演習 3.1

演習 3.1.1 以下を計算する関数を書け.

a) 実数 x の立方.

b) 3 つの整数からなる組 (int * int * int) のうちの最小の数.

c) リストの第 3 要素. この関数はリストの長さが 2 以下の場合正常に動作しなくてよい.

d) 長さ 3 の組 (tuple) の順序を逆にする.

e) 文字列の 3 番目の文字. 文字列の長さが 3 未満の時は, 正常に動作しなくてよい. ヒント: 関数 explode と演習 3.1.1 c) の関数を使うこと.

f) リストを一回だけ巡回せよ. すなわち, リスト $[a_1, a_2, \ldots, a_n]$ があったら, $[a_2, \ldots, a_n, a_1]$ を生ずる.

演習 3.1.2 下記の動作を行う関数を書け.

a) 3 つの整数を引数として取り, そのうち最小のものと最大のものの対 (pair) を生じる関数.

b) 3 つの整数を引数として取り, それを小さい順にソートしたリストを生じる関数.

c) 実数を 10 の位でまるめる関数.

d) リストを引数として取り, そのうちの 2 番目の要素を削除したリストを生じる関数. この関数はリストの長さが 2 未満であったとき正常に動作しなくてよい.

演習 3.1.3 次のように関数を定義したとする.

```
val a = 2;
fun f(b) = a*b;
val b = 3;
fun g(a) = a+b;
```

次の式の値を答えよ.

a) f(4).

b) f(4) + b.

c) g(5).

d) g(5) + a.

e) f(g(6)).

f) g(f(7)).

Jeffrey D. Ullman 著
Elements of ML Programming ML97edition
1997 より

演習 3.2

演習 3.2.1 以下の再帰関数を書け.

a) $n \geq 1$ である整数 n を取り，その階乗を計算する関数．ただしこの関数は n が 1 より小さいときは正常に動作しなくてよい.

b) 整数 i とリスト L を取り，L を i 回巡回する関数．つまりもし

$$L = [a_1, a_2, \ldots, a_n]$$

のとき，$[a_{i+1}, a_{i+2}, \ldots, a_n, a_1, a_2, \ldots, a_i]$ を生じる．i が 1 より小さいときは正しく動作しなくてよい．演習 3.1.1 f) で作成した関数を使ってよい.

c) リストの各要素を複製せよ．つまり，リスト $[a_1, a_2, \ldots, a_n]$ が与えられたら，$[a_1, a_1, a_2, a_2, \ldots, a_n, a_n]$ を生ずる.

d) リストの長さを数える関数．(ML にはこれを行う length という関数があるが，それを使わないで解答せよ．)

e) 実数 x の i 乗を計算する関数．ここで i は 0 以上の整数．この関数は x と i の二つの引数を取り，i が負の数の時は正常に動作しなくてよい.

f) 整数のリストの最大値を計算する関数．この関数はリストが空の時正常に動作しなくてよい.

演習 3.2.2 以下の関数を定義する.

```
fun foo(a,b,c,d) =
    if a=b then c+1 else
        if a>b then c else b+d
```

この関数で，a, b, c, d はすべて整数型 (integer) であると推論することができる．(ML が) どのようにして推論可能であるかを説明せよ.

演習 3.2.3 次のように始まる関数を定義したと仮定する.

```
fun f(a:int, b, c, d, e) = ...
```

この関数の本体が以下のそれぞれであったときに，b,c,d,e の型についてどんなことが推論できるかを述べよ.

a) `if a<b+c then d else e.`

b) `if a<b then c else d.`

c) `if a<b then b+c else d+e.`

d) `if a<b then b<c else d.`

e) `if b<c then a else c+d.`

f) `if b<c then d else e.`

g) `if b<c then d+e else d*e.`

演習 3.3

演習 3.3.1 以前の演習問題で書いた以下の関数をパターンを使って書け.

a) 階乗を計算する関数 (演習 3.2.1 (a))

b) リストを 1 回だけ巡回する関数 (演習 3.1.1 (f))

c) リストを i 回巡回する関数 (演習 3.2.1 (b))

d) リストの要素を重複させる関数 (演習 3.2.1 (c))

e) x の i 乗を計算する関数 (演習 3.2.1 (e))

f) 実数のリストの要素のうちの最大数を計算する関数

演習 3.3.2 リストの奇数番目と偶数番目の要素を入れ換える関数を書け. すなわち, リスト $[a_1, a_2, \ldots, a_n]$ が与えられると $[a_2, a_1, a_4, a_3, a_6, a_5, \ldots]$ を生じる. もし n が奇数ならば a_n がそのまま最後の要素となる.

