

Lecture 2C

COMP 1100



Acknowledgement of Country



- ✓ *I wish to acknowledge the traditional custodians of the land we are meeting on, the Ngunnawal people. I wish to acknowledge and respect their continuing culture and the contribution they make to the life of this city and this region. I would also like to acknowledge and welcome any other Aboriginal and Torres Strait Islander people who are enrolled in our courses.*



Interactive GHCi system

- ✓ The interactive GHCi system can be started from the terminal

```
artemlenskiy — ghc -B/Users/artemlenskiy/.ghcup/ghc/8.8.4
Last login: Mon Aug  3 15:53:24 on ttys006
[artemlenskiy@book ~] ghci
GHCi, version 8.8.4: https://www.haskell.org/ghc/  :? for help
Prelude>
```

- ✓ The GHCi prompt > indicates that the system is now waiting for the user to enter an expression to be evaluated

```
artemlenskiy — ghc -B/Users/artemlenskiy/.ghcup/ghc/8.8.4
[artemlenskiy@book ~] ghci
GHCi, version 8.8.4: https://www.haskell.org/ghc/  :? for help
[Prelude> 2 + 3 * 4
14
[Prelude> (2 + 3) * 4
20
[Prelude> sqrt (3^2 + 4^2)
5.0
Prelude>
```



Standard prelude

- ✓ Haskell comes with a large number of built-in functions

```
artemlenskiy — ghc -B/Users/artemlenskiy/.ghcup/ghc/8.8.4
[artemlenskiy@book ~] ghci
GHCi, version 8.8.4: https://www.haskell.org/ghc/  :? for help
[Prelude> head [1,2,3,4,5]
1
[Prelude> last [1,2,3,4,5]
5
```

Handwritten red annotations: checkmarks (✓) and crosses (✗) are placed around the code. A note in red says "last [1,2,3,4,5] → 5".

```
Prelude>
```



Function application

- ✓ Mathematical expressions vs Haskell expressions

$$\cancel{f(a, b)} + cd \Leftrightarrow f a b + c * d$$

Function application in Haskell is denoted silently using spacing, while the multiplication is denoted using the operator *

