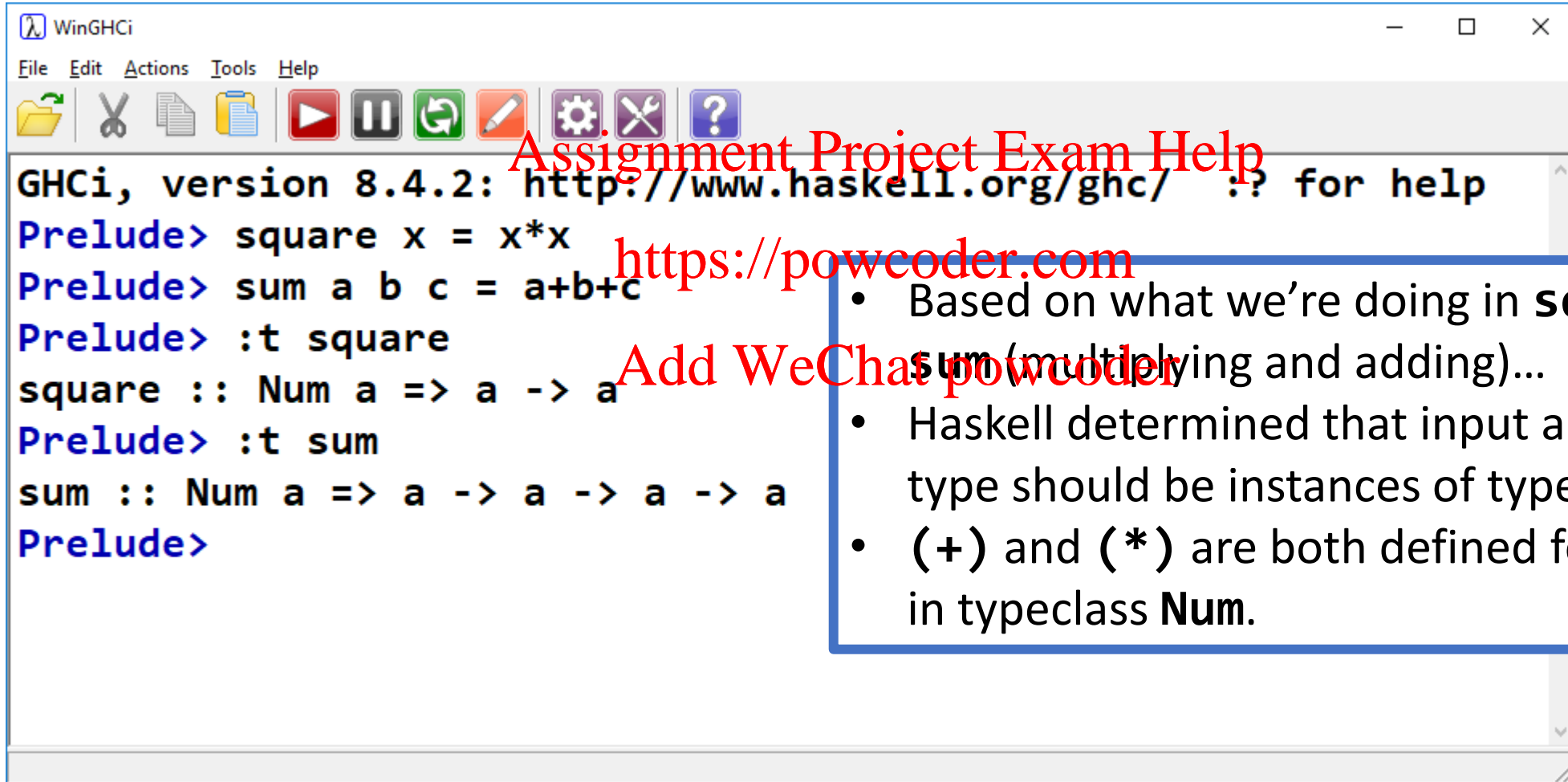
```
WinGHCi
File Edit Actions Tools Help
[Icons]
Prelude> square x = x*x
Prelude> cube x = x*x*x
Prelude> sum a b c d = a + b + c + d
Prelude> sum (square 2) (cube 2) 3 4
19
Prelude>
```

Overlaid on the screenshot are several red annotations:

- Assignment Project Exam Help** (top)
- <https://powcoder.com> (middle)
- Add WeChat powcoder** (bottom, underlined with a green bracket)

A green-bordered box at the bottom contains the text: **Passing in four args**

Functions in Haskell



The screenshot shows the WinGHCi window with the following content:

```
WinGHCi
File Edit Actions Tools Help
[Icons]
GHCi, version 8.4.2: http://www.haskell.org/ghc/  :? for help
Prelude> square x = x*x
Prelude> sum a b c = a+b+c
Prelude> :t square
square :: Num a => a -> a
Prelude> :t sum
sum :: Num a => a -> a -> a -> a
Prelude>
```

Assignment Project Exam Help

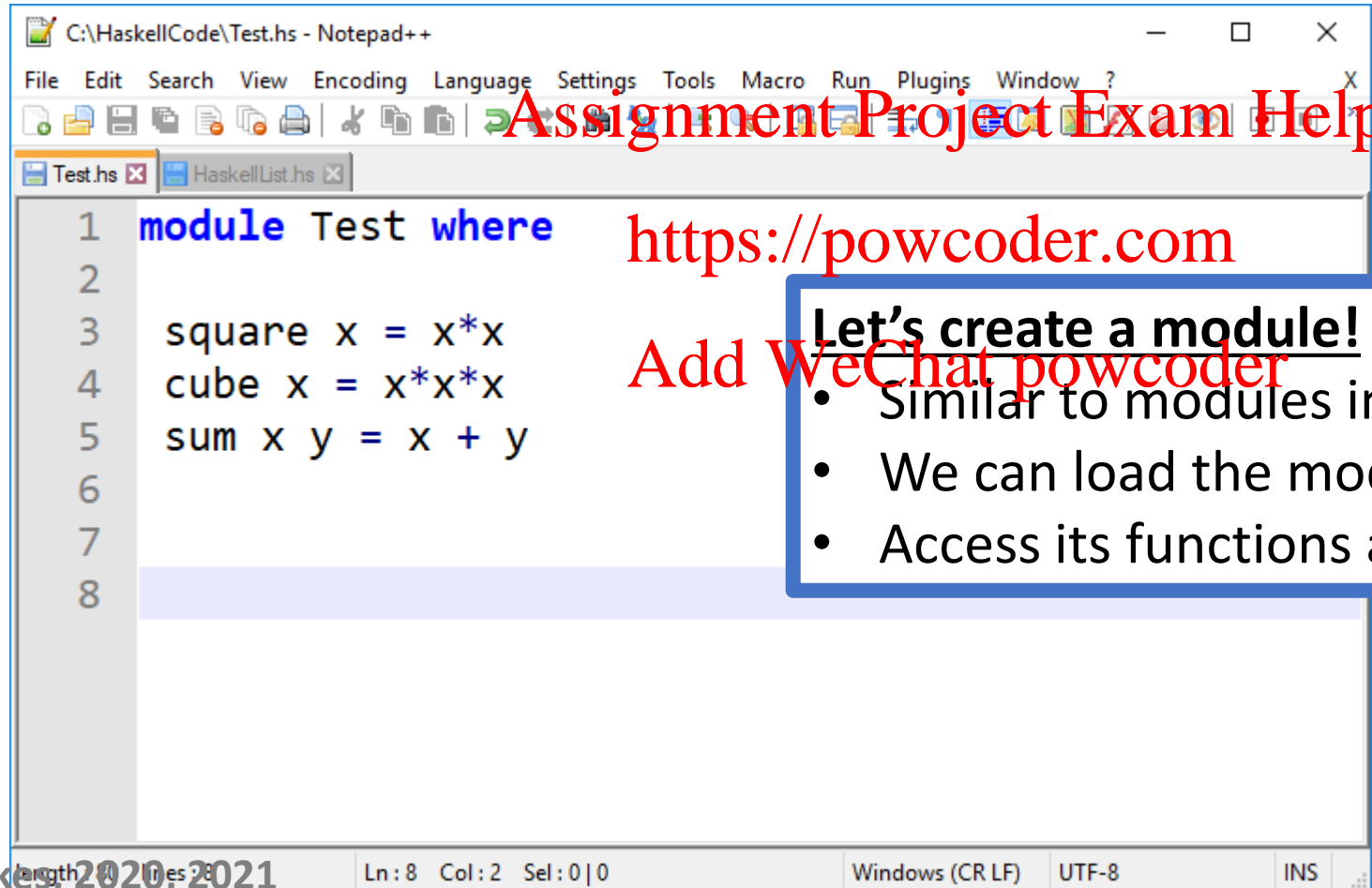
<https://powcoder.com>

Add WeChat powcoder

- Based on what we're doing in **square** and **sum** (multiplying and adding)...
- Haskell determined that input and output type should be instances of typeclass **Num**.
- **(+)** and **(*)** are both defined for all types in typeclass **Num**.

Haskell Modules

This is getting tedious to type interactively.



The screenshot shows a Notepad++ window titled 'C:\HaskellCode\Test.hs - Notepad++'. The menu bar includes File, Edit, Search, View, Encoding, Language, Settings, Tools, Macro, Run, Plugins, and Window. The toolbar contains various icons for file operations and editing. The code editor shows the following Haskell code:

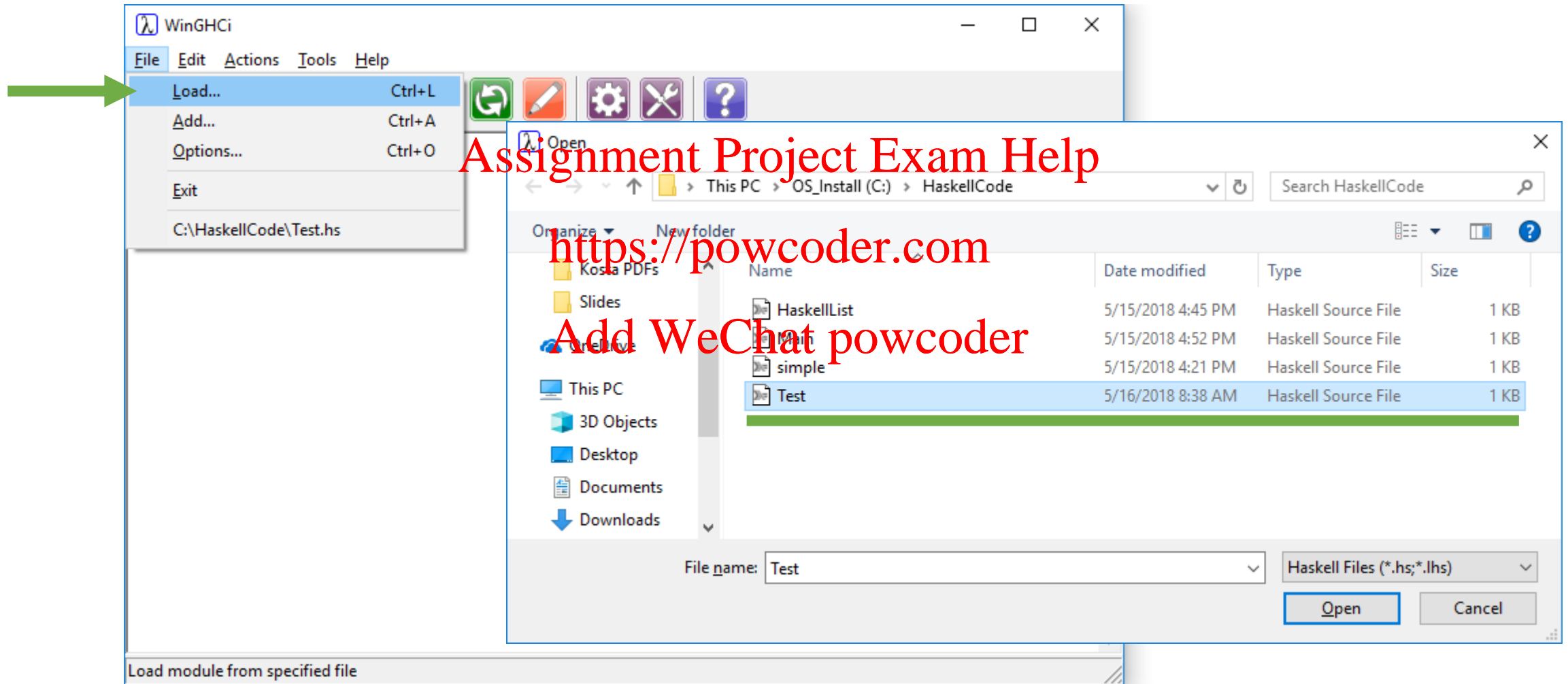
```
1 module Test where
2
3   square x = x*x
4   cube x = x*x*x
5   sum x y = x + y
6
7
8
```

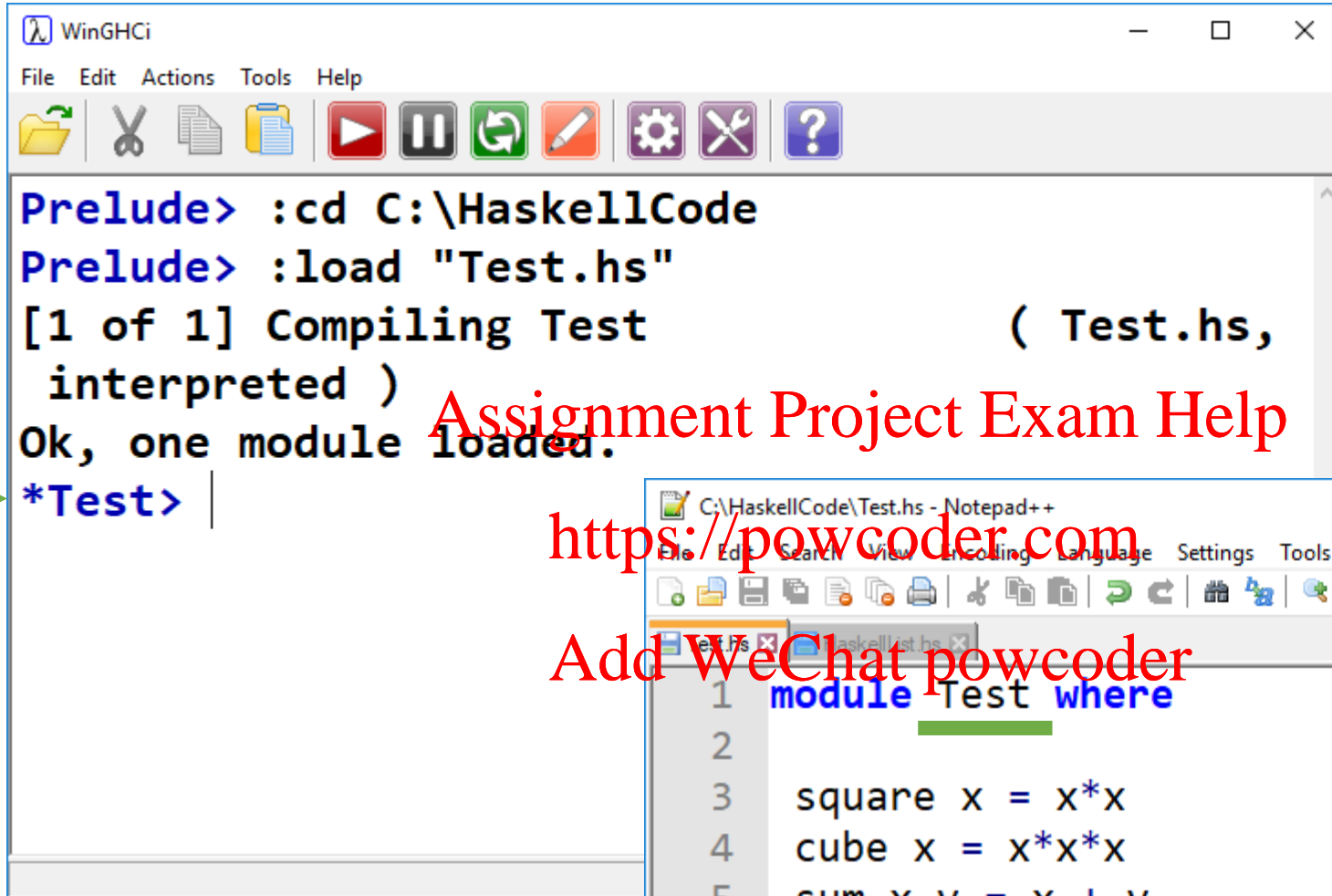
Overlaid on the screenshot is a red watermark that reads 'Assignment Project Exam Help' and a red URL 'https://powcoder.com'. A blue-bordered box on the right side of the code editor contains the text 'Let's create a module!' followed by a bulleted list.

- Similar to modules in Elixir
- We can load the module in GHCi
- Access its functions and expressions

The status bar at the bottom of the Notepad++ window shows 'Ln: 8 Col: 2 Sel: 0 | 0', 'Windows (CR LF)', 'UTF-8', and 'INS'.

Loading a Module



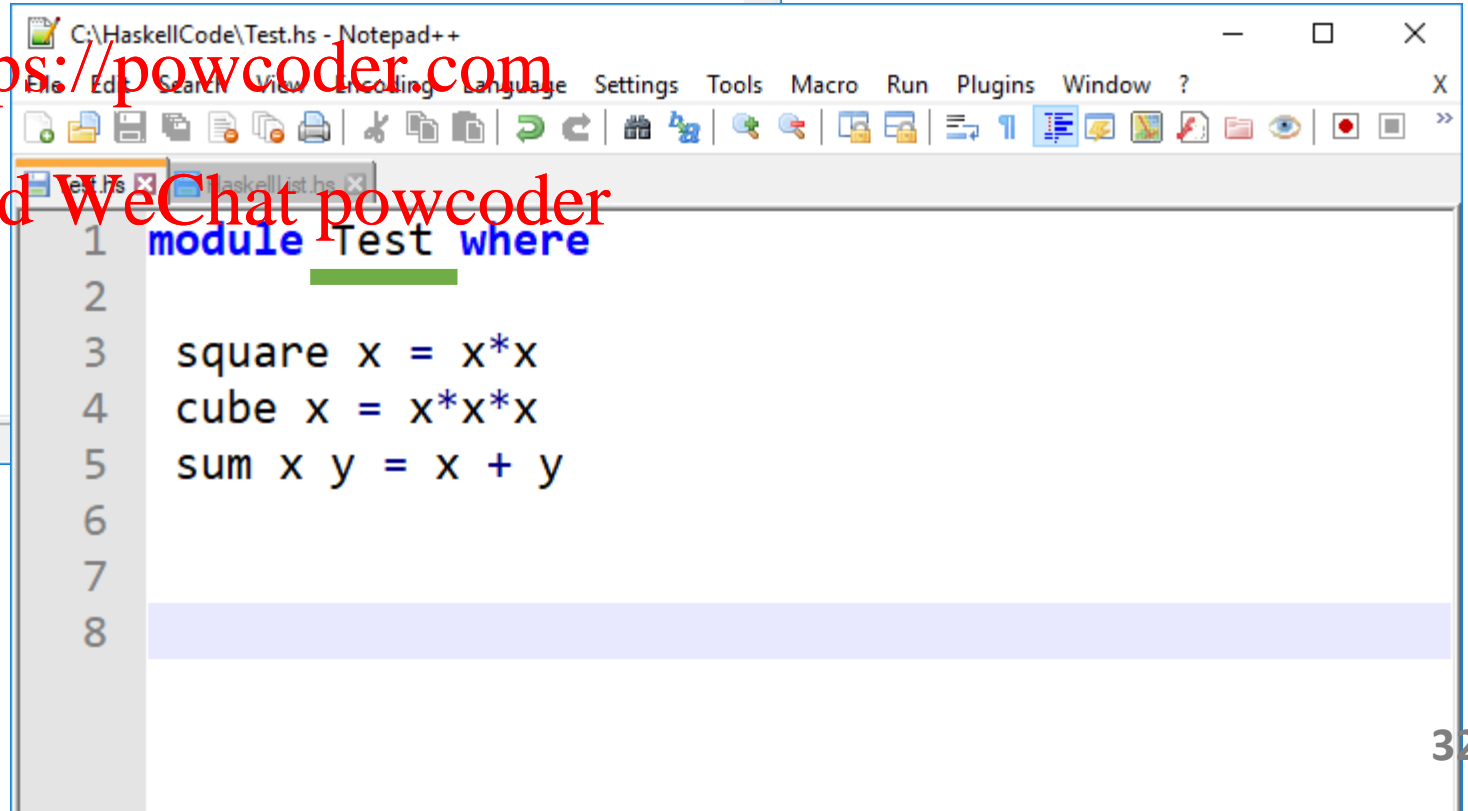
A screenshot of the WinGHCi terminal window. The window has a menu bar with 'File', 'Edit', 'Actions', 'Tools', and 'Help'. Below the menu is a toolbar with icons for file operations (open, save, copy, paste), execution (run, pause, refresh), and settings (gear, wrench, question mark). The terminal text shows the user navigating to 'C:\HaskellCode', loading 'Test.hs', and successfully compiling and interpreting it. The prompt is now '*Test>'. A green arrow points to this prompt from the left.

```
Prelude> :cd C:\HaskellCode
Prelude> :load "Test.hs"
[1 of 1] Compiling Test                ( Test.hs,
interpreted )
Ok, one module loaded.
*Test> |
```

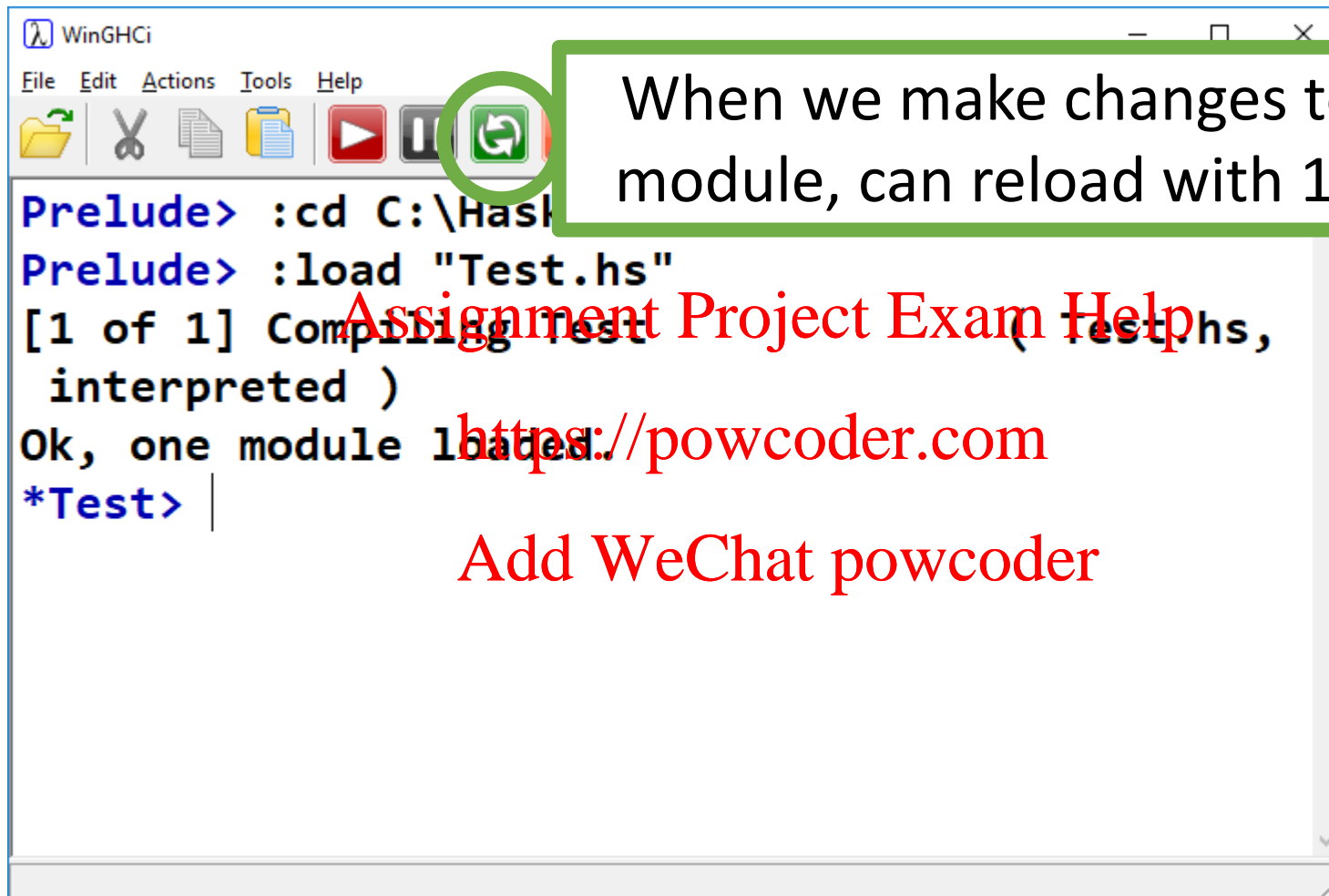
Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

A screenshot of the Notepad++ editor window showing the contents of 'C:\HaskellCode\Test.hs'. The window has a menu bar with 'File', 'Edit', 'Search', 'View', 'Encoding', 'Language', 'Settings', 'Tools', 'Macro', 'Run', 'Plugins', 'Window', and '?'. The toolbar includes icons for file operations, editing, and development. The code is as follows:

```
1 module Test where
2
3 square x = x*x
4 cube x = x*x*x
5 sum x y = x + y
6
7
8
```

When we make changes to Test module, can reload with 1 click!

