

## 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

$$as = [a_0, a_1, a_2, \dots, a_n]$$

としたとき、これは

$$f(x) = a_0 + a_1x + a_2x^2 + \dots + a_nx^n$$

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) => [a] -> [a] -> [a]
polyplus [] bs = bs
polyplus as [] = as
polyplus (a:as) (b:bs) = (a+b):polyplus as bs
```

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式の場合にのみ起こるので、次のように要素が一つであるリストのパターンを使って、最高次数が同じである場合を表し、その時の和が 0 になるかどうかを調べるようにする。

```
polyplus :: (Eq a, Num a) => [a] -> [a] -> [a]
polyplus [] bs = bs
polyplus as [] = as
polyplus [a] [b] = if (a+b) == 0 then [] else [a+b]
polyplus (a:as) (b:bs) = (a+b):polyplus as bs
```

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) => [a] -> [a] -> [a]
polyplus [] bs = bs
polyplus as [] = as
polyplus (a:as) (b:bs) | null cs = if c == 0 then [] else [c]
                        | otherwise = c:cs
  where cs = (polyplus as bs)
        c = a + b
```

## 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) => [a] -> [a] -> [a]
polytimes [] _ = []
polytimes _ [] = []
polytimes (a:as) bs = polyplus (map (a*) bs) (0:(polytimes as bs))
```

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、

$$f(x) = a_0 + a_1x + a_2x^2 + \dots$$

であるから、

$$\begin{aligned} x * f(x) &= a_0x + a_1x^2 + a_2x^3 + \dots \\ &= b_0 + b_1x + b_2x^2 + b_3x^3 + \dots \end{aligned}$$

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。