S1.4: Inductive Values and Types

CSci 2041:

Advanced Programming Principles

University of Minnesota,
Prof. Van Wyk,
Spring 2018

These slides were jointly developed by Gopalan Nadathur and Eric Van Wyk.

Exercise #1:Type abbreviations

Write down two values of each of the types defined below

```
type intandstr = int * string

type i_and_s_list = intandstr list
```

Exercise #2:Consider Homework 1

What opportunities are there to add type annontations with type synonyms to your functions?

Exercise #3:Enumerated Types

Define a type `weekday` that has as values the constants `Mon`, . . . , `Sun`

Identify a type amongst the base types that is actually an enumerated type like `color`

Exercise #4:Matching on enumerated types

Define the function

`isWorkDay : weekday -> bool`

that returns `true` just in the case that the argument represents a day between Monday and Friday

Make sure to use pattern matching over the `weekday` type in your definition

Exercise #5:Consider the following types:

```
type coord = float * float

type circ_desc = coord * float
type tri_desc = coord * coord * coord
type sqr_desc = coord * coord * coord * coord
```

The last three are meant to give us the components that characterize a circle, a triangle and a rectangle, respectively

- Define a type `shape` in OCaml that is capable of representing any one of a circle, a triangle and a rectangle
- Define a function of the following type
  `isRect : shape -> bool`
  with expected meaning.

Exercise #6:Type constructors

Pretty much *all* languages provide built in type constructors

Consider some language you are already familiar with and write down

- at least two type constructors in it,
- a concrete type constructed using each of those constructors, and
- a value corresponding to the concrete types you have described

Exercise #7:Total list head function

Write a `listHd` function that works even on empty lists by using a `option` type.

Recall:

```
'a option = None | Some of 'a
```

Exercise #8:

Recall the type declaration for binary trees from the previous slide:

```
type 'a btree = Empty
              | Node of ('a * 'a btree * 'a btree)
```

- Draw pictures of two *different* integer binary trees
- For each of the trees you have drawn, write the OCaml expressions that would represent them

Exercise #9:

Recall the definition of the `btree` type

```
type 'a btree = Empty
              | Node of ('a * 'a btree * 'a btree)
```

Define a function `sumTree` that adds up the numbers in an integer binary tree represented using these constructors