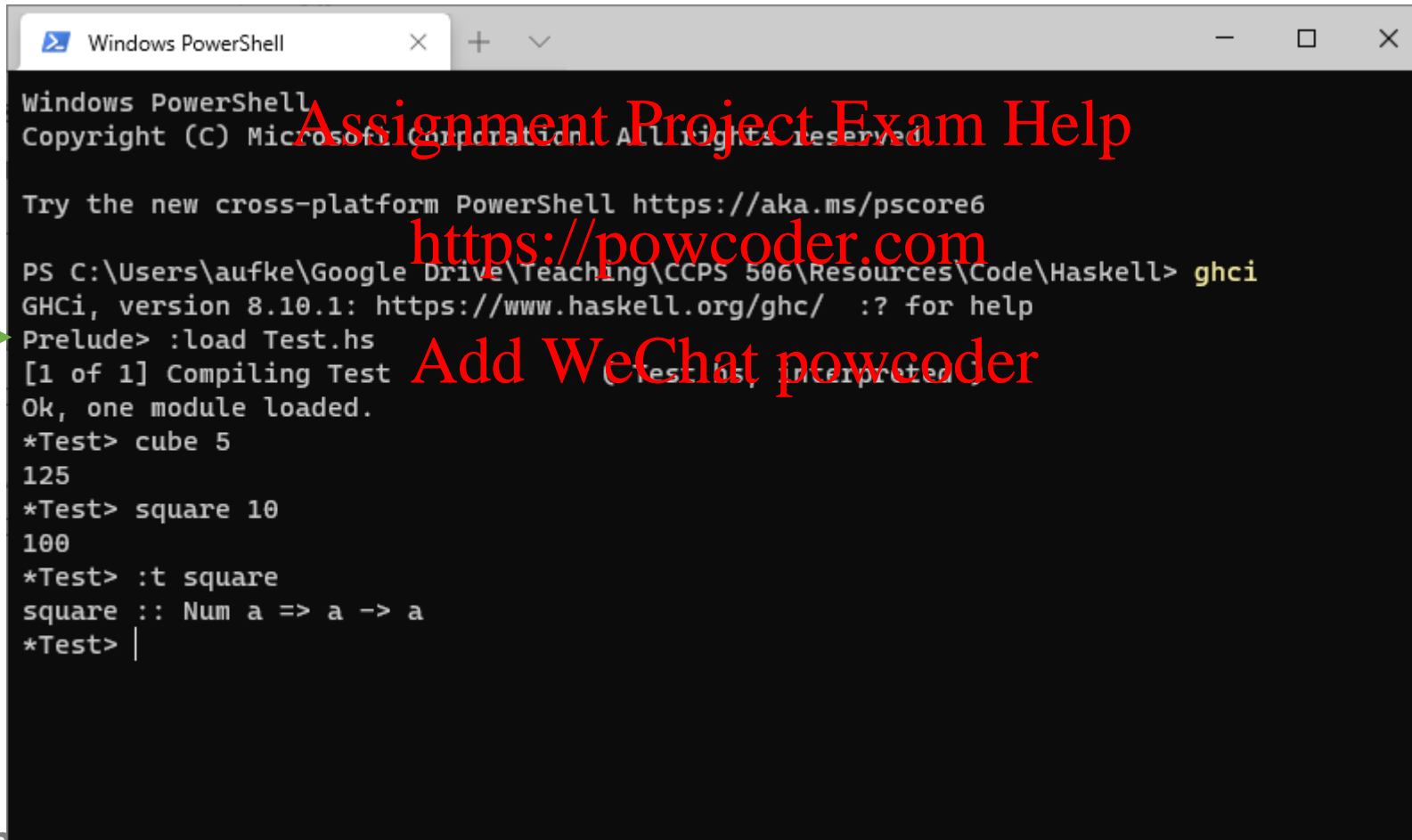
Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

Loading a Module

Use `:load` in terminal GHCi:



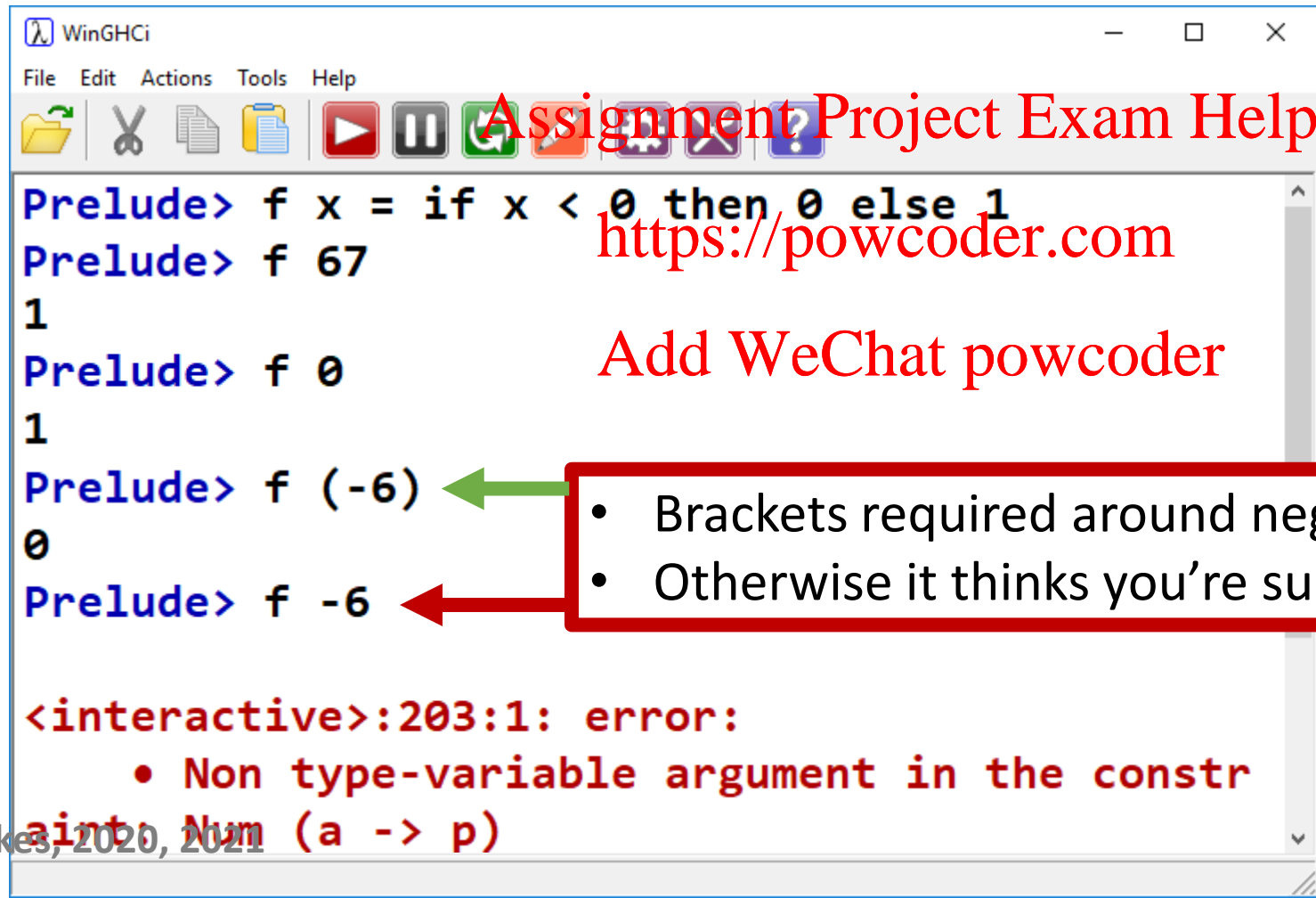
```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/powershell

PS C:\Users\aufke\Google Drive\Teaching\CCPS 506\Resources\Code\Haskell> ghci
GHCi, version 8.10.1: https://www.haskell.org/ghc/  :? for help
Prelude> :load Test.hs
[1 of 1] Compiling Test (Test.hs, interpreted)
Ok, one module loaded.
*Test> cube 5
125
*Test> square 10
100
*Test> :t square
square :: Num a => a -> a
*Test> |
```

Control Structures

if then else



The screenshot shows a WinGHCi window with the following content:

```
WinGHCi
File Edit Actions Tools Help
[Icons]
Prelude> f x = if x < 0 then 0 else 1
Prelude> f 67
1
Prelude> f 0
1
Prelude> f (-6)
0
Prelude> f -6
<interactive>:203:1: error:
  • Non type-variable argument in the constraint Num (a -> p)
```

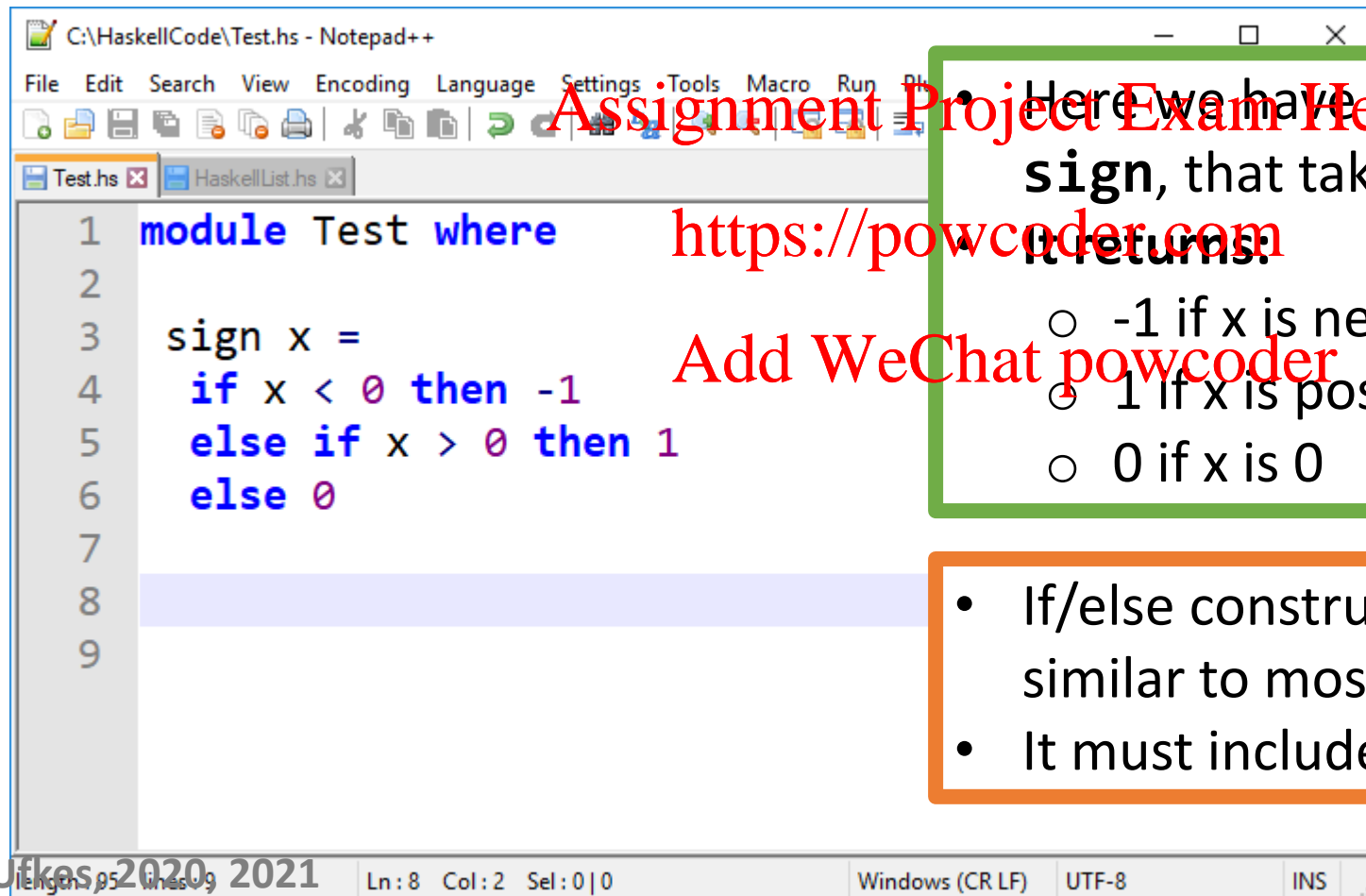
Overlaid on the screenshot is a red callout box with the following text:

- Brackets required around negative arguments
- Otherwise it thinks you're subtracting 6 from f

Red arrows point from the callout box to the arguments `(-6)` and `-6` in the code.

Control Structures

if then else if then else



The screenshot shows a Notepad++ window titled 'C:\HaskellCode\Test.hs - Notepad++'. The code is as follows:

```
1 module Test where
2
3 sign x =
4   if x < 0 then -1
5   else if x > 0 then 1
6   else 0
7
8
9
```

At the bottom of the window, the status bar shows 'Ln: 8 Col: 2 Sel: 0 | 0', 'Windows (CR LF)', 'UTF-8', and 'INS'.

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

- Here we have a function named **sign**, that takes one argument **x**
 - -1 if **x** is negative
 - 1 if **x** is positive
 - 0 if **x** is 0
- It returns:

- If/else construct in Haskell is similar to most other languages.
- It must include a **then** and an **else**

Control Structures

if then else if then else

```
1 module Test where
2
3 square x = x*x
4 cube x = x*x*x
5 sum x y = x + y
6
7 sign x =
8   if x < 0 then -1
9   else if x > 0 then 1
10  else 0
11
12
```

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

We can now format across multiple lines.
HOWEVER: Indentation matters in Haskell!

```
C:\HaskellCode\Test.hs - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?

Test.hs HaskellList.hs

1 module Test where
2
3 sign x =
4   if x < 0 then -1
5   else if x > 0 then 1
6   else 0
7
8
9

length: 92 lines: 9 Ln: 6 Col: 2 Sel: 0|0
```

```
WinGHCi
File Edit Actions Tools Help

Prelude> :load Test.hs
[1 of 1] Compiling Test
Failed, no modules loaded.
Prelude>
```

Assignment Project Exam Help

<https://powcoder.com>

parse error (possibly incorrect indentation or mismatched brackets)

Add WeChat powcoder

```
C:\HaskellCode\Test.hs - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
Test.hs HaskellList.hs

1 module Test where
2
3 sign x =
4   if x < 0 then -1
5   else if x > 0 then 1
6   else 0
7
8
9

length: 95 lines: 9 Ln: 4 Col: 3 Sel: 0|0
```

Assignment Project Exam Help

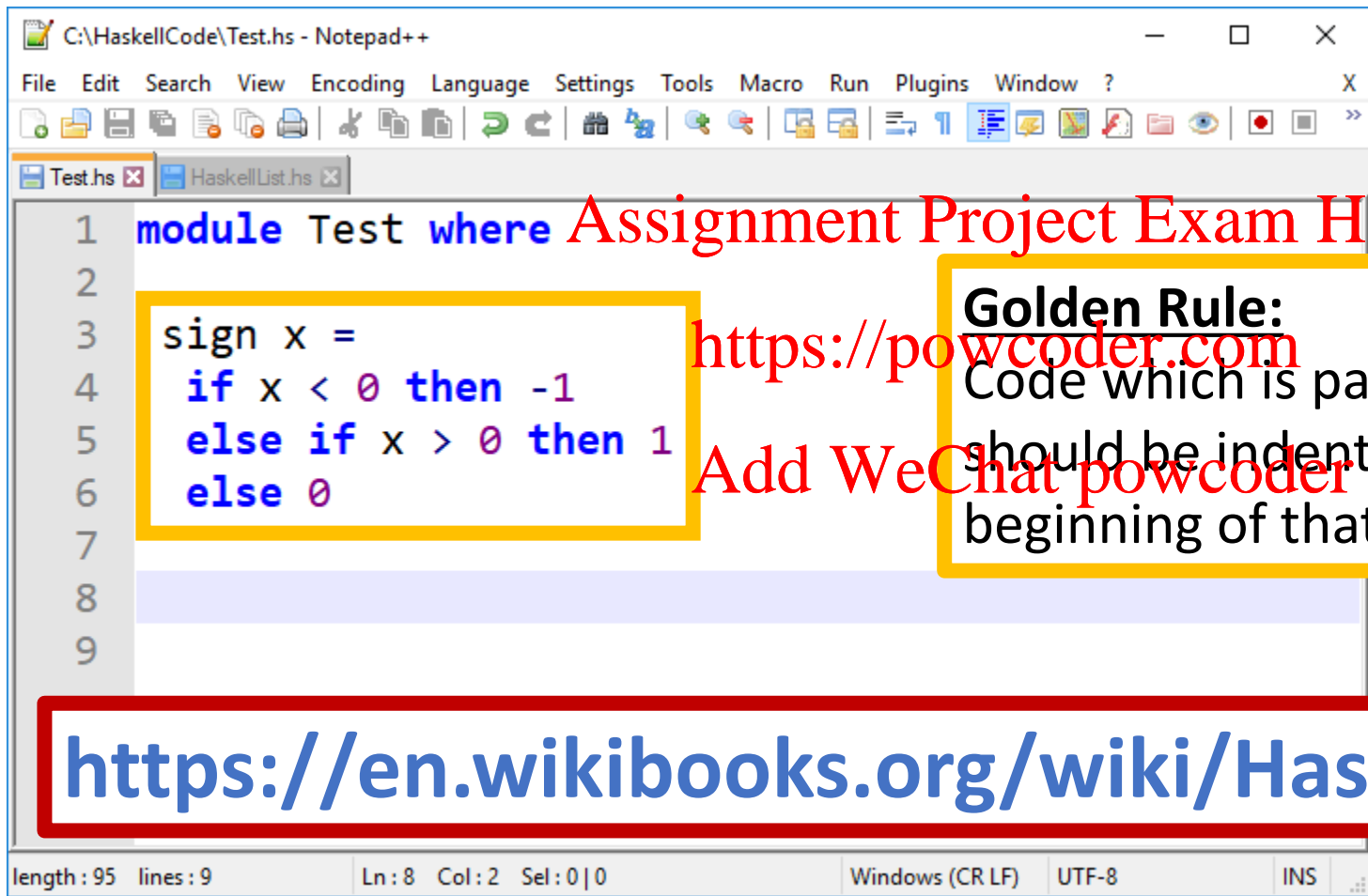
<https://powcoder.com>

Add WeChat powcoder

```
WinGHCi
File Edit Actions Tools Help
[1 of 1] Compiling Test (Test.hs,
interpreted )
Failed, no modules loaded.

Prelude> :reload
[1 of 1] Compiling Test (Test.hs,
interpreted )
Ok, one module loaded.
*Test>
```

Indenting in Haskell



Golden Rule:

Code which is part of some expression
should be indented further than the
beginning of that expression

<https://en.wikibooks.org/wiki/Haskell/Indentation>

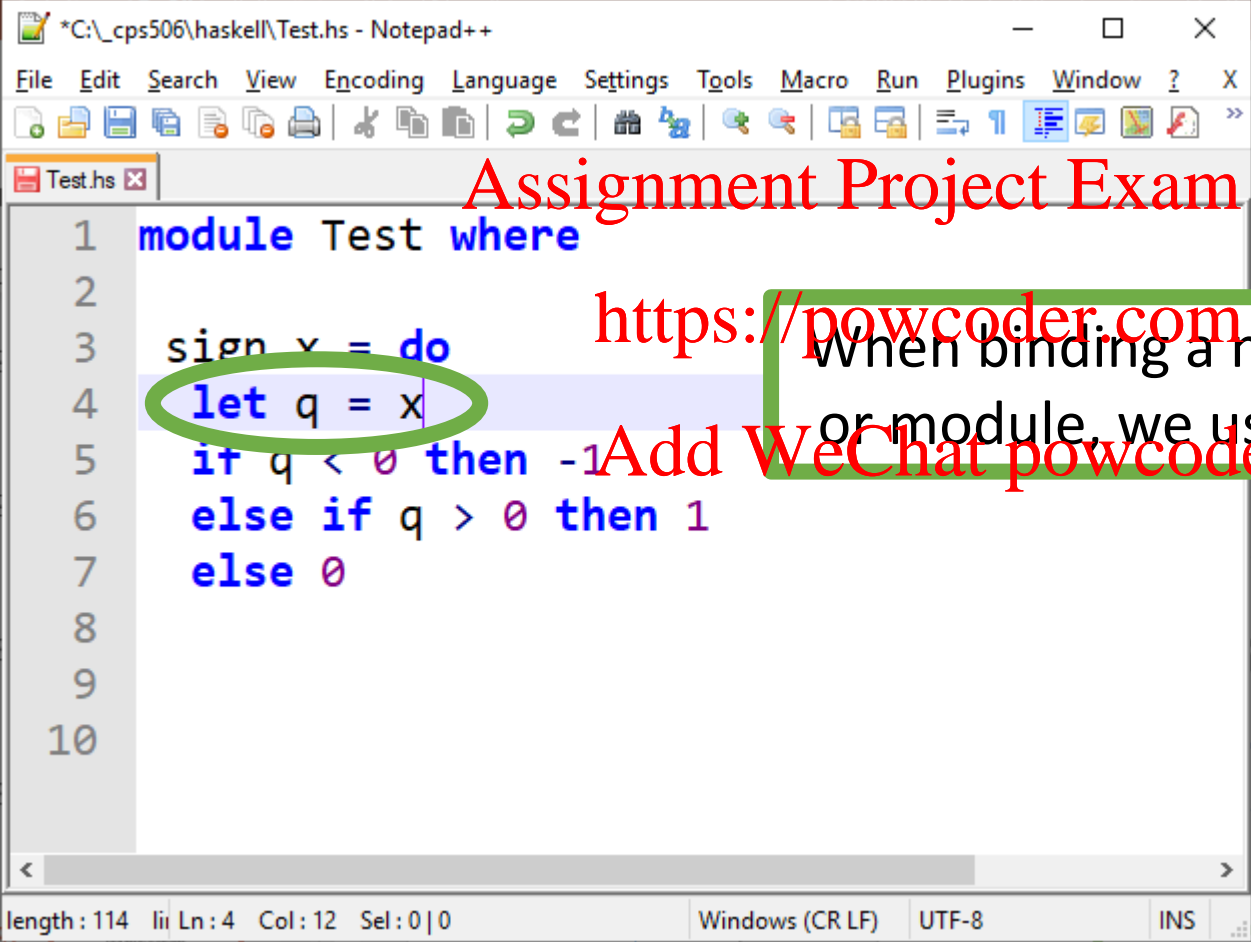
If all that weren't enough, Tabs don't work properly unless they're 8 spaces exactly.

