訂正

プリント第 ~~3~~ 章, p.10 の下〜p.11 の上あたりを以下に置き換えてください。

なお、変数の型をプログラム中に明示したい場合 “::” という記号を使って、

変数名 :: 型

と書く。例えば trivial や fact の型を明示しておきたい場合は、

```
1  trivial :: a -> a
2  trivial x = x
3
4  fact :: Integer -> Integer
5  fact n = if n == 0 then 1 else n * fact (n - 1)
```

のように書く。ただし通常は、変数や関数の型はプログラマが明示しなくても、Haskell 処理系が推論してくれる。この仕組みを 型推論 という。

2.5 パターンマッチング

Haskell では、関数定義の仮引数の部分に パターン というものを書いて、引数の形に応じて場合分けを行なう事が可能である [1]。パターンとは、大雑把に言って変数と定数、構成子（:や [] など）からのみ生成される式である。

```
1  fact :: Integer -> Integer
2  fact 0 = 1
3  fact n = n * fact (n - 1)
4
5  -- Prelude の length とほぼ同じ
6  myLength :: [a] -> Integer
7  myLength []     = 0
8  myLength (x:xs) = 1 + myLength xs
```

関数名 パターン$_{11}$ ... パターン$_{1m}$ = 式$_1$
関数名 パターン$_{21}$ ... パターン$_{2m}$ = 式$_2$
...
関数名 パターン$_{n1}$ ... パターン$_{nm}$ = 式$_n$

以降、すべての節番号が+1 になります。

[1] 一般に関数型言語はデータの種類は増えずに処理（関数）が増えるような場合が得意、オブジェクト指向言語は、データ（クラス）の種類が増えて、処理（メソッド）が増えないような場合が得意である。