Parametric Polymorphism Part 2 Week 5 Tuesday

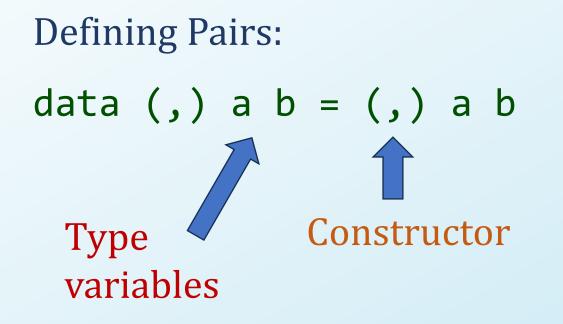
COMP1100/1130

Another Polymorphic Data Type: Tuples

Tuples can contain elements of any type.

Each of the elements can be of different types. Examples:

Recall the definition of Pairs



We usually write it as (a,b)

Definition of Tuples

Defining Tuples:

data	(,) a b = (,) a b	We usually write it as (a,b)
data	(,,) a b c = (,,) a b c	We usually write it as (a,b,c)
data	(,,,) a b c d = (,,,) a b c	d and so on
data	(,,,,) a b c d e = (,,,,) a	bcde
data	(,,,,,) a b c d e f = (,,,,	,) a b c d e f

Using Tuples

A function using tuples:

Can you write a function to get the middle item of a triple (or 3-tuple)?

Did we have to call it x above?

The Prelude contains several parametric polymorphic list functions, e.g.

- length :: [a] -> Int
- head :: [a] -> a

last returns the last
element of a list, e.g. last
[1,2,3] returns 3.

What is the type signature of tail? It returns the end part of a list, e.g. tail [1,2,3] returns [2,3].

init is similar to tail. It returns the first part of a list, e.g. init
[1,2,3] returns [1,2].

Insert an element into the front of a list:

: :: a -> [a] -> [a]

Join two lists together:

Return the element at the given position in the list (lists start at 0):

Make a given number of copies of an item.

replicate :: Int -> a -> [a]

Return a given number of elements of a list.

Remove a given number of elements from the front of a list.

drop :: Int -> [a] -> [a]

Some trickier ones:

concat :: [[a]] -> [a]

This concatenates a list of lists into a single list, e.g. concat [[1,2],[4,5],[3]] = [1,2,4,5,3]

splitAt :: Int -> [a] -> ([a],[a])

This splits a list at the given position. What does the return type ([a],[a]) mean?

Combine two lists into a list of pairs, where each pair is made up of an element from each list.

zip :: [a] -> [b] -> [(a,b)]

Example: zip [1,2,3] "bye" = [(1,'b'),(2,'y'),(3,'e')] Example: zip [1,2,3] [4,5,6] = [(1,4),(2,5),(3,6)] Example: zip [1,2] [4,5,6] = [(1,4),(2,5)]

Combine two lists into a list of pairs, where each pair is made up of an element from each list.

unzip :: [(a,b)] -> ([a],[b])

Example: unzip [(1,5),(2,6)] = ([1,2],[5,6])

Reverse a given list.

reverse :: [a] -> [a]

Example: reverse [1,2,3,4] = [4,3,2,1]

The Real Definitions in the Prelude

Some of the definitions are not quite what we've just seen, e.g.:

```
Length is not really:
length :: [a] -> Int
```

It's actually:

```
length :: Foldable t => t a -> Int
```

We'll learn this later. Now just replace t a with [a].

Monomorphic List Functions in the Prelude

There are monomorphic list functions too (only allow one type):

Conjunction of a list of Booleans: and :: [Bool] -> Bool Example: and [True, True] = True Disjunction of a list of Booleans: or :: [Bool] -> Bool

Example: or [True, False] = True

Ad-hoc Polymorphic List Functions in the Prelude

There are also ad-hoc polymorphic list functions in the Prelude, but we'll look at these later.