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

Local Names in Functions



```
*C:\cps506\haskell\Test.hs - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ? X

Test.hs
1 module Test where
2
3 sign x = do
4   let q = x
5   if q < 0 then -1
6   else if q > 0 then 1
7   else 0
8
9
10

length: 114  Ln: 4  Col: 12  Sel: 0 | 0  Windows (CR LF)  UTF-8  INS
```

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

When binding a name inside a function or module, we use the “**let**” keyword

Multiple Expressions

```
*C:\_cps506\haskell\Test.hs - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ? X

Test.hs
1 module Test where
2
3 sign x = do
4   let q = x
5   if q < 0 then -1
6   else if q > 0 then 1
7   else 0
8
9
10
```

length: 114 lln: 4 Col: 12 Sel: 0 | 0 Windows (CR LF) UTF-8 INS

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

- We now have two expressions in this function!
 - `if/else` and `let`
- Must add the `do` keyword

```
*C:\cps506\haskell\Test.hs - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ? X

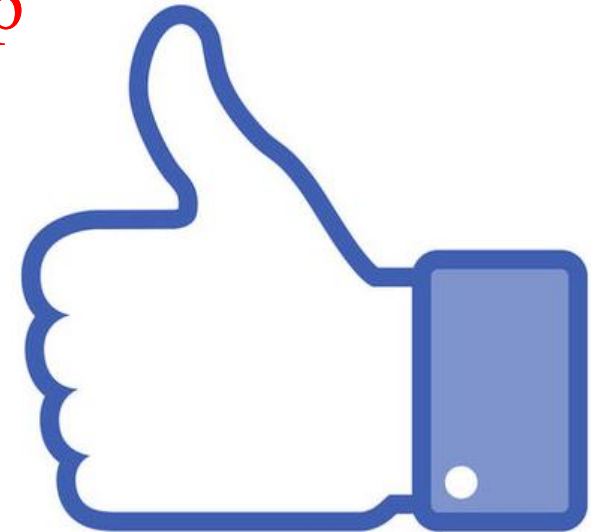
Test.hs x
1 module Test where
2
3 sign x = do
4 let q = x
5 if q < 0 then -1
6 else if q > 0 then 1
7 else 0
8
9
10
length: 114 |Ln: 4 Col: 12 Sel: 0|0 Window
```

```
WinGHCi
File Edit Actions Tools Help
Ok, one module loaded.
*Test> :reload
*Test> sign 0
0
*Test> sign 42
1
*Test> sign (-1)
-1
*Test>
```

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder



Case Statement

```
1 module Test where
2
3 isNum x =
4   case x of
5     0 -> 0
6     1 -> 1
7     2 -> 2
8     _ -> -1
9
10 sign x = do
11   let q = x
12   if x < 0 then -1
13   else if x > 0 then 1
```

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

When matching specific values, a case construct is easier to write. Just like Elixir, the `_` is wild. It catches everything else.

length: 184 lines: 17 Ln: 14 Col: 2 Sel: 0|0 Windows (CR LF) UTF-8 INS

Case Statement

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

```
C:\HaskellCode\Test.hs - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run
Test.hs HaskellList.hs
1 module Test where
2
3 isNum x =
4   case x of
5     0 -> 0
6     1 -> 1
7     2 -> 2
8     _ -> -1
9
10 sign x = do
11   let q = x
12   if x < 0 then -1
13   else if x > 0 then 1
length: 184 lines: 17 Ln: 14 Col: 2 Sel: 0|0
```

```
WinGHCi
File Edit Actions Tools Help
*Test> :reload
Ok, one module loaded.
*Test> isNum 0
0
*Test> isNum 1
1
*Test> isNum 2
2
*Test> isNum 999
-1
*Test>
```

Pattern Matching: Case

Assignment Project Exam Help
<https://powcoder.com>
Add WeChat powcoder

```
C:\HaskellCode\Test.hs - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins
Test.hs HaskellList.hs
1 module Test where
2
3 chkClr rgb =
4   case rgb of
5     (255, _, _) -> "RED"
6     (_, 255, _) -> "GREEN"
7     (_, _, 255) -> "BLUE"
8     (_, _, _)   -> "None"
9
10
11
12 nos x = x >= 0

length: 874 li Ln: 11 Col: 4 Sel: 0 | 0 Windows (CR LF) UTF-8 INS
```

```
WinGHCi
File Edit Actions Tools Help
*Test> chkClr (1, 2, 255)
"BLUE"
*Test> chkClr (155, 255, 55)
"GREEN"
*Test> chkClr (255, 255, 255)
"RED"
*Test> chkClr (55, 25, 2)
"None"
*Test>
```

Pattern Matching: Case

```
C:\HaskellCode\Test.hs - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plu
Test.hs HaskellList.hs
1 module Test where
2
3 chkClr rgb =
4   case rgb of
5     (255, _, _) -> "RED"
6     (_, 255, _) -> "GREEN"
7     (_, _, 255) -> "BLUE"
8     (255, 255, _) -> "YELLOW"
9     (255, _, 255) -> "MAGENTA"
10    (_, 255, 255) -> "CYAN"
11
12
length: 971 lin Ln: 15 Col: 4 Sel: 0|0 Windows (CR LF) UTF-
```

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

Will never match!

```
WinGHCi
File Edit Actions Tools Help
[1 of 1] Compiling Test
Test.hs interpreted )
Test.hs:8:4: warning: [-Woverlapping-p
patterns]
      Pattern match is redundant
      In a case alternative: (255, 255,
_) -> ...

8 | (255, 255, _) -> "YELLOW" |
  ^^^^^^^^^^^^^^^^^^^^^^^^^
```



```
C:\HaskellCode\Test.hs - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins
Test.hs HaskellList.hs
1 module Test where
2
3 chkClr rgb =
4   case rgb of
5     (255, 255, _) -> "YELLOW"
6     (255, _, 255) -> "MAGENTA"
7     (_, 255, 255) -> "CYAN"
8     (255, _, _) -> "RED"
9     (_, 255, _) -> "GREEN"
10    (_, _, 255) -> "BLUE"
11    (_, _, _) -> "None"
12
length: 971 lin Ln: 14 Col: 4 Sel: 0|0 Windows (CR LF) UTF-8
```

```
WinGHCi
File Edit Actions Tools Help
*Test> :reload
Ok, one module loaded.
*Test> chkClr (255, 5, 255)
"MAGENTA"
*Test> chkClr (255, 5, 55)
"RED"
*Test> chkClr (25, 255, 255)
"CYAN"
*Test> chkClr (25, 55, 5)
"None"
*Test> |
```

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder



Unlike Elixir...

```
C:\HaskellCode\Test.hs - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
Test.hs HaskellList.hs
1 module Test where
2
3 chkClr rgb =
4   case rgb of
5     (255, _, _) -> "RED"
6     (_, 255, _) -> "GREEN"
7     (_, _, 255) -> "BLUE"
8     x -> "None"
9
10
11
12
```

Try and be more general to catch anything that isn't a 3-tuple?

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

```
WinGHCi
File Edit Actions Tools Help
Test> chkClr (3, 3, 3)
"None"
Test> chkClr (3, 3)
Interactive>:406:8: error:
    • Couldn't match expected type
      '(Integer, Integer, Integer)'
      with actual type
      '(Integer, Integer)'
    • In the first argument of 'chkClr',
      namely '(3, 3)'
      In the expression: chkClr (3,
3)
```

Piecewise Functions

Just like Elixir's function signature pattern matching

```
C:\HaskellCode\Test.hs - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window
Test.hs HaskellList.hs
1 module Test where
2
3 fac 0 = 1
4 fac x = x*fac(x-1)
5
6 fib 0 = 0
7 fib 1 = 1
8 fib n = fib(n-1) + fib(n-2)
9
10
11 isNum x =
12 case x of
```

```
WinGHCi
File Edit Actions Tools Help
*Test> fac 3
6
*Test> fac 5
120
*Test> fib 5
5
*Test> fib 9
34
*Test>
```

Makes recursion very easy!

Piecewise Functions

```
C:\HaskellCode\Test.hs - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
Test.hs HaskellList.hs
1 module Test where
2
3 chkAxis (0, _) = (0, 1)
4 chkAxis (_, 0) = (1, 0)
5 chkAxis (a, b) = (a, b)
6
7
```

- Return unit vector if point lies on axis
- Return input point otherwise

length: 1,061 | Ln: 8 Col: 2 Sel: 0 | 0 Windows (CR LF) UTF-8 INS

```
WinGHCi
File Edit Actions Tools Help
*Test> chkAxis (1, 1)
(1,1)
*Test> chkAxis (8, 0)
(1,0)
*Test> chkAxis (42.3338, 0)
(1.0,0)
*Test> |
```

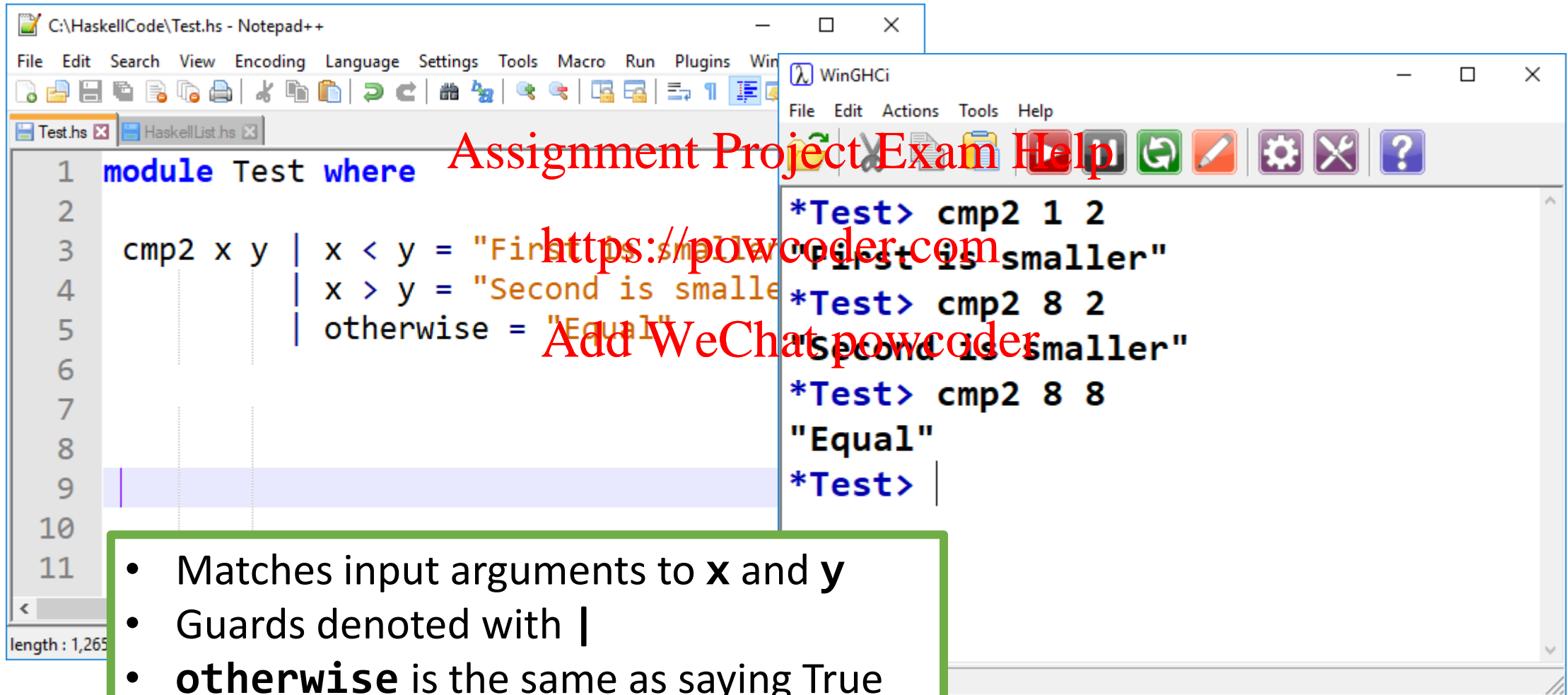
Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

Functions: Guards

Assignment Project Exam Help
<https://powcoder.com>
Add WeChat powcoder



The image shows two windows. The left window, titled 'C:\HaskellCode\Test.hs - Notepad++', displays Haskell code for a function `cmp2` with guards. The right window, titled 'WinGHCi', shows the interactive execution of the `cmp2` function with various arguments, demonstrating how the guards determine the output string.

```
1 module Test where
2
3 cmp2 x y | x < y = "First is smaller"
4          | x > y = "Second is smaller"
5          | otherwise = "Equal"
6
7
8
9
10
11
```

```
*Test> cmp2 1 2
"First is smaller"
*Test> cmp2 8 2
"Second is smaller"
*Test> cmp2 8 8
"Equal"
*Test>
```

- Matches input arguments to `x` and `y`
- Guards denoted with `|`
- **otherwise** is the same as saying True

Recursion

Assignment Project Exam Help
<https://powcoder.com>
Add WeChat powcoder

The image shows a side-by-side comparison of Haskell code and its execution. On the left, a Notepad++ window displays a Haskell module 'Test' with a custom recursive function 'llen'. On the right, a WinGHCi window shows the execution of the same module, demonstrating that both the custom 'llen' function and the built-in 'length' function return the same results for various inputs.

Classic list length finder

```
1 module Test where
2
3 llen [] = 0
4 llen x = 1 + llen(tail x)
```

Built-in length function

```
*Test> :reload
Ok, one module loaded.
*Test> 1 = [1, 3, 2, 7, 5]
*Test> length 1
5
*Test> llen 1
5
*Test> llen []
0
*Test> llen [1]
1
*Test>
```

Our own user function

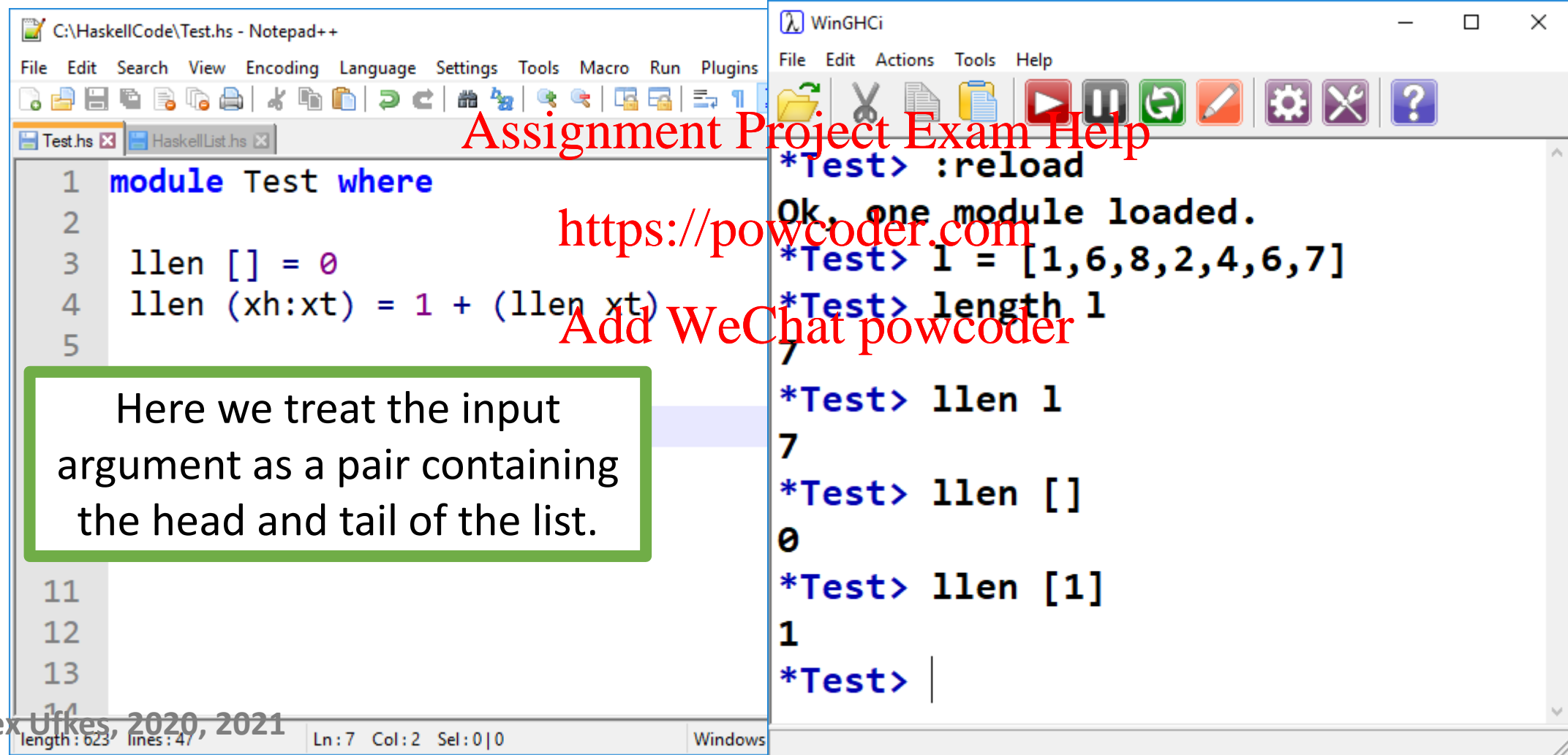
Tail Recursion?

Less important in Haskell

- In Haskell, function call model is different
- Function calls don't necessarily create a new stack frame
- In practice, tail recursion not a big deal.

Recursion: cons

Assignment Project Exam Help
<https://powcoder.com>
Add WeChat powcoder



The image shows two windows from a Haskell development environment. The left window, titled 'C:\HaskellCode\Test.hs - Notepad++', contains the following Haskell code:

```
1 module Test where
2
3 llen [] = 0
4 llen (xh:xt) = 1 + (llen xt)
5
11
12
13
14
```

A green box highlights the recursive definition of `llen`, with the text: "Here we treat the input argument as a pair containing the head and tail of the list." The status bar at the bottom of this window shows "length: 623 lines: 47".

The right window, titled 'WinGHCi', shows the GHCi prompt with the following interactions:

```
*Test> :reload
Ok, one module loaded.
*Test> l = [1,6,8,2,4,6,7]
*Test> length l
7
*Test> llen l
7
*Test> llen []
0
*Test> llen [1]
1
*Test> |
```

© Alex Ufkes, 2020, 2021

Recursion: filter

```
*C:\HaskellCode\Test.hs - Notepad++
File Edit Search View Encoding Language Settings To
Test.hs HaskellList.hs
1 module Test where
2
3 pos x = x >= 0
4
5 filt p [] = []
6 filt p (xh:xt) =
7   if p xh then xh : filt p xt
8   else filt p xt
9
10
11
12
13
14
```

- Returns true if $x \geq 0$
- False otherwise

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

- First argument is a Boolean function
- Second input is a list
- Base case is if the list is empty
- Otherwise, we call the function p with the head of the list.
- If true, append it to the running list
- If false, do not append
- In both cases, make the recursive call with the tail.

Recursion: filter

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

Test pos function

```
*C:\HaskellCode\Test.hs - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window
Test.hs HaskellList.hs
1 module Test where
2
3 pos x = x >= 0
4
5 filt p [] = []
6 filt p (xh:xt) =
7   if p xh then xh : filt p xt
8   else filt p xt
9
10
11
12
13
14
```

```
WinGHCI
File Edit Actions Tools Help
*Test> pos 4
True
*Test> pos (-5)
False
*Test> l = [-1, 2, -3, 4, -5, 6]
*Test> filt pos l
[2,4,6]
*Test> filt pos [-1]
[]
*Test> filt pos []
[]
*Test>
```

Return Multiple *Things*?

