COMP105 Lecture 12

Types

Types

Everything has a type in Haskell

`:type` or `:t` will display the type of an expression

```
ghci> :t 'a'
'a' :: Char

ghci> :t True
True :: Bool

ghci> :t 1 + 1 == 2
1 + 1 == 2 :: Bool
```

Basic Number Types

Int holds 64-bit integers (between -2^{63} and $2^{63} - 1$)

- eg. `1`, `1000000`, `-5`

Integer holds arbitrary size integers

- eg. `265252859812191058636308480000000`

Float holds 32-bit floating point numbers

- eg. `0.5`, `1.0`, `3.14159`

Double holds 64-bit floating point numbers

- Same as float, but more accurate

Other Basic Types

Bool holds truth values

- either `True` or `False`

Char holds a single character

- eg. `'a'`, `'z'`, `'!'` `'é'`
- Char can store any unicode character

Tuple types

The type of a tuple is the type of its constituents

```
ghci> :t (True, 'a')
(True, 'a') :: (Bool, Char)
```

```
ghci> :t ('x', 'y', 'z')
('x', 'y', 'z') :: (Char, Char, Char)
```

Note

- The size of the tuple is encoded in its type
- Tuples elements can be of different types

List types

The type of a list is determined by the type of its elements

```
ghci> :t [True, False, False]
[True, False, False] :: [Bool]
```

```
ghci> :t "abcdef"
"abcdef" :: [Char]
```

Note

- A list of type `x` is denoted `[x]`
- This is why lists must contain elements of the same type
- The length is not encoded in the type

Types can get quite complex!

When you nest tuples and lists, you can get complex types

```
ghci> :t [(True, 'a'), (False, 'b')]
[(True, 'a'), (False, 'b')] :: [(Bool, Char)]
```

```
ghci> :t ["hello", "there"]
["hello", "there"] :: [[Char]]
```

```
ghci> :t ("hello", "there")
("hello", "there") :: ([Char], [Char])
```

Exercises

Without using `:type`, what are the types of the following values?

1. `[True, False, True]`

2. `([[True]], "hello")`

3. `[([False], False)]`