問題 4

リスト as を多項式とみなして、次のような計算をする関数を定義しなさい。

1. 二つの多項式 as と bs の和を求める関数 polyplus
2. 二つの多項式 as と bs の積を求める関数 polytimes

解答及び解説

テキストの No.3 にあるように、リスト as を

$$as \quad = \quad [a_0, a_1, a_2, \ldots, a_n]$$

としたとき、これは

$$f(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n$$

を表していると考える。

1. 和を求める関数 polyplus

さて、二つの多項式の和を求めるには、同次の係数同士を足し算すれば良いので、

```
polyplus :: (Eq a, Num a) => Poly -> Poly -> Poly
polyplus [] ys = ys
polyplus xs [] = xs
polyplus (x:xs) (y:ys) = (x+y):polyplus xs ys
```

と定義できる。

ところが、係数の和が 0 になる場合があり、最高次数の係数が 0 になると、多項式の次数が下がる。すなわち、リストの最後の要素が 0 になってしまう。

そのままでも、多項式の計算結果などには影響しないが、通常は係数が 0 となった項は書かないので、それをリストで表現する場合も、最後の 0 は無い方が良い。そこで、最高次の係数が 0 になったらリストを短くすることを考える。このことは、足し算する多項式が同次数の場合にのみ起こるので、次のようにパターンを使って、最高次のところで和が 0 になるかどうかを調べるようにする。

```
polyplus :: (Eq a, Num a) => Poly -> Poly -> Poly
polyplus [] ys = ys
polyplus xs [] = xs
polyplus [x] [y] = if (x+y) == 0 then [] else [x+y]
polyplus (x:xs) (y:ys) = (x+y):polyplus xs ys
```

ところが、実際には最高次の項だけでなく、その一つ下の次数の項でも係数が 0 になり、より次数が低い多項式になることもある。この場合も、やはりリストの終わりが 0 になってしまわないような工夫が必要である。

```
polyplus :: (Eq a, Num a) => Poly -> Poly -> Poly
polyplus [] ys = ys
polyplus xs [] = xs
polyplus (x:xs) (y:ys) | null zs   = if z == 0 then [] else [z]
                       | otherwise = z:zs
         where zs = (polyplus xs ys)
               z = x + y
```

2. 積を求める関数 polytimes

ところで、二つの多項式の積を求める場合は、和の時のように次数が変わることはない。従って、次のように定義できる。

```
polytimes :: (Eq a, Num a) => Poly a -> Poly a -> Poly a
polytimes [] _ = []
polytimes _ [] = []
polytimes (x:xs) ys = polyplus (map (x*) ys) (0:(polytimes xs ys))
```

なお、多項式 f(x) を表すリストを as とし、f(x) に x を掛けた多項式 x*f(x) を表すリストを bs とすると、

$$f(x) \quad = \quad a_0 + a_1 x + a_2 x^2 + ...$$

であるから、

$$\begin{aligned} x * f(x) &= b_0 + b_1 x + b_2 x^2 + b_3 x^3 + ... \\ &= 0 + a_0 x + a_1 x^2 + a_2 x^3 + ... \end{aligned}$$

となることに注意すると、多項式 bs は、as を用いて

```
bs = 0 : as
```

と表すことができる。