Lists/tuples to the rescue!

```
1 module Test where
2
3 roots a b c =
4   ( (-b + sqrt(b*b - 4*a*c))/(2*a),
5     (-b - sqrt(b*b - 4*a*c))/(2*a) )
6
7 fac 0 = 1
8 fac x = x*fac(x-1)
9
10 fib 0 = 0
11 fib 1 = 1
12 fib n = fib(n-1) + fib(n-2)
```

length: 270 Lines: 20 Ln: 7 Col: 11 Sel: 0|0 Windows (CR LF) UTF-8 INS

let/in Structure

```
1 module Test where
2
3 roots a b c =
4   let disc = sqrt(b*b - 4*a*c)
5   in  ((-b + disc)/(2*a),
6       (-b - disc)/(2*a))
7
8 fac 0 = 1
9 fac x = x*fac(x-1)
10
11 fib 0 = 0
12 fib 1 = 1
```

length: 387 lines: 29 Ln: 10 Col: 2 Sel: 0|0 Windows (CR LF) UTF-8 INS

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

- This is one expression (**let/in**)
- We don't need to add **do** keyword

```
C:\HaskellCode\Test.hs - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?

Test.hs HaskellList.hs
1 module Test where
2
3 roots a b c =
4   let disc = sqrt(b*b - 4*a*c)
5   in  ((-b + disc)/(2*a),
6       (-b - disc)/(2*a))
7
8 fac 0 = 1
9 fac x = x*fac(x-1)
10
11 fib 0 = 0
12 fib 1 = 1

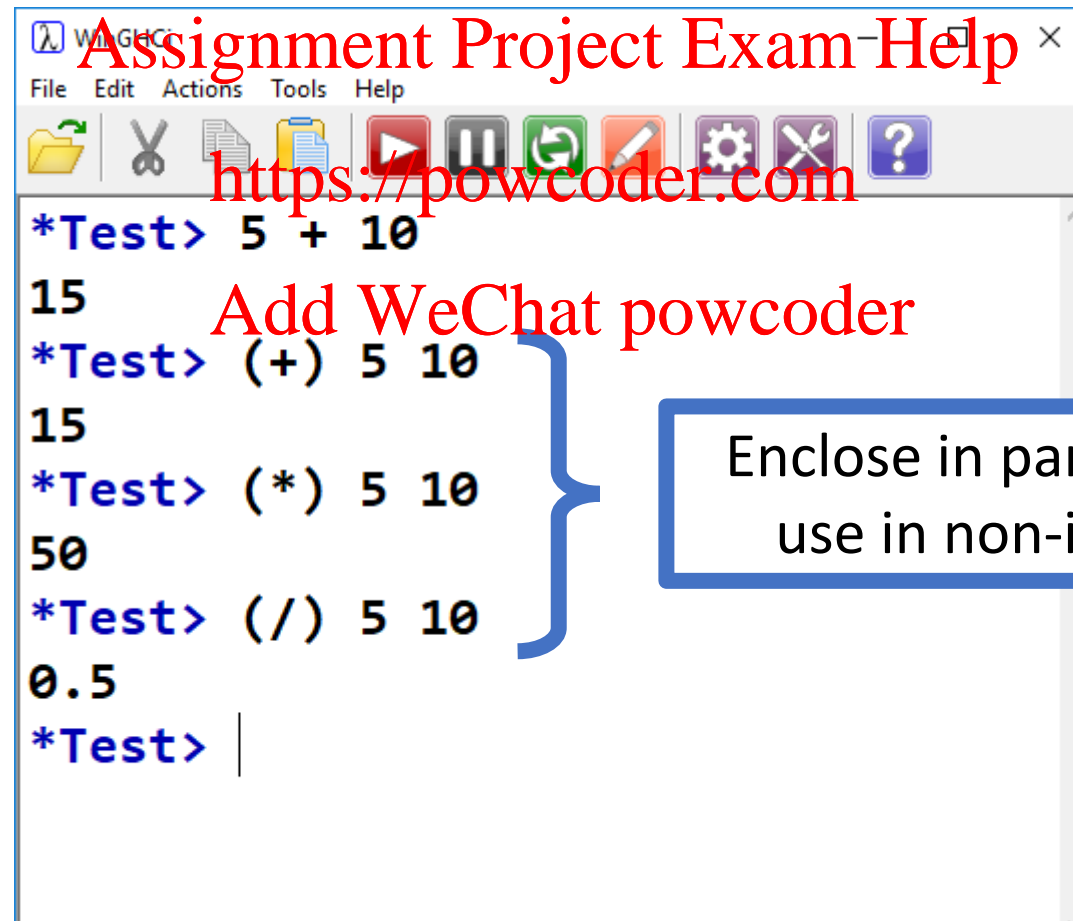
length: 387 lines: 29 Ln: 10 Col: 2 Sel: 0|0 Window
```

```
WinGHCi
File Edit Actions Tools Help

*Test> roots 1 2 (-6)
(1.6457513110645907,-3.6457513110645907)
*Test> roots 1 (-2) 4
(NaN,NaN)
*Test> roots (-1) 2 (-4)
(NaN,NaN)
*Test> roots (-1) 2 4
(-1.2360679774997898,3.2360679774997898)
*Test> |
```

Infix Functions

Use symbolic operators as functions:



The screenshot shows a terminal window with a menu bar (File, Edit, Actions, Tools, Help) and a toolbar with icons for file operations and execution. The terminal displays the following commands and outputs:

```
*Test> 5 + 10
15
*Test> (+) 5 10
15
*Test> (*) 5 10
50
*Test> (/) 5 10
0.5
*Test> |
```

Red text overlays the image: "Assignment Project Exam Help" and "https://powcoder.com" are at the top, and "Add WeChat powcoder" is in the middle. A blue bracket groups the last three commands, pointing to a callout box.

Enclose in parentheses to use in non-infix mode

Function Composition

Assignment Project Exam Help
<https://powcoder.com>
Add WeChat powcoder

```
WinGHCi
File Edit Actions Tools Help
[Icons]
*Test> fac(fib(4))
6
*Test> fac(fib(5))
120
*Test> fac(fib(6))
40320
*Test> |
```

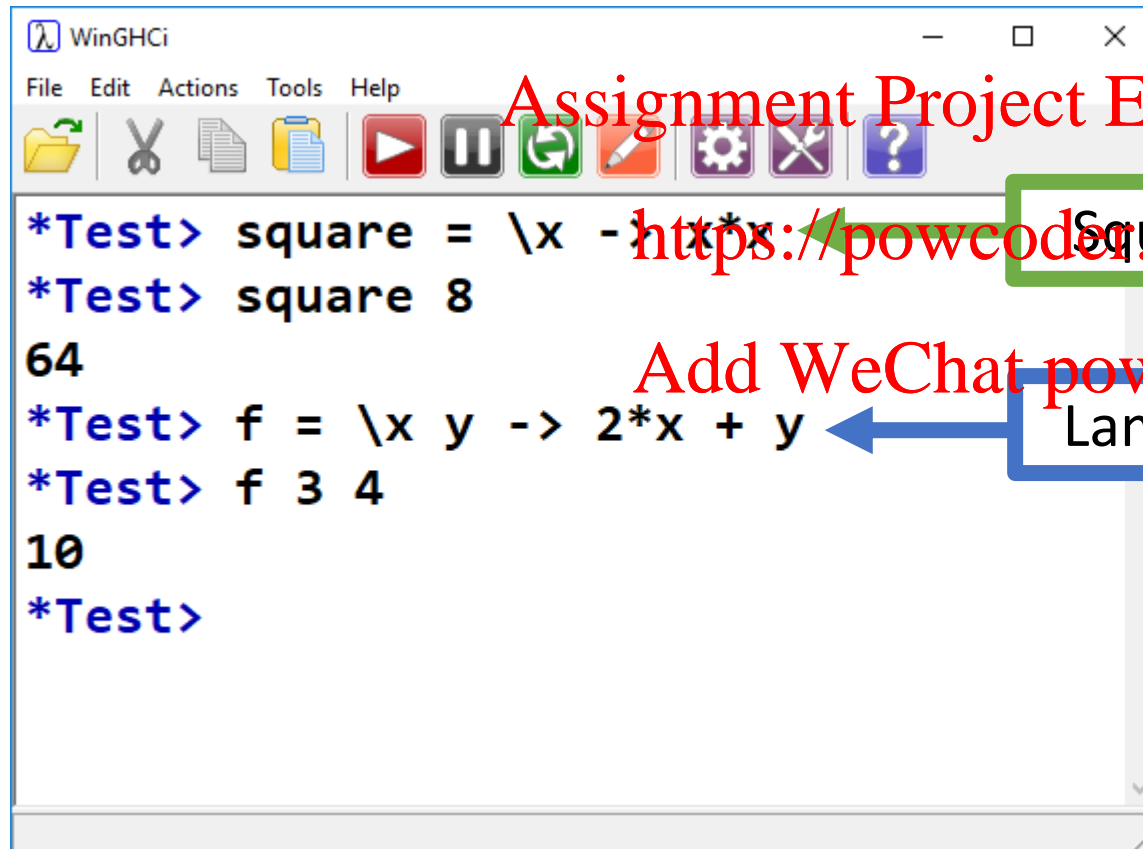
Can be written as:

```
WinGHCi
File Edit Actions Tools Help
[Icons]
*Test> (fac.fib) 4
6
*Test> (fac.fib) 5
120
*Test> (fac.fib) 6
40320
*Test> |
```

In math, $f \circ g$ means “*f* following *g*”. Same thing in Haskell.

Lambda Functions

Like anonymous functions in Elixir:



```
WinGHCi
File Edit Actions Tools Help
*Test> square = \x -> x*x
*Test> square 8
64
*Test> f = \x y -> 2*x + y
*Test> f 3 4
10
*Test>
```

Assignment Project Exam Help

<https://powcoder.com>

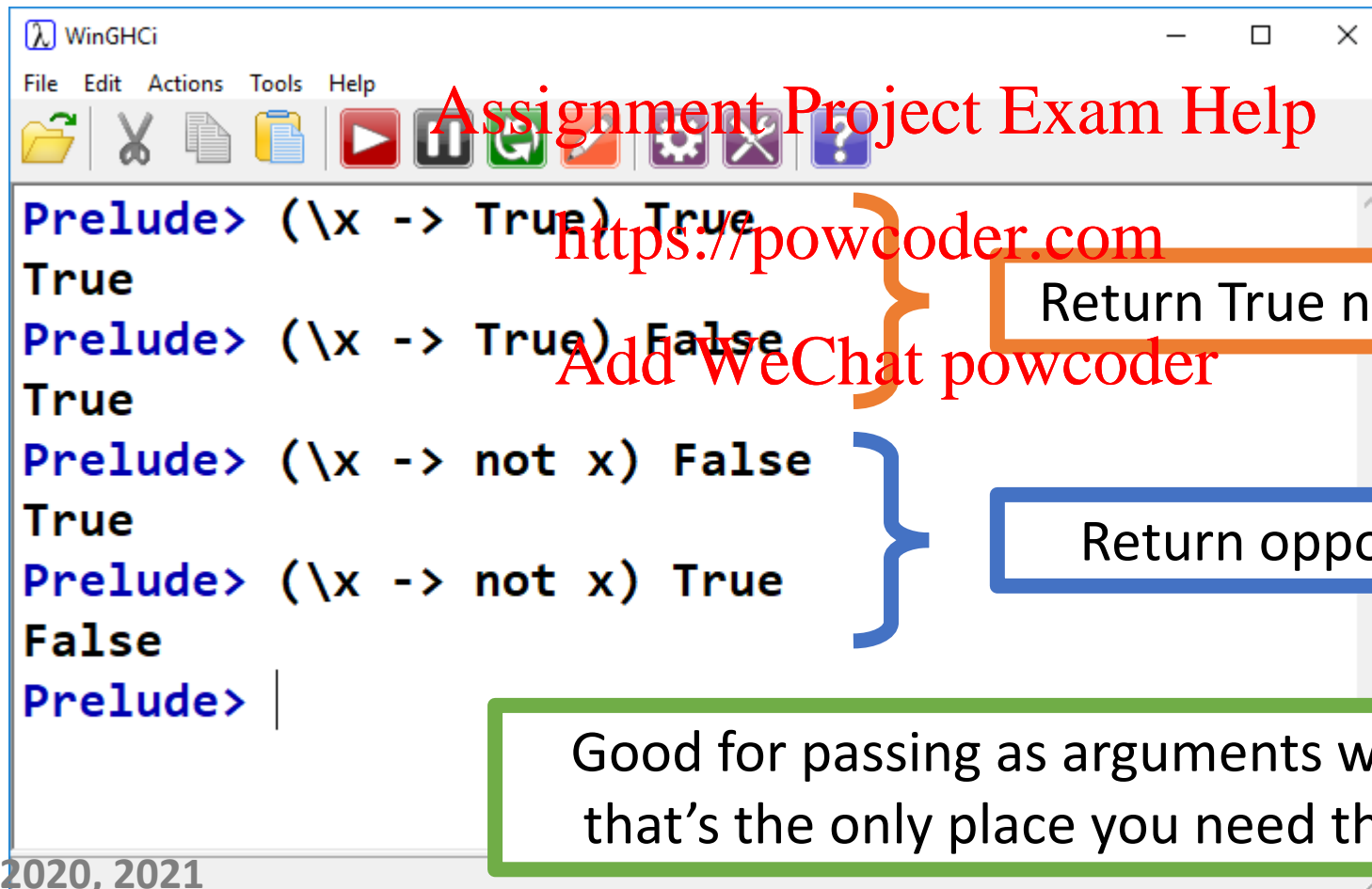
Add WeChat powcoder

Square as Lambda function

Lambda function with two args

Lambda Functions

They don't need names!



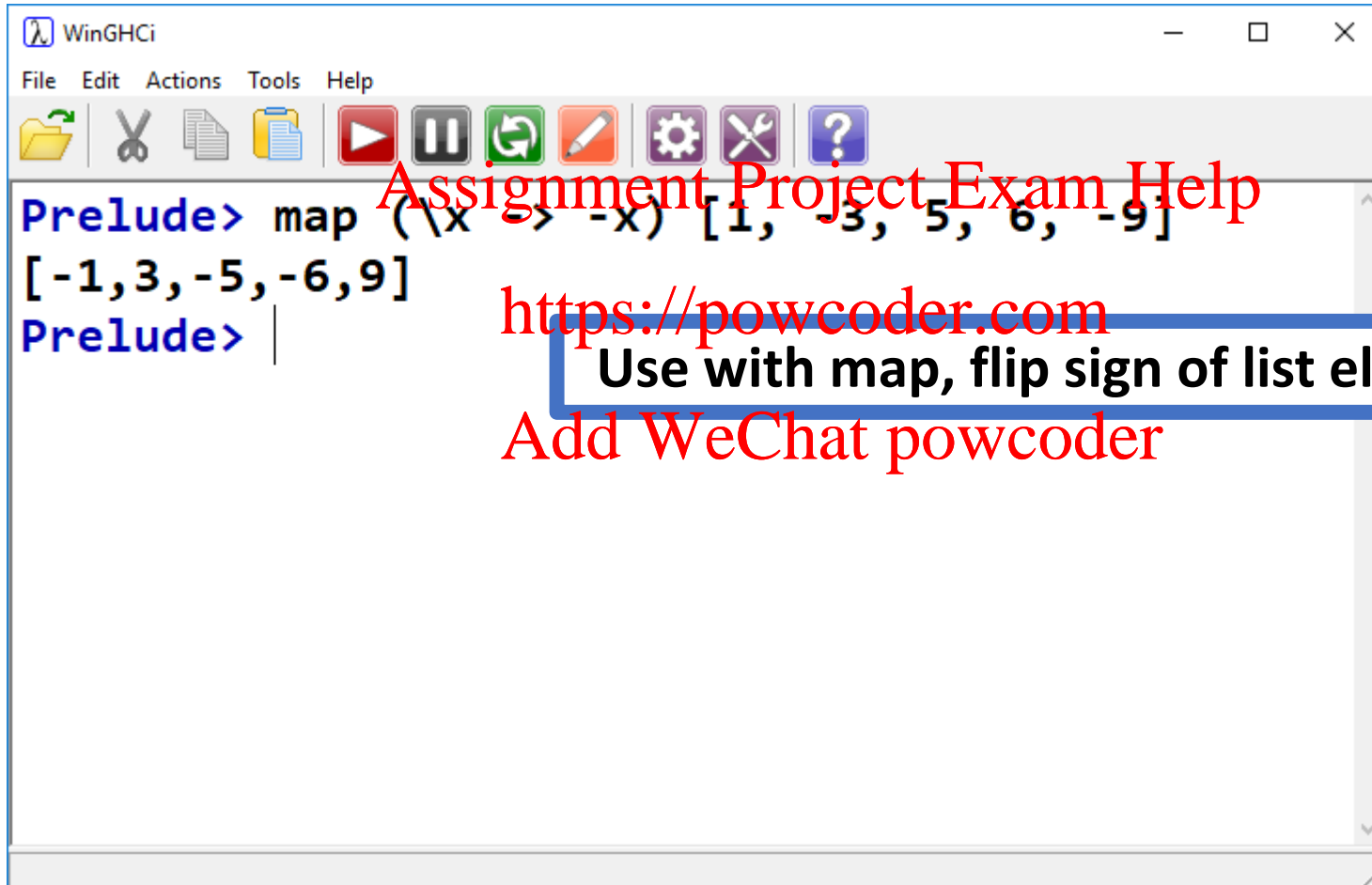
The screenshot shows the WinGHCi window with the following content:

```
WinGHCi
File Edit Actions Tools Help
[Icons]
Prelude> (\x -> True) True
True
Prelude> (\x -> True) False
True
Prelude> (\x -> not x) False
True
Prelude> (\x -> not x) True
False
Prelude> |
```

Annotations on the screenshot:

- A red watermark "Assignment Project Exam Help" is at the top.
- A red watermark "https://powcoder.com" is in the middle.
- A red watermark "Add WeChat powcoder" is in the middle.
- An orange box on the right contains the text "Return True no matter what", with a bracket pointing to the first two lines of code.
- A blue box on the right contains the text "Return opposite of input", with a bracket pointing to the last two lines of code.
- A green box at the bottom contains the text "Good for passing as arguments when that's the only place you need them".

Good for passing as arguments when that's the only place you need them

A screenshot of the WinGHCi window. The title bar says 'WinGHCi'. The menu bar includes 'File', 'Edit', 'Actions', 'Tools', and 'Help'. The toolbar contains icons for file operations (folder, scissors, document), execution (play, pause, refresh), editing (pencil), settings (gear), and help (question mark). The main text area shows the following interaction:

```
Prelude> map (\x -> -x) [1, -3, 5, 6, -9]  
[-1,3,-5,-6,9]  
Prelude> |
```

Assignment Project Exam Help

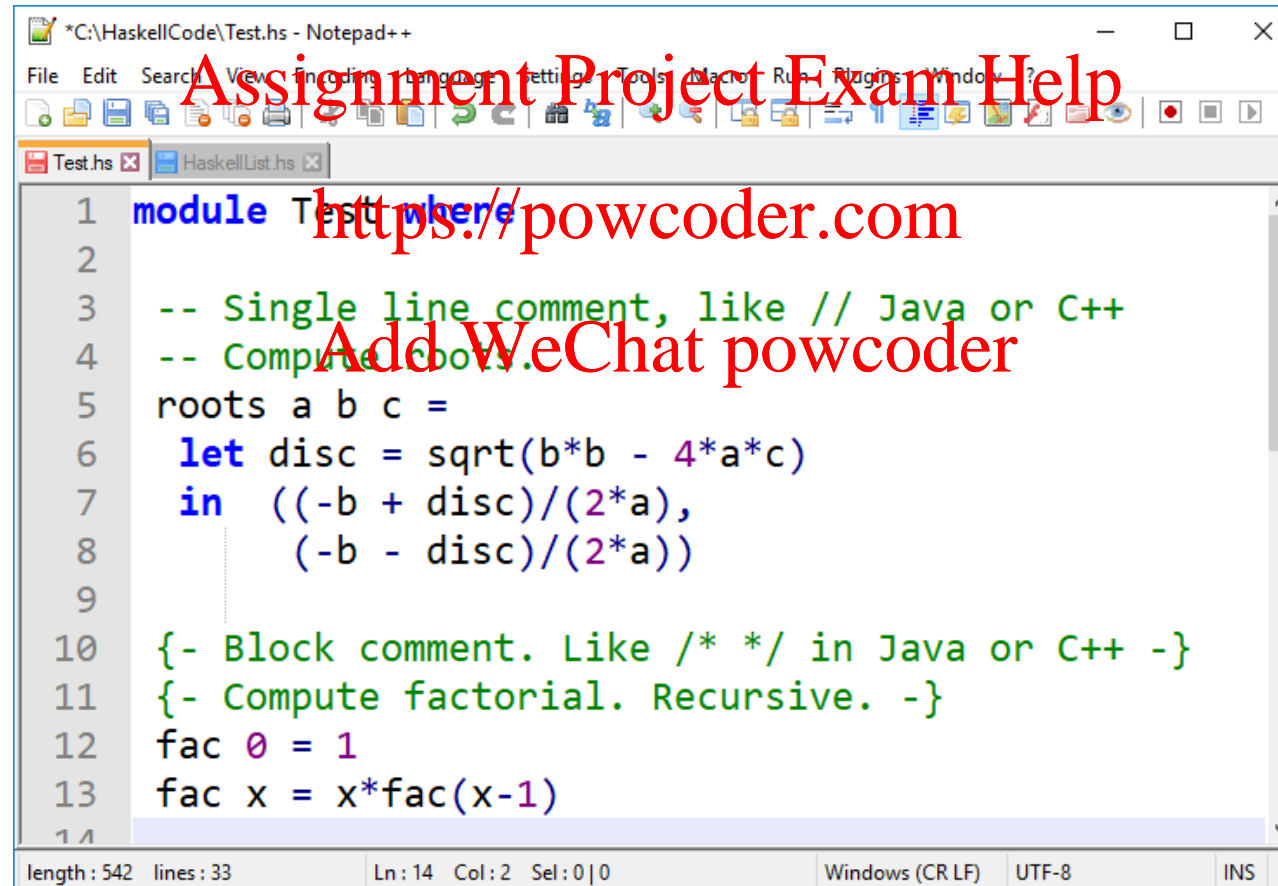
<https://powcoder.com>

Use with map, flip sign of list elements

Add WeChat powcoder

Comments

About time we mentioned them...



```
1 module Test where
2
3 -- Single line comment, like // Java or C++
4 -- Compute roots.
5 roots a b c =
6   let disc = sqrt(b*b - 4*a*c)
7   in  ((-b + disc)/(2*a),
8       (-b - disc)/(2*a))
9
10 {- Block comment. Like /* */ in Java or C++ -}
11 {- Compute factorial. Recursive. -}
12 fac 0 = 1
13 fac x = x*fac(x-1)
```