演習 3.3.3 リスト L と整数 i が与えられたとき, リスト L の i 番目の要素を削除したリストを生じる関数を書け. ただしリストの先頭を 1 番目と考える.

演習 3.3.4 リストのリストの要素の和を求める関数 sumLists が以下のように定義されている.

```
fun sumLists(nil) = 0
|   sumLists(nil::YS) = sumLists(YS)
|   sumLists((x::xs)::YS) = x + sumLists(xs::YS);
```

val sumLists = fn : int list list → int

この関数を

```
sumLists([[1,2], nil, [3]]);
```

と呼んだとき, どのように実行されるか (どのように関数が再帰的に呼ばれるか, どの変数にどの値が入るか) を答えよ.

演習 3.3.5 (x::y::zs, w) というパターンは以下の式と一致するだろうか? 一致する場合, それぞれの変数 x, y, zs, w にはどれが対応するかを答えよ.

a) (["a", "b", "c"], ["d", "e"])

b) (["a", "b"], 4.5)

c) ([5], [6,7])

演習 3.3.6 (x,y)::zs というパターンがどのように [((1,2),3)] とマッチするのかを答えよ. つまり, 変数 x, y, zs はどれに対応するか.

演習 3.3.7 負でない数の二乗を計算する再帰的な定義は以下のようになる.

基底 $0^2 = 0$

帰納段階 $n^2 = (n-1)^2 + 2n - 1 \quad (n > 0)$

この式を用いて負でない数の二乗を再帰的に計算する関数を書け．入力が負の整数であった時に正しく動作しなくてよい．

演習 3.3.8 整数の対 (pair) のリストを取り，その対で小さい方の数を 1 番目の要素とする関数を書け． 要素を並べ替える必要がない時は全体を as を用いて参照せよ．

演習 3.3.9 文字のリストを取り，最初の要素が母音であったら true を，そうでなければ false を返す関数を書け．パターンの中で使える時はワイルドカード _ を使うこと．

演習 3.3.10 「ピッグ・ラテン (Pig Latin)」へ翻訳する簡単な規則は，母音で始まる単語には”yay” を付加し，1 つ以上の子音で始まる単語にはその子音を後ろに回してから “ay” を追加する．例えば，“able” は “ableyay” になり，“stripe” は “ipestray” となる．文字列をピッグ・ラテンに変換する関数を書け．ヒント: explode と母音かどうかを判定する演習 3.3.9 で作成した関数を使用せよ．

演習 3.3.11 リストで集合を表す．集合の要素はどんな順番でリスト中に表れてもよいが，同じ要素はたかだか一度しかリストには表れないとする．集合における以下の操作を実現する関数を書け．

a) x が S に含まれるとき true, 含まれないとき false を返す関数 member (x, S). x は S のどこに現れてもよい．

b) x を S から削除する関数 delete (x,S). x はリスト S 中にたかだか一度しか現れないと仮定してよい．

c) x を S に追加する関数 insert (x, S). x は S 中にたかだか一度しか現れないという条件を満たすために，x がすでに S 中に存在するかどうかを確認しなければならない．ただ単純に S の頭部 (head) に x を追加するだけでは正しくない．

演習 3.3.12 任意の要素 a と，a と同じ型の要素からなるリストのリスト L を取り，L 中の各リストの最初に a を挿入する関数を書け．たとえば，$a = 1$ で L が $[[2,3],[4,5,6],nil]$ であれば，結果は $[[1,2,3],[1,4,5,6],[1]]$ となる．

演習 3.3.13 集合を演習 3.3.11 のようにリストで表すとする．集合 S のべき集合 (power set) とは，S のすべての部分集合の集合である．集合の集合は ML ではリストを要素とするリストで表現できる．例えば，もし S が 集合 $\{1,2\}$ であったとすれば，S のべき集合は $\{\emptyset,\{1\},\{2\},\{1,2\}\}$ となる．ここで $\emptyset$ は空集合を示す．このべき集合は ML でリストのリストで $[nil,[1],[2],[1,2]]$ と表される．つまりこのリストの要素はリストでそのリストのひとつひとつが S の部分集合を示す．集合を表現するリストを引数として取り，その集合のべき集合を計算する関数を書け．ヒント:リストの尾部 (tail) のべき集合を再帰的に構成せよ．リスト全体のべき集合を構成するには演習 3.3.12 で作成した関数を利用せよ．