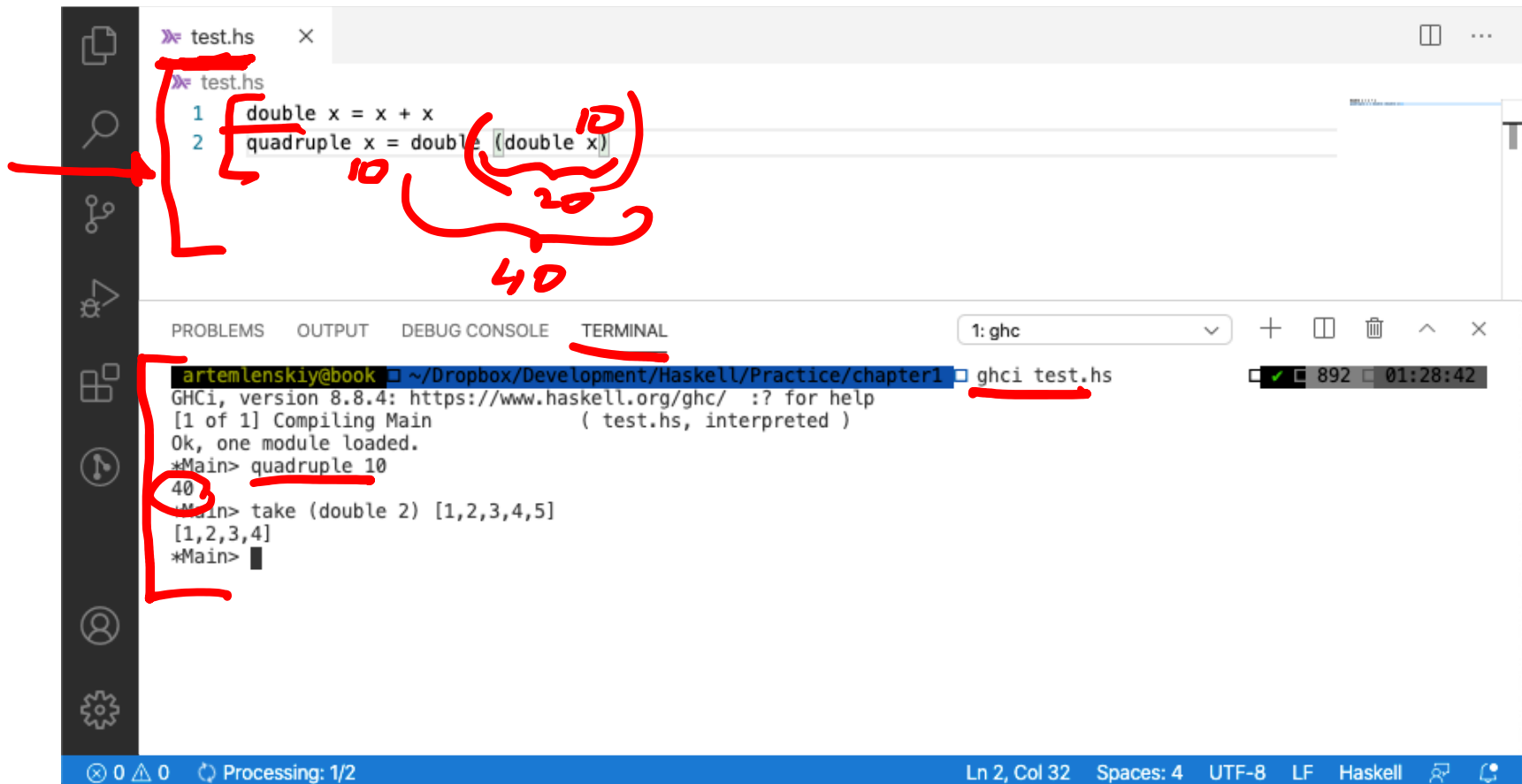
- ✓ More example

Mathematics	Haskell
$f(x)$	<code>f x</code>
$f(x, y)$	<code>f x y</code>
$f(g(x))$	<code>f (g x)</code>
$f(x, g(y))$	<code>f x (g y)</code>
$f(x)g(y)$	<code>f x * g y</code>

- ✓ Note that parentheses are still required e.g. `f x (g y)`

Haskell scripts: setup

- ✓ When running Haskell, keep two windows open:
 - ✓ An editor (e.g. VSCoDe) ←
 - ✓ GHCi



The screenshot shows a VS Code editor window with a Haskell file named `test.hs`. The code contains two lines:

```
1 double x = x + x
2 quadruple x = double (double x)
```

Handwritten red annotations in the editor show the calculation of `quadruple 10`. The value `10` is passed to `double`, which returns `20`. This `20` is then passed to another `double` function, resulting in `40`.

Below the editor is a terminal window titled `1: ghc`. The terminal shows the command `ghci test.hs` being executed. The output is:

```
artemlenskiy@book: ~/Dropbox/Development/Haskell/Practice/chapter1$ ghci test.hs
GHCi, version 8.8.4: https://www.haskell.org/ghc/ :? for help
[1 of 1] Compiling Main                 ( test.hs, interpreted )
Ok, one module loaded.
*Main> quadruple 10
40
*Main> take (double 2) [1,2,3,4,5]
[1,2,3,4]
*Main>
```

Handwritten red annotations in the terminal highlight the command `ghci test.hs` and the output `40`.

Haskell scripts: commands



```
test.hs x
test.hs
1 double x = x + x
2 quadruple x = double (double x)
3 factorial n = product [1..n]
4 average ns = sum ns `div` length ns

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
1: ghc
artemlenskiy@book ~/Dropbox/Development/Haskell/Practice/chapter1 ghci test.hs
GHCi, version 8.8.4: https://www.haskell.org/ghc/ :? for help
[1 of 1] Compiling Main
Ok, one module loaded.
*Main> :type factorial
factorial :: (Num a, Enum a) => a -> a
```

$[1..n] = [1, 2, \dots, n]$
 $n! = 1 * 2 * 3 * 4 * \dots * n$

command	meaning
<u>:load</u> name	load script name
<u>:reload</u>	reload current script
<u>:edit</u> name	edit script name
<u>:edit</u>	edit current script
<u>:type</u> <u>expr</u>	show type of expr
<u>:?</u>	Show all commands
<u>:quit</u>	quit GHCi

Haskell scripts: Naming requirements



- ✓ The names of the function and its arguments must begin with a lower-case letter, but can be followed by zero, more letters, digits, underscores and forward single quotes.

- ✓ **Examples:**

myFun

fun1

arg_2

x'

- ✓ The following list of keywords have a special meaning in the language

case	class	data	default	deriving
do	else	foreign	if	import
in	infix	infixl	infixr	instance
let	module	newtype	of	then
type	where			

- ✓ By convention, list arguments in Haskell usually have the suffix s on their name to indicate that they may contain multiple value.

✓
✓
✓

Haskell scripts: the layout rule

- ✓ The layout rule makes it possible to determine the grouping of definitions from their indentation.

✓ **Example:**

```
a = b + c
  where
    b = 1
    c = 2
d = a * 2
```

(Handwritten red annotations: "b = 10" above the code, "b = 1" and "c = 2" next to their respective lines, and various lines and arrows indicating the layout rule's application.)

It is clear that b and c are local definitions for use within the body of a.

- ✓ The grouping can be made explicitly using curly parentheses and semi-colon.

```
a = b + c
where {
  b = 1;
  c = 2;
d = a * 2
```

- ✓ Don't use tabs

Haskell scripts: comments

- ✓ Ordinary comments begin with the symbol `--`

```
-- Factorial of a positive integer:
factorial n = product [1..n]
-- Average of a list of integers:
average ns = sum ns `div` length ns
average ns = div (sum ns) (length ns)
```

$\{1, 2, 3, 4, 5\}$
15

$$\text{div}(a, b) := \frac{a}{b}$$

- ✓ Nested comments begin and with the symbols `{-` and `-}` and may span multiple lines

```
{-
double x = x + x
quadruple x = double (double x)
-}
```

$a \text{ 'div' } b$
 $\text{div } a \text{ } b$

Exercise

- ✓ The script below contains three syntactic errors. Correct these errors and then check that your script works properly using GHCi



```
N = a "div" length xs
  where
    a = 10
    xs = [1, 2, 3, 4, 5]
```

- ✓ The library function last selects the last element of a non-empty list;
 - For example: last [1, 2, 3, 4, 5] = 5Show how the function last could be defined in terms of the other library functions.