Type Inference

```
C:\HaskellCode\Test.hs - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
Test.hs HaskellList.hs
1 module Test where
2
3 chkClr rgb =
4   case rgb of
5     (255, _, _) -> "RED"
6     (_, 255, _) -> "GREEN"
7     (_, _, 255) -> "BLUE"
8     x -> "None"
9
10
11
12
length: 864 lin Ln: 11 Col: 4 Sel: 0|0 Windows (CR LF) UTF-8 INS
```

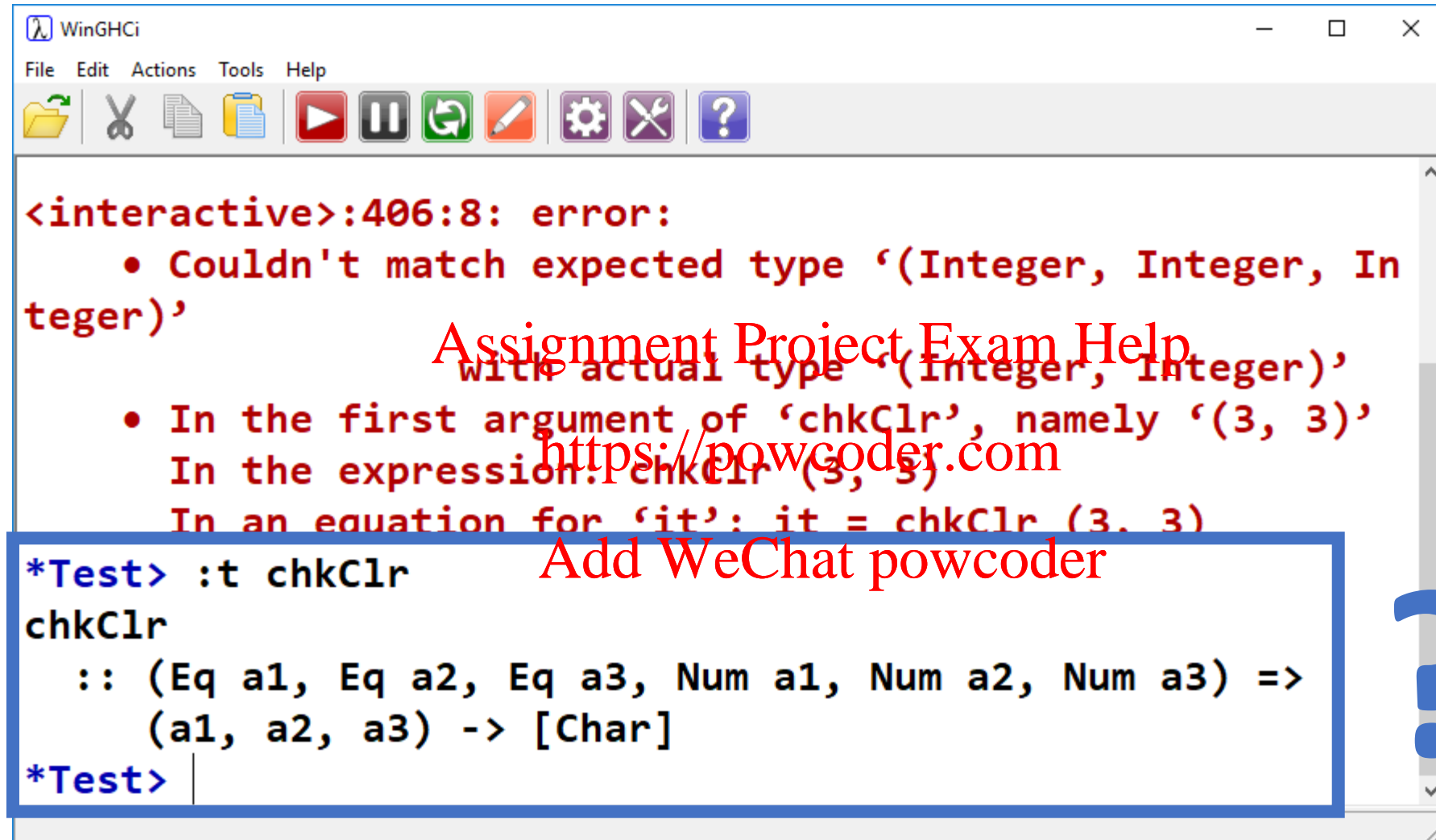
Remember this error?

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

```
WinGHCi
File Edit Actions Tools Help
Test> chkClr (3, 3, 3)
"None"
Test> chkClr (3, 3)
Interactive>:406:8: error:
    • Couldn't match expected type
      '(Integer, Integer, Integer)'
      with actual type
      '(Integer, Integer)'
    • In the first argument of 'chkClr',
      namely '(3, 3)'
      In the expression: chkClr (3,
3)
```



The image shows a screenshot of the WinGHCi window. The title bar says 'WinGHCi'. The menu bar includes 'File', 'Edit', 'Actions', 'Tools', and 'Help'. The toolbar contains icons for file operations (folder, scissors, document), execution (play, pause, refresh), editing (pencil), settings (gear), and help (question mark). The main text area displays a red error message from the Haskell compiler. The error message states that a function couldn't match the expected type '(Integer, Integer, Integer)' with the actual type '(Integer, Integer)'. It points to the first argument of 'chkClr', '(3, 3)', and mentions an equation for 'it': 'it = chkClr (3, 3)'. Below the error, a blue box highlights the function definition for 'chkClr'. The definition shows the type signature ':t chkClr' and the function body 'chkClr :: (Eq a1, Eq a2, Eq a3, Num a1, Num a2, Num a3) => (a1, a2, a3) -> [Char]'. The prompt '*Test>' is visible at the bottom left of the blue box. A large blue question mark is positioned to the right of the blue box.

```
<interactive>:406:8: error:
  • Couldn't match expected type '(Integer, Integer, Integer)'
    with actual type '(Integer, Integer)'
  • In the first argument of 'chkClr', namely '(3, 3)'
    In the expression: chkClr (3, 3)
    In an equation for 'it': it = chkClr (3, 3)

*Test> :t chkClr
chkClr
  :: (Eq a1, Eq a2, Eq a3, Num a1, Num a2, Num a3) =>
     (a1, a2, a3) -> [Char]
*Test> |
```

C:\HaskellCode\Test.hs - Notepad++

File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ? X

Test.hs x Has

WinGHCi

File Edit Actions Tools Help

1 modu
2
3 chk
4 ca
5 (• Couldn't match exp
6 (teger)'
7
8 x
9
10
11
12 pos

<interactive>:406:8: error
• Couldn't match expected type 'Integer' with actual type 'Integer'
• In the first argument of 'chkClr', namely 'a1'.
In the expression: ...
In an equation for 'chkClr':

***Test> :t chkClr**
chkClr
:: (Eq a1, Eq a2, Eq a3, Num a1, Num a2, Num a3) =>
(a1, a2, a3) -> [Char]
***Test> |**

Type of chkClr is inferred:

- Takes as input a tuple with three values
 - (a1, a2, a3)
- The type of each value in the tuple must be instances of type class **Eq** and **Num**.
- The output of chkClr is a character list.

Type Inference

```
C:\HaskellCode\Test.hs - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ? X
Test.hs HaskellList.hs
1 module Test where
2
3 chkClr rgb =
4   case rgb of
5     (255, _, _) -> "RED"
6     (_, 255, _) -> "GREEN"
7     (_, _, 255) -> "BLUE"
8     x -> "None"
9
10
11
12 pos x = x
```

length: 864 lin Ln: 11 Col: 4 Sel: 0|0 Windows (CR LF) UTF-8 INS

Based on the contents of `chkClr`:

- Haskell determined that the output is `[Char]`
- The input is a 3-tuple whose elements must be instances of `Num` and `Eq`.

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

```
*Test> :t chkClr
chkClr
  :: (Eq a1, Eq a2, Eq a3, Num a1, Num a2, Num a3) =>
     (a1, a2, a3) -> [Char]
*Test> |
```

Specify Function Type

C:\HaskellCode\Test.hs - Notepad++

File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?

Test.hs HaskellList.hs

```
1 module Test where
2
3 chkAxis :: (Float, Float) -> (Float, Float)
4 chkAxis (0, _) = (0, 1)
5 chkAxis (_, 0) = (1, 0)
6 chkAxis (a, b) = (a, b)
7
```

- **chkAxis** takes a pair-tuple of Floats as input, and returns the same as output.
- Instead of constants being of type Num or Fractional, they are treated as Floats

WinGHCi

File Edit Actions Tools Help

```
*Test> chkAxis (1, 0)
(1.0,0.0)
*Test> chkAxis (0, 4.5)
(0.0,1.0)
*Test> chkAxis (3, 4)
(3.0,4.0)
*Test> chkAxis (4.333, 0)
(1.0,0.0)
*Test> :t chkAxis
chkAxis :: (Float, Float) ->
(Float, Float)
*Test> |
```


Specify Function Type

The image shows two windows from a Haskell development environment. The left window, titled 'C:\HaskellCode\Test.hs - Notepad++', contains the following Haskell code:

```
1 module Test where
2
3 cmp2 :: Int -> Int -> [Char]
4 cmp2 x y | x < y = "First is smaller"
5          | x > y = "Second is smaller"
6          | otherwise = "Equal"
7
8
9
10
11
```

The type signature `cmp2 :: Int -> Int -> [Char]` on line 3 is highlighted with a green box. The right window, titled 'WinGHCi', shows the execution of the `cmp2` function:

```
*Test> cmp2 1 2
"First is smaller"
*Test> cmp2 8 2
"Second is smaller"
*Test> cmp2 8 8
"Equal"
*Test> |
```

Overlaid on the image in red text are the phrases 'Assignment Project Exam Help', 'https://powcoder.com', and 'Add WeChat powcoder'.

Thoughts?

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

*Test> cmp2 1 2
"First is smaller"
*Test> cmp2 8 2
"Second is smaller"
*Test> cmp2 8 8
"Equal"
*Test> cmp2 1.1 1.2
```

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

*Test> cmp2 8 8
"Equal"
*Test> cmp2 1.1 1.2
<Interactive: 162.0.0.0>
Add WeChat powcoder:
• No instance for (Fractional Int) arising from
the literal '1.1'
• In the first argument of 'cmp2', namely '1.1'
In the expression: cmp2 1.1 1.2
In an equation for 'it': it = cmp2 1.1 1.2
*Test>
```

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder:

- No instance for (Fractional Int) arising from the literal '1.1'
- In the first argument of 'cmp2', namely '1.1'
In the expression: cmp2 1.1 1.2
In an equation for 'it': it = cmp2 1.1 1.2

```
C:\HaskellCode\Test.hs - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
Test.hs HaskellList.hs
1 module Test where
2
3 --cmp2 :: Int -> Int -> [Char]
4 cmp2 x y | x < y = "First is smaller"
5          | x > y = "Second is smaller"
6          | otherwise = "Equal"
7
8
9
10
11
length: 1,267 lines Ln: 9 Col: 2 Sel: 0|0
```

```
WinGHCi
File Edit Actions Tools Help
*Test> cmp2 1.1 1.2
"First is smaller"
*Test> :t cmp2
cmp2 :: Ord a => a -> a -> [Char]
*Test>
```

Ord is a type class:

- When we didn't explicitly define our types, Haskell inferred the type for us.
- Ord is a type class under which the operations used on our inputs are defined.
- I.e., comparison operators.

Type VS Type Class

```
C:\HaskellCode\Test.hs - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ? X
Test.hs HaskellList.hs
1 module Test where
2
3 cmp2 :: Int -> Int -> [Char]
4 cmp2 x y | x < y = "First is smaller"
5          | x > y = "Second is smaller"
6          | otherwise = "Equal"
7
8
9
10
```

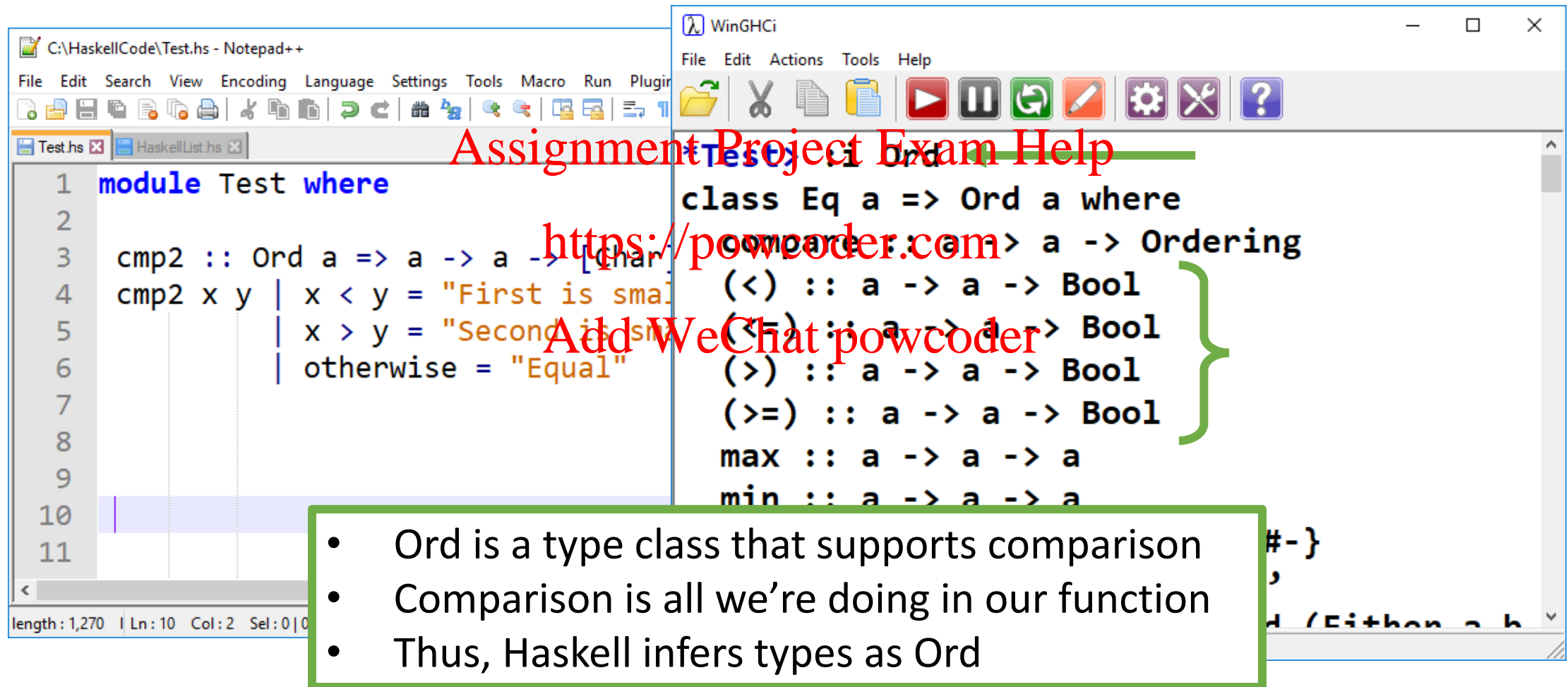
- Int & Char are types, **not** type classes
- We can use the above notation

```
C:\HaskellCode\Test.hs - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ? X
Test.hs HaskellList.hs
1 module Test where
2
3 cmp2 :: Ord a => a -> a -> [Char]
4 cmp2 x y | x < y = "First is smaller"
5          | x > y = "Second is smaller"
6          | otherwise = "Equal"
7
8
9
10
```

- Ord is a type class, thus we specify that **a** is an instance of Ord
- **cmp2** accepts two instances of Ord as arguments.
- Ord contains many different types, **a** can be any of them

Ord Type Class

Assignment Project Exam Help
<https://powcoder.com>
Add WeChat powcoder



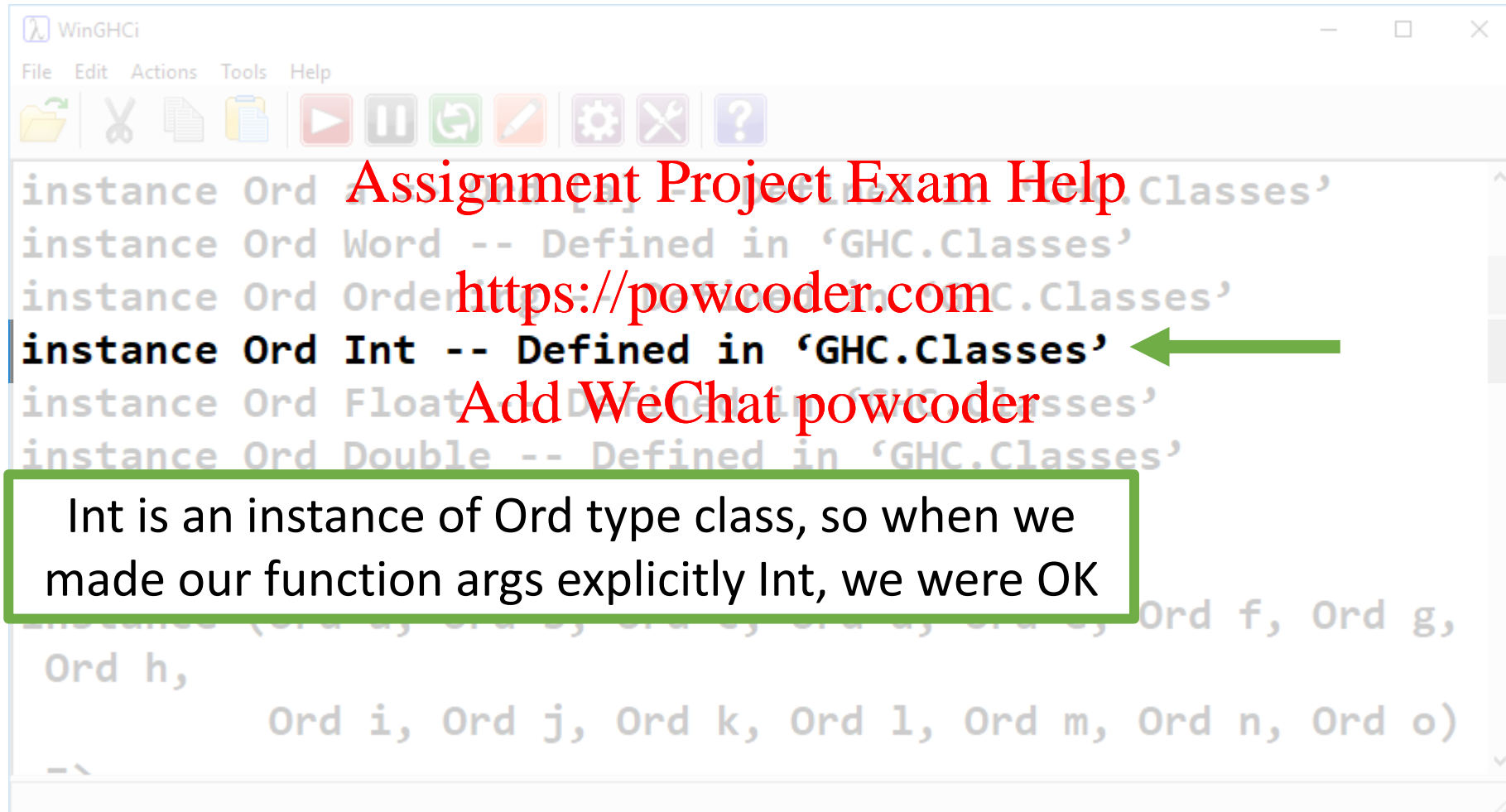
The image shows two windows. The left window, Notepad++, displays a Haskell module 'Test' with a function 'cmp2' that takes two 'Ord' values and returns a string indicating their relative order. The right window, WinGHCi, shows the definition of the 'Ord' type class, which includes methods for comparison ('lt', 'gt', 'le', 'ge') and finding maximum/minimum values ('max', 'min'). A green bracket groups the comparison methods.

```
1 module Test where
2
3 cmp2 :: Ord a => a -> a -> [Char]
4 cmp2 x y | x < y = "First is small"
5          | x > y = "Second is small"
6          | otherwise = "Equal"
7
8
9
10
11
```

```
class Eq a => Ord a where
  compare :: a -> a -> Ordering
  (<) :: a -> a -> Bool
  (<=) :: a -> a -> Bool
  (>) :: a -> a -> Bool
  (>=) :: a -> a -> Bool
  max :: a -> a -> a
  min :: a -> a -> a
```

- Ord is a type class that supports comparison
- Comparison is all we're doing in our function
- Thus, Haskell infers types as Ord

Ord Type Class



The image shows a screenshot of the WinGHCi window. The menu bar includes 'File', 'Edit', 'Actions', 'Tools', and 'Help'. The toolbar contains icons for file operations and execution. The code in the editor is as follows:

