## What are the components of the Glasgow Haskell Compiler system?

- ghc, an optimized compiler for generating native code.
- ghci, an interactive interpreter and debugger.
- runghe, a program for running Haskell programs as scripts

without compilation.

### How can you view the available ghci commands?

:?

What is ghci's default prompt?

How can you change it?

ghci starts with Prelude>, being the standard pre-loaded library, and grows longer with each new loaded module or file.

The prompt can be changed with:

:set prompt NEWPROMPT

Real World Haskell

1

How do you add/remove modules in ghci?

Add: :module + NewModule
Remove: module - NewModule

The abbreviation: m also works.

# How can arithmetic operators be used in prefix form?

#### Use parens to get desired association:

```
ghci> (+) 3 5
8
ghci> (^) 3 5
243
```

## What's Haskell's quirk with negative numbers?

You nearly always need to surround the negative number with parens to get the desired association.

Describe Booleans in Haskell.

- The Boolean literals are True and False. Their type is Bool.
- Numbers and other types are not coerced into Boolean interpretations.

Describe negation in Haskell.

- Haskell uses /= for "not equal", instead of !=.
- Haskell uses not for "not", instead of !.

### How can you get information about a command in ghci?

:info funchame

### How can you create a temporary variable in ghci?

#### Use let.

let meaningOfLife = 42

How do you raise a power?

^ for integer exponents.

\*\* for floating point exponents.

Describe lists in Haskell.

They are homogeneous and use the common bracket

notation. Final commas are not allowed.

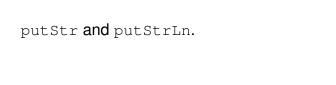
What is enumeration notation?

```
ghci> [1..5] [1,2,3,4,5]
```

ghci> ['a'..'j']

"abcdefghij"

How do you print to stdout?



What is a string, really?

#### A list of characters. [Char] and String are synonyms.

```
ghci> let lst = ['h', 'i']
ghci> lst
```

```
ghci> lst
"hi"
ghci> "" == []
```

True qhci>

How do you concatenate and build strings?

#### Using list operations.

```
ghci> 'a':"bc"
"abc"
ghci> "foo" ++ "bar"
"foobar"
```

## How can you get ghci to print the types of the expressions it evaluates?

How do you tell it to stop?

:set +t

:unset +t

What is it?

A special variable where ghci stores the last expression it returned.

1

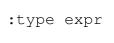
How can you construct rational numbers?

#### Use the % operator.

```
ghci> 11 % 29
11%29
it :: Ratio Integer
```

1

How can you find the type of an expression?



What is Haskell's comment syntax?

Lines beginning with -- are comments.