CPCS449 Tutorial

Si Zhang
si.zhang2@ucalgary.ca

University of Calgary

2021 Winter

Module Import and Export

```
import MyModule
import MyModule hiding (fun1, fun2)
import qualified MyModule as My

module MyModule (fun1, fun2) where
```

hiding Prelude, do it yourself

```
module MyModule where
import Prelude hiding (not, fst, snd, id, const)

not :: Bool -> Bool
fst :: (a, b) -> a
snd :: (a, b) -> b
id :: a -> a
const :: a -> b -> a
```

Recursion

The basic mechanism for looping in Haskell.

```
fac :: Int -> Int
fac 1 = 1
fac n = n * fac (n-1)

fac1 n = if n == 1 then 1 else  n * fac1 (n-1)

fac2 n = case n of
    1 -> 1
    _ -> n * fac2 (n-1)

fac3 n = product [1..n]
```

Local definitions

- let ... in ...
- ... where ...

```
fac4 n = loop n 1
  where
    loop n x | n > 1  = loop (n-1) (x * n)
             | otherwise  = x

fac5 n =
  let fac' 1 = 1; fac' n = n * fac' (n-1)
  in fac' n
```

Recursive function, do it yourself

```
sum', product' :: [Int] -> Int
length' :: [a] -> Int

append' :: [a] -> [a] -> [a]
reverse' :: [a] -> [a]
filter' :: (a -> Bool) -> [a] -> [a]

take', drop' :: Int -> [a] -> [a]
odds, evens :: [a] -> [a]
```

Tuple, List, String

- Tuple: a finite sequence of elements with different types.
- List: a sequence of elements with the same type.
- String: a list of characters.

```
(1, 'a', True) :: (Int, Char, Bool)
[['H', 'e', 'l', 'l', 'o'], "World"] :: [[Char]]
"Haskell" :: [Char]

fst :: (a, b) -> a
snd :: (a, b) -> b

head :: [a] -> a
tail :: [a] -> [a]

(:) :: a -> [a] -> [a]
(++) :: [a] -> [a] -> [a]
```

Enumerate list

- $[n..m]$ means $[n, n+1, ..., m]$ (if $n > m$ then empty list return)
- $[n, p..m]$ means the elements ascending in steps $p - n$.

```
[1.1 .. 5] = [1.1,2.1,3.1,4.1,5.1]
[10,9 .. 5] = [10,9,8,7,6,5]
['a', 'd' .. 'm'] = "adgjm"
```

List comprehension

It is just a syntactic sugar for some list operations.

- [result | enumerate list, conditions]

```
let list = [2,4,7] in
[ m+n | n<-list, even n, m<-list, odd m]
= [9,11]

[(x, y) | x <- [1..3], y <- ['a'..'c'], x > 2]
= [(3,'a'),(3,'b'),(3,'c')]
```

Participation question 02

What are the types of the following values:

- ['a', 'b', 'c']
- ('a', 'b', 'c')
- [(False, 'O'), (True, '1')]
- ([False, True], ['0', '1'])
- [tail, init, reverse]