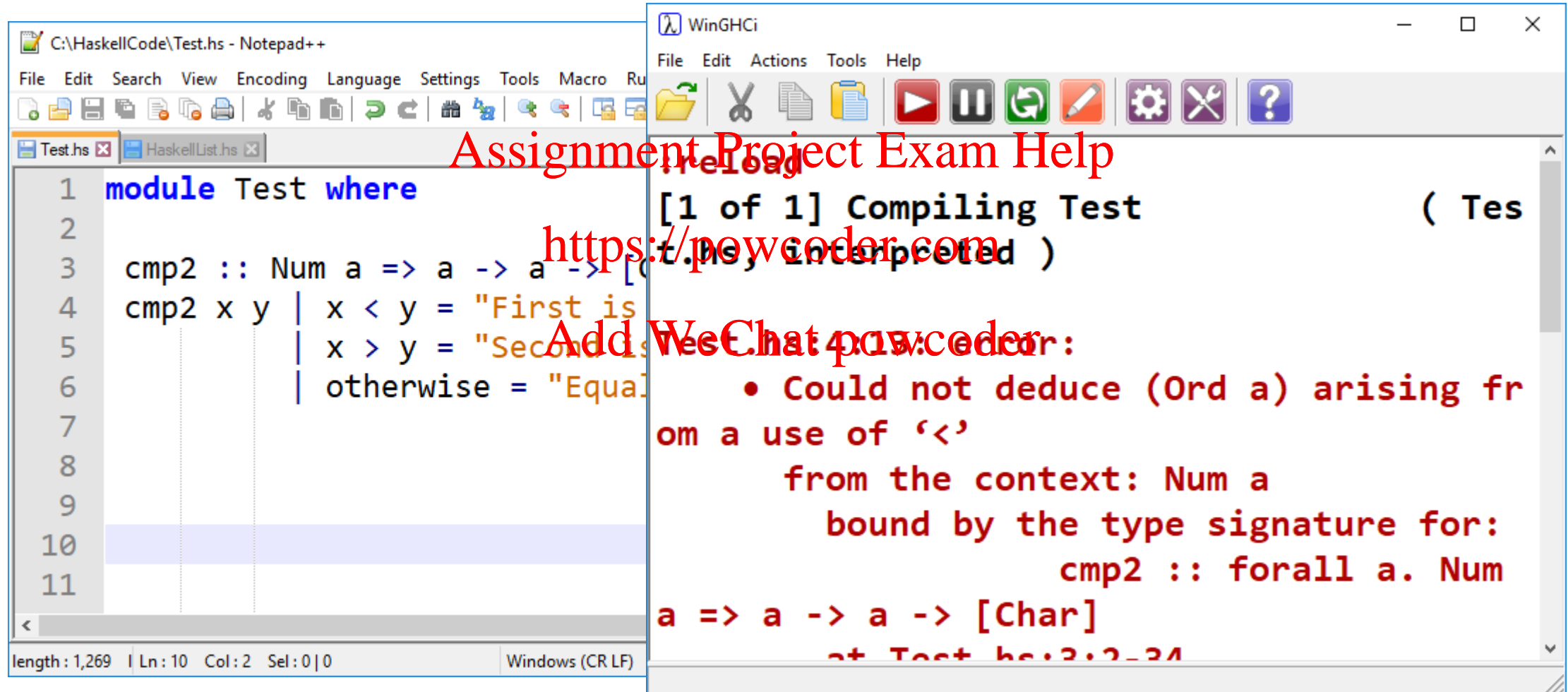
```
instance Ord Char -- Defined in 'GHC.Classes'
instance Ord Word -- Defined in 'GHC.Classes'
instance Ord Order -- Defined in 'GHC.Classes'
instance Ord Int -- Defined in 'GHC.Classes'
instance Ord Float -- Defined in 'GHC.Classes'
instance Ord Double -- Defined in 'GHC.Classes'
instance Ord a => Ord (Maybe a) -- Defined in 'GHC.Classes'
instance Ord a => Ord (a -> b) -- Defined in 'GHC.Classes'
instance Ord a => Ord (a -> b -> c) -- Defined in 'GHC.Classes'
instance Ord a => Ord (a -> b -> c -> d) -- Defined in 'GHC.Classes'
instance Ord a => Ord (a -> b -> c -> d -> e) -- Defined in 'GHC.Classes'
instance Ord a => Ord (a -> b -> c -> d -> e -> f) -- Defined in 'GHC.Classes'
instance Ord a => Ord (a -> b -> c -> d -> e -> f -> g) -- Defined in 'GHC.Classes'
instance Ord a => Ord (a -> b -> c -> d -> e -> f -> g -> h) -- Defined in 'GHC.Classes'
instance Ord a => Ord (a -> b -> c -> d -> e -> f -> g -> h -> i) -- Defined in 'GHC.Classes'
instance Ord a => Ord (a -> b -> c -> d -> e -> f -> g -> h -> i -> j) -- Defined in 'GHC.Classes'
instance Ord a => Ord (a -> b -> c -> d -> e -> f -> g -> h -> i -> j -> k) -- Defined in 'GHC.Classes'
instance Ord a => Ord (a -> b -> c -> d -> e -> f -> g -> h -> i -> j -> k -> l) -- Defined in 'GHC.Classes'
instance Ord a => Ord (a -> b -> c -> d -> e -> f -> g -> h -> i -> j -> k -> l -> m) -- Defined in 'GHC.Classes'
instance Ord a => Ord (a -> b -> c -> d -> e -> f -> g -> h -> i -> j -> k -> l -> m -> n) -- Defined in 'GHC.Classes'
instance Ord a => Ord (a -> b -> c -> d -> e -> f -> g -> h -> i -> j -> k -> l -> m -> n -> o) -- Defined in 'GHC.Classes'
--
```

Annotations on the image include:

- Red text: "Assignment Project Exam Help" and "https://powcoder.com" overlaid on the code.
- Red text: "Add WeChat powcoder" overlaid on the code.
- Green arrows pointing to the line `instance Ord Int -- Defined in 'GHC.Classes'` from the left and right.
- A green box containing the text: "Int is an instance of Ord type class, so when we made our function args explicitly Int, we were OK".

How About This?

Assignment Project Exam Help
<https://powcoder.com>
Add WeChat powcoder:



The image shows two windows side-by-side. The left window is Notepad++ editing 'C:\HaskellCode\Test.hs'. The code defines a function `cmp2` with a type signature `cmp2 :: Num a => a -> a -> [Char]` and a body that uses `<` and `>` operators. The right window is WinGHCi, showing the compilation of 'Test.hs' and a type error: 'Could not deduce (Ord a) arising from a use of '<'' from the context: `Num a` bound by the type signature for: `cmp2 :: forall a. Num a => a -> a -> [Char]`.

```
1 module Test where
2
3 cmp2 :: Num a => a -> a -> [Char]
4 cmp2 x y | x < y = "First is smaller"
5          | x > y = "Second is smaller"
6          | otherwise = "Equal"
7
8
9
10
11
```

length: 1,269 | Ln: 10 Col: 2 Sel: 0 | 0 Windows (CR LF)

WinGHCi

[1 of 1] Compiling Test (Test.hs, interpreted)

• Could not deduce (Ord a) arising from a use of '<'' from the context: Num a bound by the type signature for: cmp2 :: forall a. Num a => a -> a -> [Char]

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

Prelude> :i Num
class Num a where
  (+) :: a -> a -> a
  (-) :: a -> a -> a
  (*) :: a -> a -> a
  negate :: a -> a
  abs :: a -> a
  signum :: a -> a
  fromInteger :: Integer -> a
  {-# MINIMAL (+), (*), abs, signum, fromInteger,
    (negate | (-)) #-}
  -- Defined in (GHC.Num)
```

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

Num type class does not define comparison!

Hmmm...

Num doesn't have comparison, **Ord** doesn't have addition

The image shows two windows. The left window is Notepad++ editing 'Test.hs' with the following code:

```
1 module Test where
2
3 cmp2 x y | x+1 < y+1 = "First is smaller"
4          | x+1 > y+1 = "Second is smaller"
5          | otherwise = "Equal"
6
7
8
9
10
11
```

The right window is WinGHCi showing the compilation and loading of the module:

```
Prelude> :reload
[1 of 1] Compiling Test
( Test.hs, interpreted )
OK, one module loaded.
*Test> |
```

Overlaid on the image is a red watermark that reads "Assignment Project Exam Help" and "https://powcoder.com Add WeChat powcoder".

Below the code windows, an orange-bordered box contains the text: "It compiled and loaded, what type did Haskell infer for x and y?"

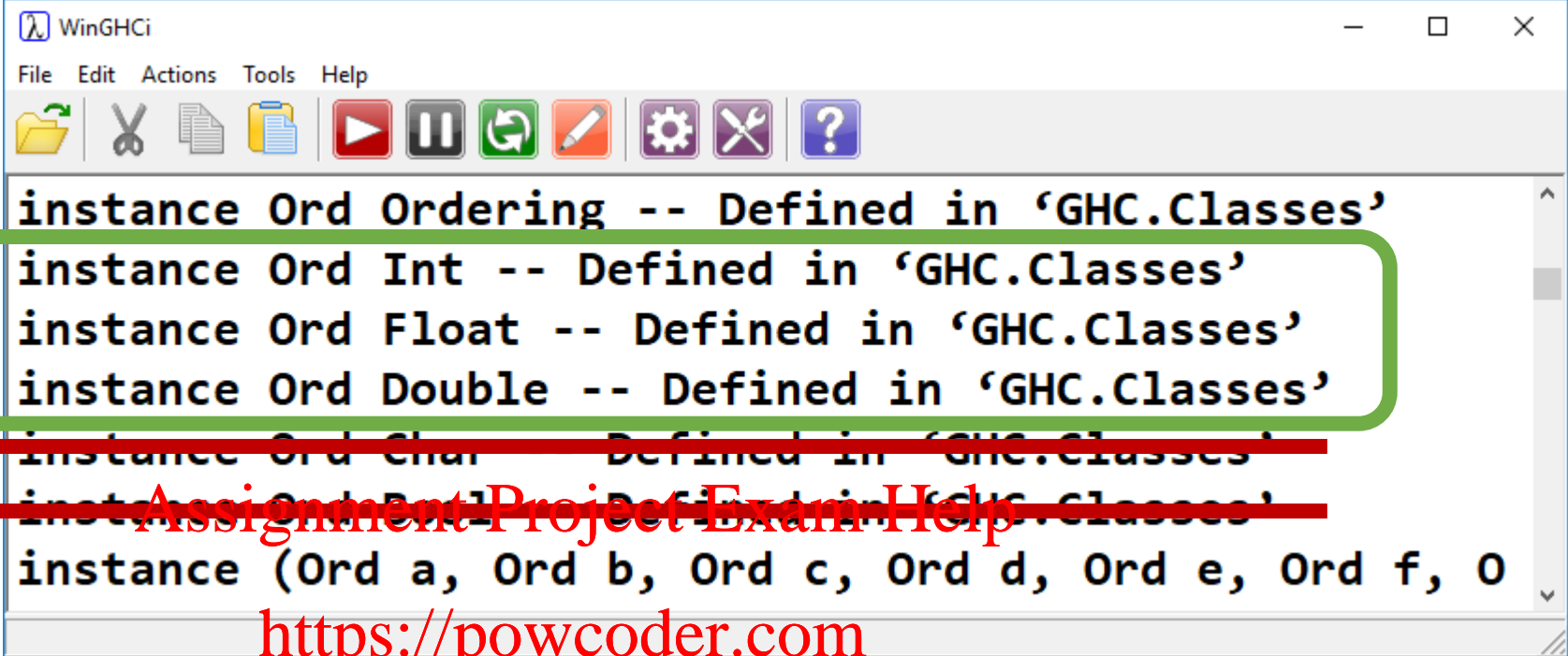
The image shows a screenshot of the WinGHCi window. The title bar says 'WinGHCi'. The menu bar includes 'File', 'Edit', 'Actions', 'Tools', and 'Help'. The toolbar contains icons for file operations (folder, copy, paste, save), execution (play, pause, refresh), and settings (gear, wrench, question mark). The main text area shows the following Haskell code and its output:

```
Prelude> :reload
[1 of 1] Compiling Test           ( Test.hs, interpreted )
Ok, one module loaded.
*Test> :t cmp2
cmp2 :: (Ord a, Num a) => a -> a -> [Char]
*Test>
```

Overlaid on the image are several red text annotations and a green-bordered box:

- A large red text 'Assignment Project Exam Help' is centered over the code.
- A red text '<https://powcoder.com>' is positioned below the first red text.
- A red text 'Add WeChat powcoder' is positioned below the URL.
- A green-bordered box contains the following text:
 - Both!**
 - Whatever type we pass in (**a**), it must be an instance of both Ord and Num.
 - **Int** is one such type, as is **Float**

Ord:



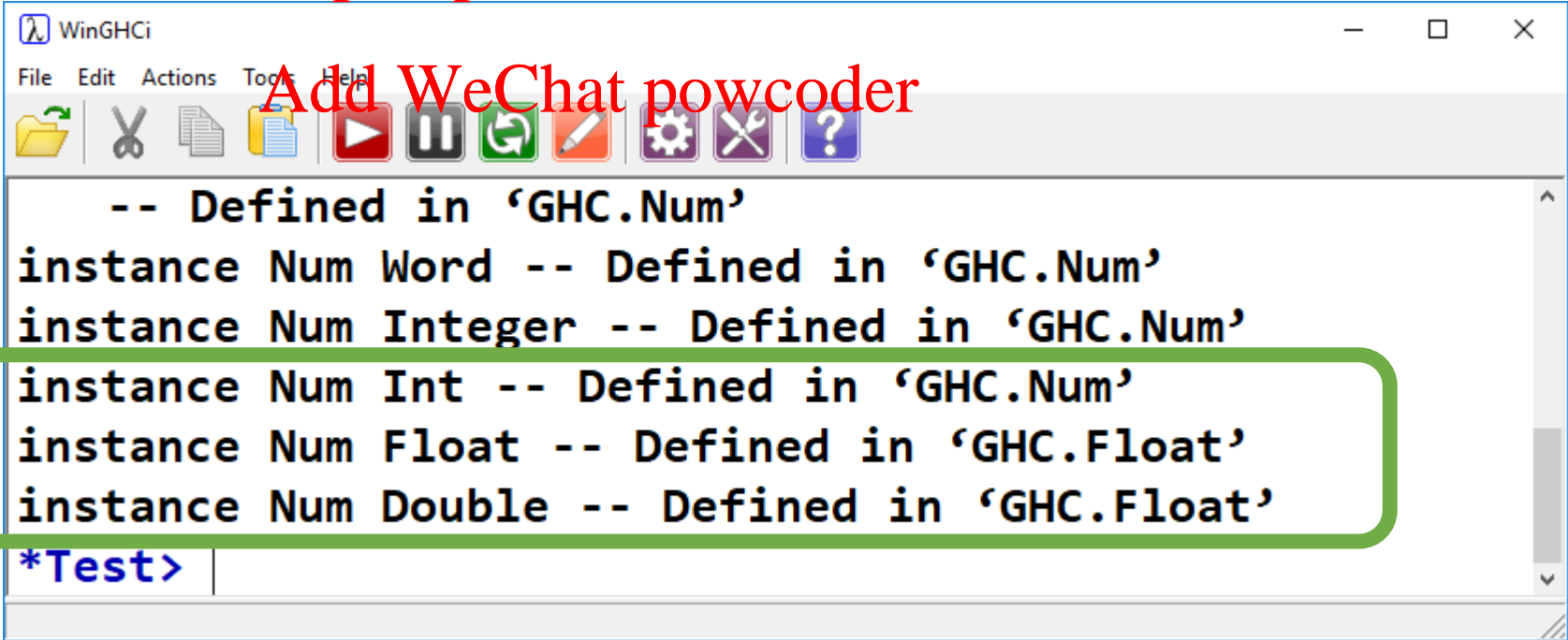
WinGHCi

File Edit Actions Tools Help

instance Ord Ordering -- Defined in 'GHC.Classes'
instance Ord Int -- Defined in 'GHC.Classes'
instance Ord Float -- Defined in 'GHC.Classes'
instance Ord Double -- Defined in 'GHC.Classes'
~~instance Ord Char -- Defined in 'GHC.Classes'~~
~~instance Ord Bool -- Defined in 'GHC.Classes'~~
instance (Ord a, Ord b, Ord c, Ord d, Ord e, Ord f, Ord g, Ord h, Ord i, Ord j, Ord k, Ord l, Ord m, Ord n, Ord o, Ord p, Ord q, Ord r, Ord s, Ord t, Ord u, Ord v, Ord w, Ord x, Ord y, Ord z) => Ord (a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s, t, u, v, w, x, y, z)

<https://powcoder.com>

Num:



WinGHCi

File Edit Actions Tools Help

-- Defined in 'GHC.Num'
instance Num Word -- Defined in 'GHC.Num'
instance Num Integer -- Defined in 'GHC.Num'
instance Num Int -- Defined in 'GHC.Num'
instance Num Float -- Defined in 'GHC.Float'
instance Num Double -- Defined in 'GHC.Float'

*Test>

Custom Data Types

Assignment Project Exam Help

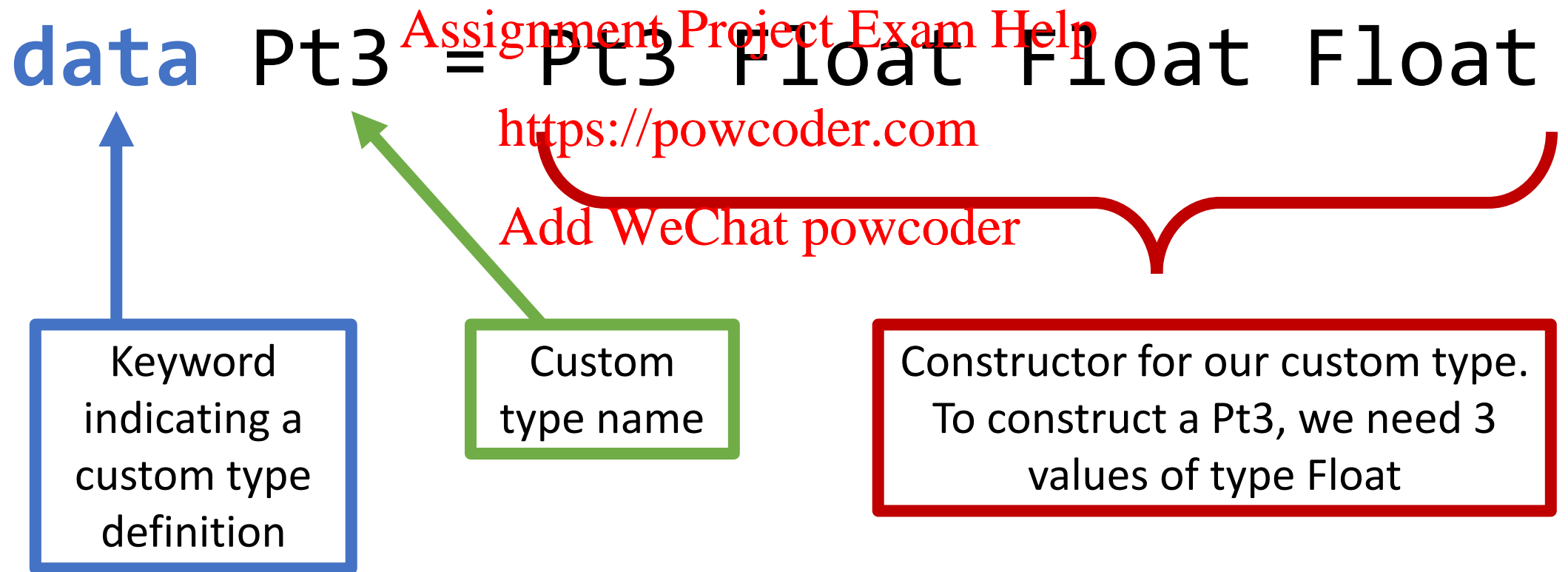
<https://powcoder.com>

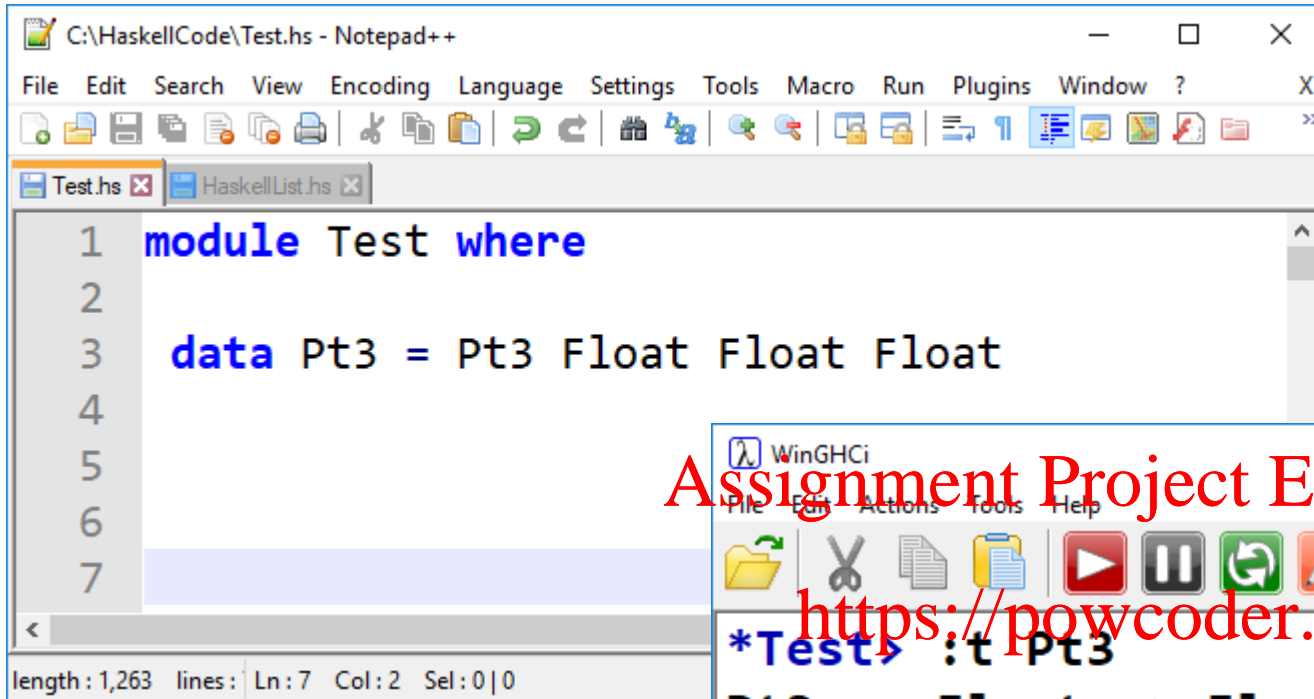
Add WeChat powcoder

Custom Data Types

- Lists and tuples are already quite powerful for organizing data
- What if we want to add custom behaviors over our data?
- For example, we can declare a pair tuple (1, 2).
- What if we want to treat these as coordinates and compute the sum? The dot product? Etc.?
- Addition is not defined for tuples, let alone more complicated operations.

Custom Coordinate Types

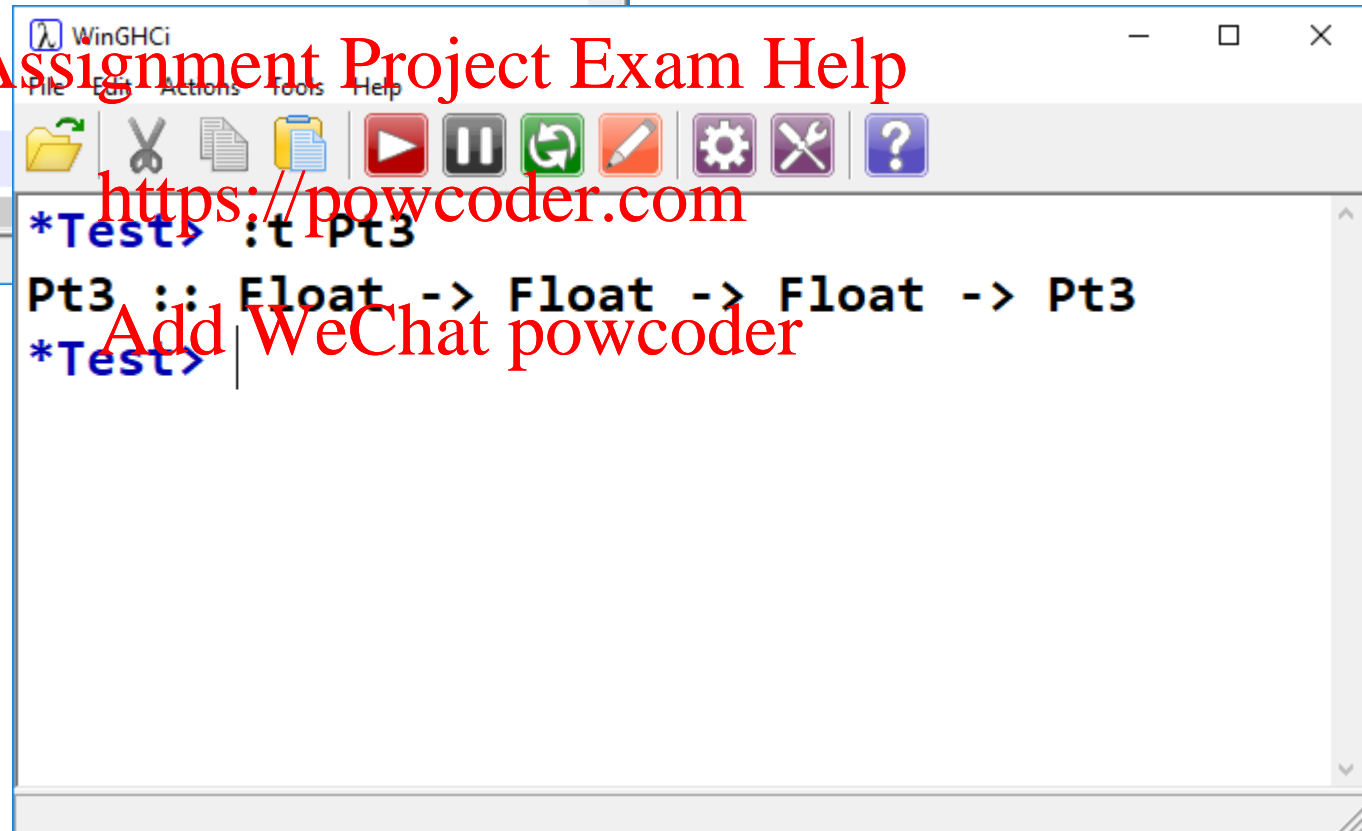




A screenshot of the Notepad++ text editor. The title bar shows 'C:\HaskellCode\Test.hs - Notepad++'. The menu bar includes File, Edit, Search, View, Encoding, Language, Settings, Tools, Macro, Run, Plugins, Window, and ?. The toolbar contains various icons for file operations and editing. Two tabs are open: 'Test.hs' and 'HaskellList.hs'. The code in 'Test.hs' is as follows:

```
1 module Test where
2
3 data Pt3 = Pt3 Float Float Float
4
5
6
7
```

The status bar at the bottom indicates 'length: 1,263 lines: Ln: 7 Col: 2 Sel: 0|0'.

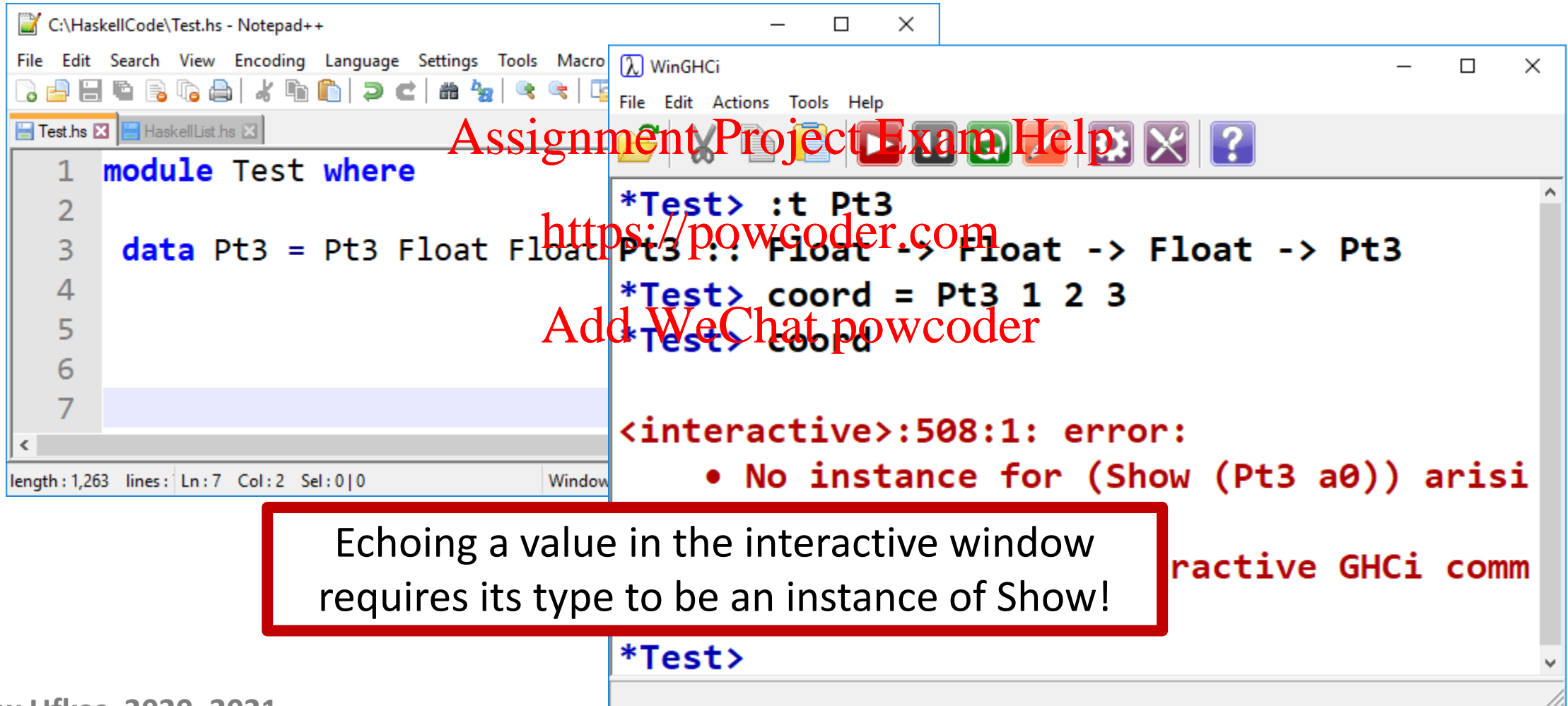


A screenshot of the WinGHCi terminal window. The title bar shows 'WinGHCi'. The menu bar includes File, Edit, Actions, Tools, and Help. The toolbar contains icons for file operations, running, and debugging. The terminal output is as follows:

```
*Test> :t Pt3
Pt3 :: Float -> Float -> Float -> Pt3
*Test>
```

Custom Type Usage

Assignment Project Exam Help
<https://powcoder.com>
Add WeChat powcoder



The image shows two windows. The left window, Notepad++, displays a Haskell module 'Test' with a custom type 'Pt3' defined as a function type: `data Pt3 = Pt3 Float Float Pt3 :: Float -> Float -> Float -> Pt3`. The right window, WinGHCi, shows the interactive session. It first defines `coord = Pt3 1 2 3`. Then, the user enters `*Test> coord`, which results in a type error: `<interactive>:508:1: error: • No instance for (Show (Pt3 a0)) arising from a call to `print` at <interactive>:508:1-2:1-2: Note: `print` is defined in `Ghci1`, imported from `Prelude``. A red box highlights the text: 'Echoing a value in the interactive window requires its type to be an instance of Show!'. The WinGHCi window also shows the prompt `*Test>`.

```
1 module Test where
2
3 data Pt3 = Pt3 Float Float Pt3 :: Float -> Float -> Float -> Pt3
4
5
6
7
```

```
*Test> :t Pt3
Pt3 :: Float -> Float -> Float -> Pt3
*Test> coord = Pt3 1 2 3
*Test> coord

<interactive>:508:1: error:
• No instance for (Show (Pt3 a0)) arising from a call to `print` at <interactive>:508:1-2:1-2:
Note: `print` is defined in `Ghci1`, imported from `Prelude`

*Test>
```

Echoing a value in the interactive window
requires its type to be an instance of Show!

active GHCi comm

Hmmm...

- The values contained in Pt3 are Float, and we know that Float is an instance of Show.
- How can we access the individual elements of Pt3?

```
-- Defined in 'GHC.Show'
instance (Show a, Show b) => Show (a, b) -- Defined
in 'GHC.Show'
instance Show () -- Defined in 'GHC.Show'
instance Show Float -- Defined in 'GHC.Float'
instance Show Double -- Defined in 'GHC.Float'
*Test>
```

```
C:\HaskellCode\Test.hs - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window
?
Test.hs HaskellList.hs
1 module Test where
2
3 data Pt3 = Pt3 Float Float Float
4
5 ptX (Pt3 x y z) = x
6 ptY (Pt3 x y z) = y
7 ptZ (Pt3 x y z) = z
8
```

```
WinGHCi
File Edit Actions Tools Help
*Test> coord = Pt3 1 2 3
*Test> ptX coord
1.0
*Test> ptY coord
2.0
*Test> ptZ coord
3.0
*Test> |
```

- Three access functions, one for each of the three values.
- Take as arguments Pt3 (and by extension its three members)
- Return x, y, or z coordinate respectively.

Overloading Constructor

```
C:\HaskellCode\Test.hs - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
Test.hs HaskellList.hs
1 module Test where
2
3 data Pt = Pt3 Float Float Float
4         | Pt2 Float Float
5
6 ptX (Pt3 x y z) = x
7 ptY (Pt3 x y z) = y
8 ptZ (Pt3 x y z) = z
9
10
11
12
13
length: 1,358 lines: 78 Ln: 12 Col: 2 Sel: 0|0 Windows (CR LF) UTF-8 INS
```

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

- Define Pt3 with three parameters
- Define Pt2 with two parameters
- Name of our data type is now simply Pt, because we have made it more generic.

There is now a problem with our access functions

There is now a problem with our access functions.

```
*C:\HaskellCode\Test.hs - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ? X
Test.hs HaskellList.hs
1 module Test where
2
3 data Pt = Pt3 Float Float Float
4         | Pt2 Float Float
5
6 ptX (Pt2 x _) = x
7 ptX (Pt3 x _ _) = x
8
9 ptY (Pt2 _ y) = y
10 ptY (Pt3 _ y _) = y
11
12 ptZ (Pt3 _ _ z) = z
13
14
length: 1,404 lines: 8 Ln: 14 Col: 2 Sel: 0 | 0 Windows (CR LF) UTF-8 INS
```

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

Now our access
functions work for
both Pt2 and Pt3

```
WinGHCi
File Edit Actions Tools Help
*Test> coord2 = Pt2 3 4
*Test> coord3 = Pt3 5 6 7
*Test> ptX coord2
3.0
*Test> ptX coord3
5.0
*Test> ptY coord3
6.0
*Test> ptY coord2
4.0
*Test>
```

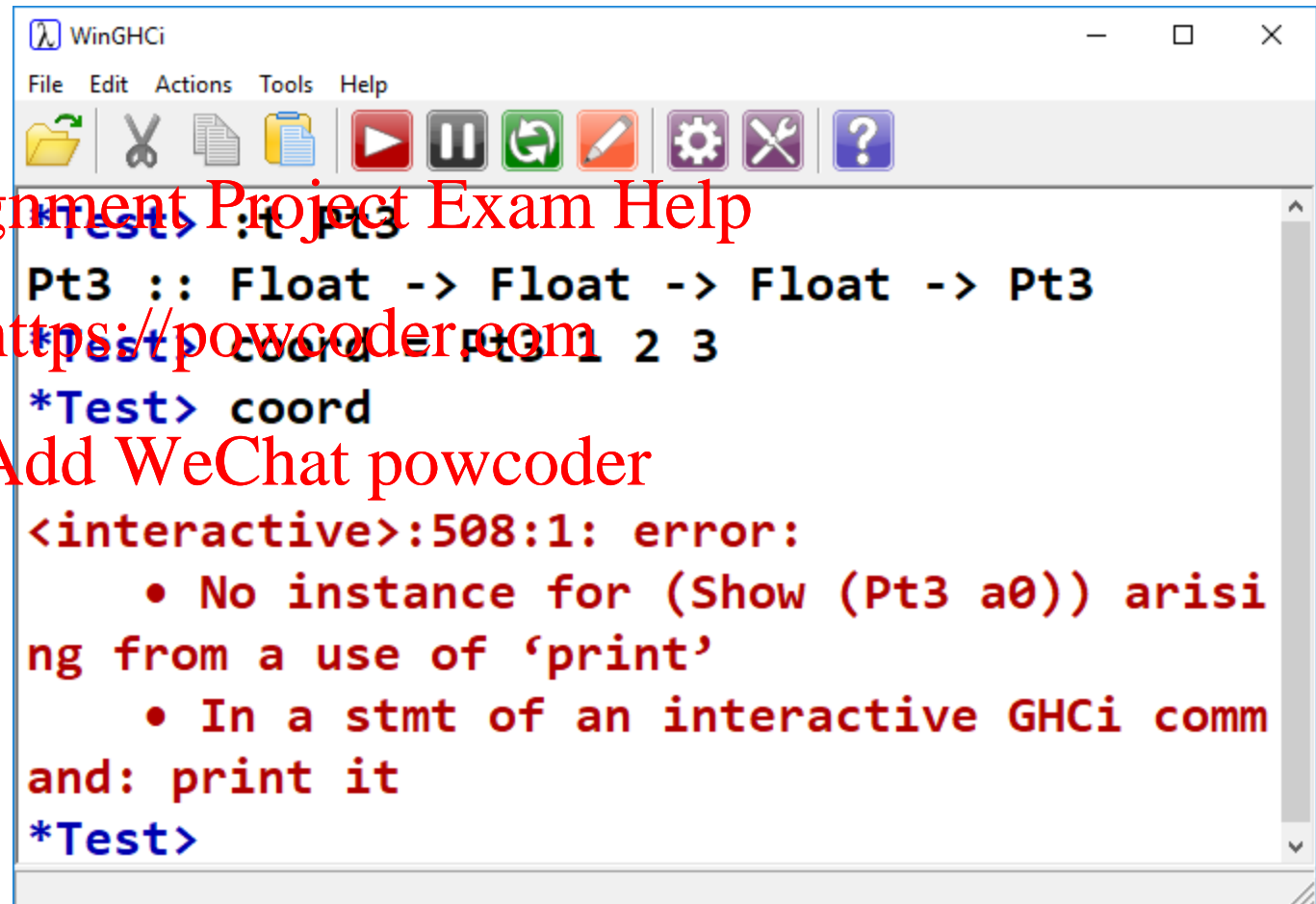
Deriving Show

Recall:

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder



```
WinGHCi
File Edit Actions Tools Help
[Icons]
*Test> :t Pt3
Pt3 :: Float -> Float -> Float -> Pt3
*Test> coord = Pt3 1 2 3
*Test> coord
<interactive>:508:1: error:
    • No instance for (Show (Pt3 a0)) arising from a use of 'print'
    • In a stmt of an interactive GHCi command: print it
*Test>
```

Deriving Show

```
*C:\HaskellCode\Test.hs - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins
Test.hs HaskellList.hs
1 module Test where
2
3 data Pt = Pt3 Float Float Float
4         | Pt2 Float Float
5         deriving (Show)
6
7
11
12 ptY (Pt2 v) = v
length: 1,437 lines: 8 Ln: 8 Col: 2 Sel: 0|0 Windows (CR LF) UTF
```

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

Our custom type will inherit some default display behavior from **Show**

```
WinGHCi
File Edit Actions Tools Help
*Test> c2 = Pt2 1 2
*Test> c3 = Pt3 5 6 7
*Test> c2
Pt2 1.0 2.0
*Test> c3
Pt3 5.0 6.0 7.0
*Test> |
```

Similar to the toString() method in Java!

More Advanced Functions

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

```
C:\HaskellCode\Test.hs - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
Test.hs HaskellList.hs
1 module Test where
2
3 data Pt = Pt3 Float Float Float
4         | Pt2 Float Float
5         deriving (Show)
6
7 vecLen (Pt2 x y) = sqrt(x^2 + y^2)
8 vecLen (Pt3 x y z) = sqrt(x^2 + y^2 + z^2)
9
10
11
12
length: 1,516 lines: 86 Ln: 14 Col: 21 Sel: 0 | 0 Windows (CR LF) UTF-8
```

Compute length of Pt2 and Pt3,
treating them as vectors

```
WinGHCi
File Edit Actions Tools Help
*Test> c2 = Pt2 1 2
*Test> c3 = Pt3 5 6 7
*Test> vecLen c2
2.236068
*Test> vecLen c3
10.488089
*Test> |
```

Addition, Subtraction, Equality?

```
C:\HaskellCode\Test.hs - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
Test.hs HaskellList.hs
1 module Test where
2
3 data Pt = Pt3 Float Float Float
4         | Pt2 Float Float
5         deriving (Show)
6
7 ptAdd (Pt2 x1 y1) (Pt2 x2 y2) =
8     Pt2 (x1+x2) (y1+y2)
9
10 ptSub (Pt2 x1 y1) (Pt2 x2 y2) =
11     Pt2 (x1-x2) (y1-y2)
12
13 ptEq (Pt2 x1 y1) (Pt2 x2 y2) =
14     (x1 == x2) && (y1 == y2)
15
```

length: 1705, lines: 16, 17 Col: 27 Sel: 0|0 Windows (CR LF) UTF-8 INS

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

Let's add more functions!

- We can very easily define addition as the sum of each respective X and Y coord
- Likewise for subtraction and equality.

Addition, Subtraction, Equality?

The image shows a Haskell program in a Notepad++ window and its execution in a WinGHCi window. The Haskell code defines a point type and functions for addition, subtraction, and equality. The WinGHCi window shows the results of these functions being called with specific values.

```
C:\HaskellCode\Test.hs - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
Test.hs HaskellList.hs
1 module Test where
2
3 data Pt = Pt3 Float Float Float
4         | Pt2 Float Float
5         deriving (Show)
6
7 ptAdd (Pt2 x1 y1) (Pt2 x2 y2) =
8     Pt2 (x1+x2) (y1+y2)
9
10 ptSub (Pt2 x1 y1) (Pt2 x2 y2) =
11     Pt2 (x1-x2) (y1-y2)
12
13 ptEq (Pt2 x1 y1) (Pt2 x2 y2) =
14     (x1 == x2) && (y1 == y2)
15

WinGHCi
File Edit Actions Tools Help
*Test> c1 = Pt2 4 2
*Test> c2 = Pt2 (-1) 3
*Test> ptAdd c1 c2
Pt2 3.0 5.0
*Test> ptSub c1 c2
Pt2 5.0 (-1.0)
*Test> ptEq c1 c2
False
*Test> ptEq c1 (Pt2 4 2)
True
*Test>
```

Assignment Project Exam Help
<https://powcoder.com>
Add WeChat powcoder

length: 1705, lines: 16, Col: 27 Sel: 0|0 Windows (CR LF) UTF-8 INS

Addition, Subtraction, Equality?

This seems very clunky. Why can't we simply add, subtract, or check equality with the symbolic operators (+, -, ==)?

<https://powcoder.com>

We can! Equality is defined for instances of type class **Eq**

+, -, etc. are defined for instances of type class **Num**.

How do we make Pt2 and Pt3 instances of another type class?

Custom Types & Type Classes

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

```
1 module Test where
2
3 data Pt = Pt3 Float Float Float
4         | Pt2 Float Float
5         deriving (Show)
6
7
8
9 instance Eq Pt where
10     (Pt2 x1 y1) == (Pt2 x2 y2) = (x1==x2 && y1==y2)
11
12
```

Declare **Pt** to be an instance of **Eq**

Define what it means for two Pt2 values to be considered equal

Haskell length : 1,623 lines : 99 Ln : 14 Col : 2 Sel : 0 | 0 Windows (CR LF) UTF-8 INS

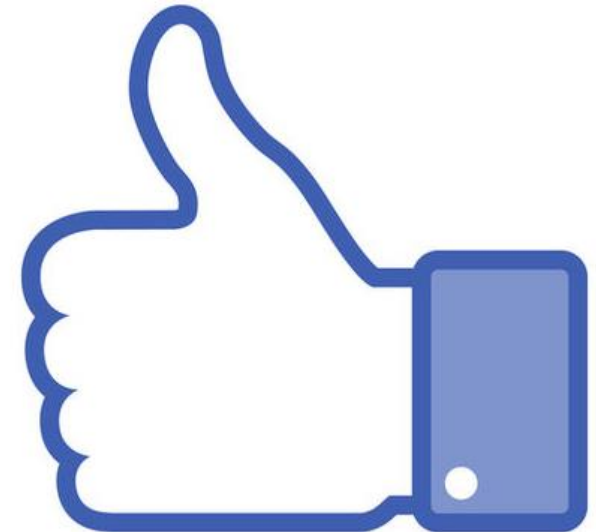
```
C:\HaskellCode\Test.hs - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Win
Test.hs HaskellList.hs
1 module Test where
2
3 data Pt = Pt3 Float Float Float
4         | Pt2 Float Float
5         deriving (Show)
6
7
8
9 instance Eq Pt where
10     (Pt2 x1 y1) == (Pt2 x2 y2) = (x1==x2 && y1==y2)
11
12
Haskell length: 1,623 lines: 99 Ln: 14 Col: 2 Sel: 0|0 Windows (CR LF) UTF-8 INS
```

```
WinGHCi
File Edit Actions Tools Help
*Test> Pt2 1 2 == Pt2 2 3
False
*Test> Pt2 1 2 == Pt2 1 2
True
*Test>
```

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder



Minimal Definition

The screenshot shows the WinGHCi window with the following content:

```
*Test> :i Eq  
class Eq a where  
    (==) :: a -> a -> Bool  
    (/=) :: a -> a -> Bool  
{-# MINIMAL (==) | (/=) #-}  
-- Defined in 'GHC.Classes'  
instance [safe] Eq Pt -- Defined at Test.hs:7:11  
instance (Eq a) => Eq (List a) -- Defined in 'Data.List'  
instance Eq a => Eq (Maybe a)
```

Overlaid on the image are two red annotations:

- A large red "0" circled around the minimal definition line.
- Red text: "Assignment Project Exam Help" and "https://powcoder.com".

In the bottom right corner, there is a white box containing a black QR code and the text "Add WeChat powcoder".

```
C:\HaskellCode\Test.hs - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
Test.hs HaskellList.hs
1 module Test where
2
3 data Pt = Pt3 Float Float Float
4         | Pt2 Float Float
5         deriving (Show)
6
7
8 instance Eq Pt where
9     (Pt2 x1 y1) == (Pt2 x2 y2) = (x1==x2 && y1==y2)
10
11
12
```

```
WinGHCi
File Edit Actions Tools Help
*Test> c1 = Pt2 2 3
*Test> c2 = Pt2 2 4
*Test> c1 == c2
False
*Test> c1 /= c2
True
*Test>
```

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

Haskell is clever enough to
derive /= from our definition of
==, and vice versa.

Let's Add /= Anyway

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

```
C:\HaskellCode\Test.hs - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
Test.hs HaskellList.hs
1 module Test where
2
3 data Pt = Pt3 Float Float Float
4         | Pt2 Float Float
5         deriving (Show)
6
7
8 instance Eq Pt where
9     (Pt2 x1 y1) == (Pt2 x2 y2) = (x1==x2 && y1==y2)
10    (Pt2 x1 y1) /= (Pt2 x2 y2) = not (x1==x2 && y1==y2)
11
12
Haskell length: 1,674 lines: 98 Ln: 12 Col: 2 Sel: 0|0 Windows (CR LF) UTF-8 INS
```

```
WinGHCi
File Edit Actions Tools Help
*Test> c1 = Pt2 2 3
*Test> c2 = Pt2 2 4
*Test> c1 == c2
False
*Test> c1 /= c2
True
*Test> |
```

By The Way...



This should remind you of interfaces in Java

Assignment Project Exam Help

If you create a Java class that implements an interface, it must define all method signatures present in that interface

<https://powcoder.com>

Add WeChat powcoder

Most common is the **Comparable** interface

If our Java class implements Comparable, we must define some way of comparing two instances of our class

Instance of Num

Assignment Project Exam Help
<https://powcoder.com>
Add WeChat powcoder

```
C:\HaskellCode\Test.hs - Notepad++
File Edit Search View Encoding Language Settings Tools
Test.hs HaskellList.hs
1 module Test where
2
3 data Pt = Pt3 Float Float
4         | Pt2 Float Float
5         deriving (Show)
6
7 instance Num Pt where
8     (Pt2 x1 y1) + (Pt2 x2 y2) = Pt2 (x1 + x2) (y1 + y2)
9
10
11
12
Hasl length: 1,765 lines: 104 Ln: 11 Col: 3 Sel: 0|0
```

WinGHCi
File Edit Actions Tools Help
[1 of 1] Compiling Test (Test.hs, interpreted)
Test.hs:7:11: warning: [-Wmissing-methods]
• No explicit implementation for
 ‘*’, ‘abs’, ‘signum’, ‘fromInteger’, and
(either ‘negate’ or ‘-’)
• In the instance declaration for ‘Num Pt’
7 | instance Num Pt where
Ok, one module loaded.
*Test>

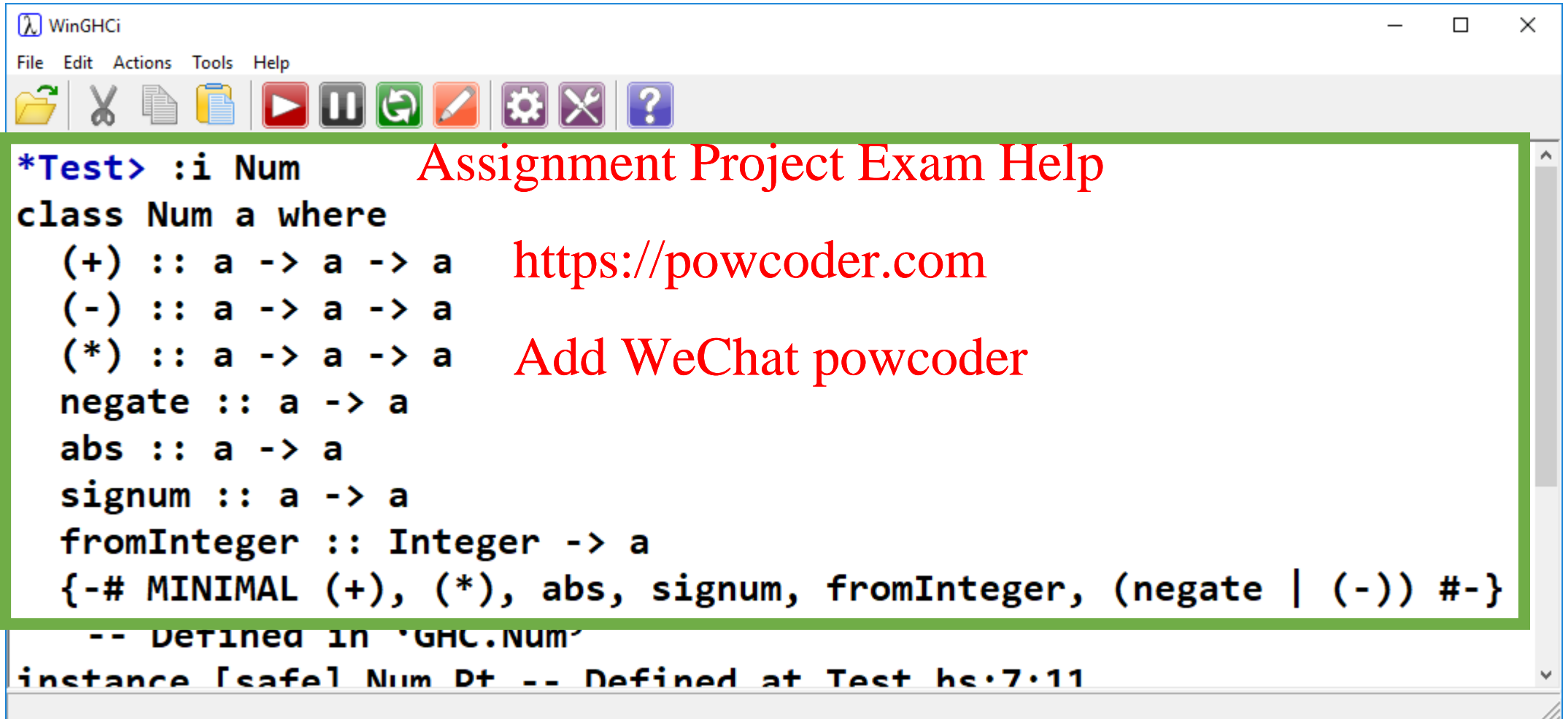
```
C:\HaskellCode\Test.hs - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
Test.hs HaskellList.hs
1 module Test where
2
3 data Pt = Pt3 Float Float Float
4         | Pt2 Float Float
5         deriving (Show)
6
7 instance Num Pt where
8     (Pt2 x1 y1) + (Pt2 x2 y2) = Pt2 (x1+x2) (y1+y2)
9
10
11
12
```

```
WinGHCi
File Edit Actions Tools Help
*Test> Pt2 1 2 + Pt2 3 4
Pt2 4.0 6.0
*Test> x = Pt2 1 2
*Test> y = Pt2 6 7
*Test> x+y
Pt2 7.0 9.0
*Test>
```

```
WinGHCi
File Edit Actions Tools Help
*Test> Pt3 1 2 3 + Pt3 1 2 3
*** Exception: Test.hs:8:3-49: Non-exhaustive patterns in function +
*Test> |
```

- We're only implementing for Pt2.
- Adding Pt3 follows the same pattern

Instance of Num



A screenshot of a WinGHCi window. The window has a title bar with the WinGHCi logo and standard window controls. Below the title bar is a menu bar with 'File', 'Edit', 'Actions', 'Tools', and 'Help'. Under the menu bar is a toolbar with icons for file operations (folder, copy, paste, save), execution (play, pause, refresh), and settings (gear, wrench, question mark). The main text area contains Haskell code for defining a Num instance. The code is as follows:

```
*Test> :i Num
class Num a where
  (+) :: a -> a -> a
  (-) :: a -> a -> a
  (*) :: a -> a -> a
  negate :: a -> a
  abs :: a -> a
  signum :: a -> a
  fromInteger :: Integer -> a
  {-# MINIMAL (+), (*), abs, signum, fromInteger, (negate | (-)) #-}
  -- Defined in 'GHC.Num'
instance [safe] Num D+ -- Defined at Test.hs:7:11
```

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

```
C:\HaskellCode\Test.hs - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
Test.hs HaskellList.hs
1 module Test where
2
3 data Pt = Pt3 Float Float Float
4         | Pt2 Float Float
5         deriving (Show)
6
7 instance Num Pt where
8   (Pt2 x1 y1) + (Pt2 x2 y2) = Pt2 (x1 + x2) (y1 + y2)
9   (Pt2 x1 y1) - (Pt2 x2 y2) = Pt2 (x1 - x2) (y1 - y2)
10  (Pt2 x1 y1) * (Pt2 x2 y2) = Pt2 (x1 * x2) (y1 * y2)
11  abs (Pt2 x1 y1) = Pt2 (abs x1) (abs y1)
12  signum (Pt2 x1 y1) = Pt2 (signum x1) (signum y1)
13
14 instance Eq Pt where
15   (Pt2 x1 y1) == (Pt2 x2 y2) = (x1==x2 && y1==y2)
16   (Pt2 x1 y1) /= (Pt2 x2 y2) = not (x1==x2 && y1==y2)
17
18
```

- This may look circular
- We're using abs and signum in our definition of abs and signum.
- However! x1 and y1 are Float.
- abs and signum are defined for Float
- We're defining them for Pt2

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

```
C:\HaskellCode\Test.hs - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
Test.hs HaskellList.hs
1 module Test where
2
3 data Pt = Pt3 Float Float Float
4         | Pt2 Float Float
5         deriving (Show)
6
7 instance Num Pt where
8     (Pt2 x1 y1) + (Pt2 x2 y2) = Pt2 (x1+x2) (y1+y2)
9     (Pt2 x1 y1) - (Pt2 x2 y2) = Pt2 (x1-x2) (y1-y2)
10    (Pt2 x1 y1) * (Pt2 x2 y2) = Pt2 (x1*x2) (y1*y2)
11    abs (Pt2 x1 y1) = Pt2 (abs x1) (abs y1)
12    signum (Pt2 x1 y1) = Pt2 (signum x1) (signum y1)
13    fromInteger n = let a = (fromInteger n) in Pt2 a a
14
15 instance Eq Pt where
16     (Pt2 x1 y1) == (Pt2 x2 y2) = (x1==x2 && y1==y2)
17     (Pt2 x1 y1) /= (Pt2 x2 y2) = not (x1==x2 && y1==y2)
```

- fromInteger is a *coercion* function.
- Dictates how our custom type can be created from an Integer
- Takes an Integer, returns a Pt
- Lets us do this...

Assignment Project Exam Help

<https://powcoder.com>

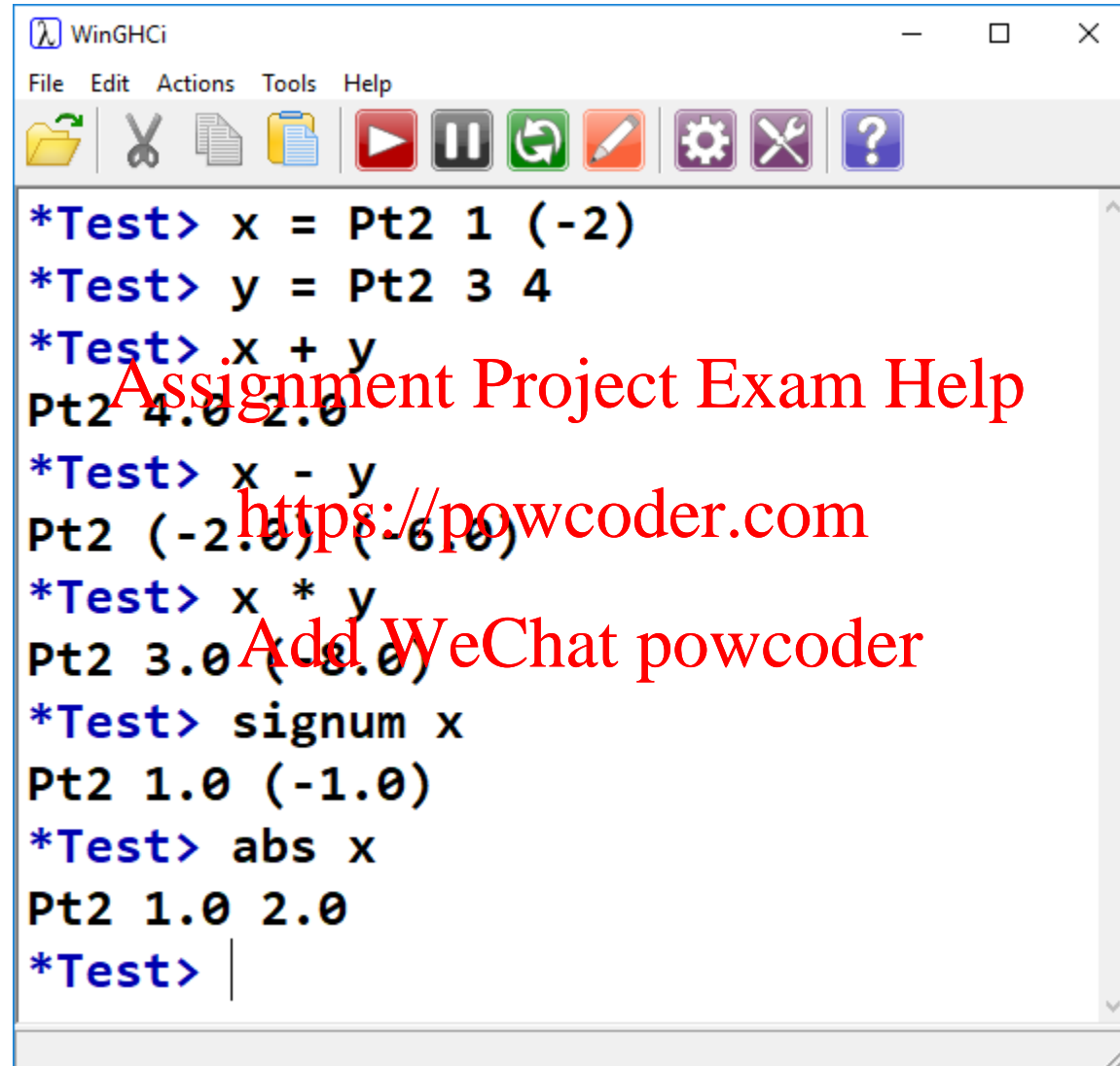
Add WeChat powcoder

```
WinGHCi
File Edit Actions Tools Help
*Test> (Pt2 2 3) + 4
Pt2 6.0 7.0
*Test>
```

```
C:\HaskellCode\Test.hs - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
Test.hs HaskellList.hs
1 module Test where
2
3 data Pt = Pt3 Float Float Float
4         | Pt2 Float Float
5         deriving (Show)
6
7 instance Num Pt where
8     (Pt2 x1 y1) + (Pt2 x2 y2) = Pt2 (x1+x2) (y1+y2)
9     (Pt2 x1 y1) - (Pt2 x2 y2) = Pt2 (x1-x2) (y1-y2)
10    (Pt2 x1 y1) * (Pt2 x2 y2) = Pt2 (x1*x2) (y1*y2)
11    abs (Pt2 x1 y1) = Pt2 (abs x1) (abs y1)
12    signum (Pt2 x1 y1) = Pt2 (signum x1) (signum y1)
13    fromInteger n = let a = (fromInteger n) in Pt2 a a
14
15 instance Eq Pt where
16     (Pt2 x1 y1) == (Pt2 x2 y2) = (x1==x2 && y1==y2)
17     (Pt2 x1 y1) /= (Pt2 x2 y2) = not (x1==x2 && y1==y2)
18
Haskell length: 2,004 lines: 105 Ln: 21 Col: 2 Sel: 0|0 Windows (CR LF) UTF-8 INS
```

```
WinGHCi
File Edit Actions Tools Help
*Test> :reload
[1 of 1] Compiling Test
( Test.hs, interpreted )
Ok, one module loaded.
*Test> |
```

No more warnings!

A screenshot of a WinGHCi window. The window has a title bar with the WinGHCi logo and standard window controls. Below the title bar is a menu bar with 'File', 'Edit', 'Actions', 'Tools', and 'Help'. Underneath the menu bar is a toolbar with icons for file operations (open, save, copy, paste), execution (run, pause, step over, step into), and settings (preferences, quit, help). The main text area contains the following Haskell code and its output:

```
*Test> x = Pt2 1 (-2)
*Test> y = Pt2 3 4
*Test> x + y
Pt2 4.0 2.0
*Test> x - y
Pt2 (-2.0) (-6.0)
*Test> x * y
Pt2 3.0 (-8.0)
*Test> signum x
Pt2 1.0 (-1.0)
*Test> abs x
Pt2 1.0 2.0
*Test> |
```

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

Instance of Show

In Java-speak, define our own toString(), instead of deriving the default



WinGHCi

File Edit Actions Tools Help

<https://powcoder.com>

```
*Test> :i Show
```

```
class Show a where
  showsPrec :: Int -> a -> ShowS
  show :: a -> String
  showList :: [a] -> ShowS
  {-# MINIMAL showsPrec | show #-}
```

Add WeChat powcoder

- The minimal definition for Show is easy
- Need to implement show OR showsPrec
- Let's do show
- Need to go from Pt2 to a String

```
-- Defined in 'GHC.Show'
instance [safe] Show Pt -- Defined at Test.hs:5:20
instance (Show a, Show b) => Show (Either a b)
-- Defined in 'Data.Either'
instance Show a => Show [a] -- Defined in 'GHC.Show'
```


C:\HaskellCode\Test.hs - Notepad++

File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?

Test.hs HaskellList.hs

```
1 module Test where
2
3 data Pt = Pt3 Float Float Float
4         | Pt2 Float Float
5         --deriving (Show)
6
7 instance Show Pt where
8     show (Pt2 x y) =
9         "< " ++ (show x) ++ ", " ++ (show y) ++ " >"
10
11
```

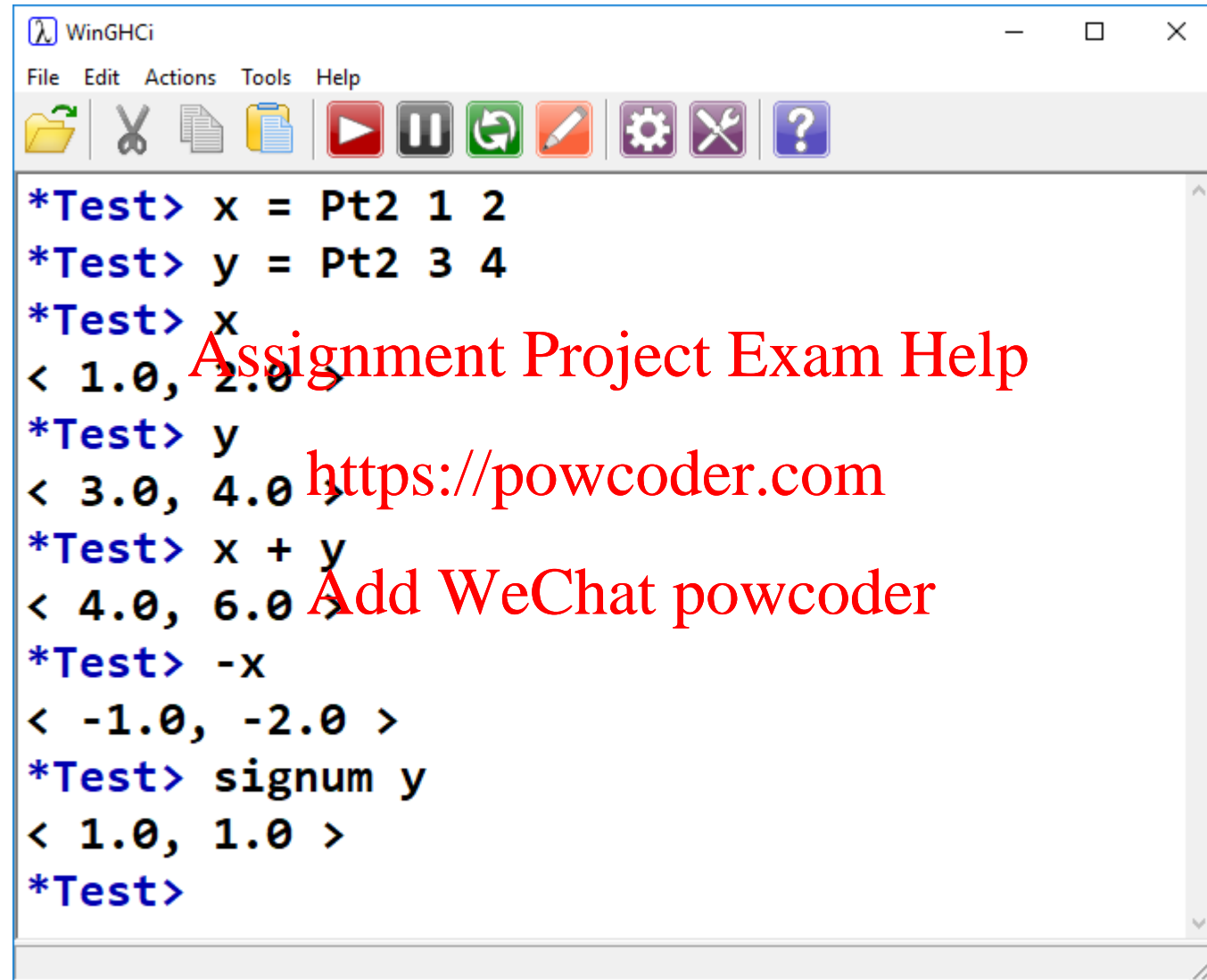
Assignment Project Exam Help

No longer need to derive Show, we've made our own

<https://powcoder.com>

Add WeChat powcoder

- Use string concatenation to create a pleasing visual output for Pt2
- In doing so, we make use of show as defined for Floats

A screenshot of a WinGHCi window. The window has a title bar with the WinGHCi logo and standard window controls. Below the title bar is a menu bar with 'File', 'Edit', 'Actions', 'Tools', and 'Help'. Underneath the menu bar is a toolbar with icons for file operations (open, save, copy, paste), execution (run, pause, refresh), editing (undo, redo), and settings (gear, wrench, help). The main text area contains the following Haskell code and its output:

```
*Test> x = Pt2 1 2
*Test> y = Pt2 3 4
*Test> x
< 1.0, 2.0 >
*Test> y
< 3.0, 4.0 >
*Test> x + y
< 4.0, 6.0 >
*Test> -x
< -1.0, -2.0 >
*Test> signum y
< 1.0, 1.0 >
*Test>
```

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

Haskell Tutorials/References:

https://en.wikibooks.org/wiki/Yet_Another_Haskell_Tutorial

<https://powcoder.com>

<http://cheatsheet.codeslower.com/CheatSheet.pdf>

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