演習 3.3.14 実数のリスト $[a_1,a_2,\ldots,a_n]$ を取り，$\prod_{i<j}(a_i - a_j)$ を計算する関数を書け．つまり，要素間のすべての差分，すなわちリストの要素からそれより後に現れるすべての要素を引いたものの積を計算する．もし対がなければ「積」は 1.0 となる．ヒント: 与えられた a と $[b_1,\ldots,b_n]$ から $\prod_{i=1}^{n}(a - b_i)$ を計算する補助関数の作成から始めよ．

演習 3.3.15 リストが空であるかどうかを判定する関数を書け．つまり，リストが空であるときまたその時だけ true を返す．ML には，これを判定するビルトイン関数 null があるが，この演習では関数 null は使わないこと．

演習 3.3.16 以下の関数 sumPairs : (int * int) list → int が何をする関数であるかを説明し，どのようにしてこの関数の定義域が (int * int) list と推論されるかを説明せよ．

```
fun sumPairs(nil) = 0
 |  sumPairs((x,y)::zs) = x + y + sumPairs(zs);
```

演習 3.4

演習 3.4.1 x^{1000} を計算する簡単な関数を書け.

演習 3.4.2 以下のプログラムの (4) 行目の val 文をパターンを使わないように書き直せ. つまり (4) 行目を val x = split(cs) にして, 必要な時に対 x の要素にどのようにアクセスするのかを書け.

```
(1) fun split(nil) = (nil, nil)
(2) |   split([a]) = ([a], nil)
(3) |   split(a::b::cs) =
            let
(4)             val (M, N) = split(cs)
            in
(5)             (a::M, b::N)
            end;
```

演習 3.4.3 演習 3.3.13 で作成したべき集合を計算する関数を let を使い尾部 (tail) のべき集合の計算を一度にすますように修正せよ.

演習 3.4.4 演習 3.2.1(f) の実数のリストの最大値を計算する関数を let を使って書け. ヒント:最初にリストの尾部 (tail) の最大値を計算する.

演習 3.4.5 実数 x, 負でない整数 i に対して x^{2^i} を計算する関数を書け. 関数の再帰呼び出しは 1 回だけにせよ. ヒント:x に対し, 2 乗計算を i 回繰り返す. 例えば, $i = 3$ ならば $((x^2)^2)^2$ を計算せよ.

演習 3.4.6 以下の関数 sumPairs を第一要素の和と第二要素の和を別々に計算し, 最後にその二つを足すように書き換えよ.

```
fun sumPairs(nil) = 0
 |  sumPairs((x,y)::zs) = x + y + sumPairs(zs);
```

演習 3.4.7 整数のリストを取り, その偶数番目の数の和と奇数番目の数の和の対 (pair) を返す関数を書け. ただし, 補助関数を使ってはいけない.

Jeffrey D. Ullman 著
Elements of ML Programming ML97edition
1997 より

演習 3.5

演習 3.5.1 リスト L と M を連結する関数 cat(L,M) を書け．ただし，@ 演算子を使ってはいけない．:: を使うこと．この関数はリスト L の長さに比例した時間で動作し，リスト M の長さには依存しないこと．

演習 3.5.2 演習 3.2.1(b) と同じく，リスト L を i 回巡回する関数 cycle(L,i) を書け．ただし，この関数はリスト L の長さ (少なくとも i はあると仮定する) に比例した時間で動くものとする．ヒント:補助関数に分割して作成せよ．

Jeffrey D. Ullman 著
Elements of ML Programming ML97edition
1997 より

演習 3.6

演習 3.6.1 省略.

演習 3.6.2 省略.

演習 3.6.3 与えられた実数 a に対し，多項式を評価する関数を書け．すなわち，リスト P と実数 a を引数にとり $P(a)$ を計算する関数 eval(P,a) を書け.

演習 3.6.4 実数のリスト $[a_1, a_2, \ldots, a_n]$ を取り，解が $a_1, a_2, \ldots, a_n$ となる多項式を見つける関数を書け．ヒント: この多項式は $(x - a_i)\ \ i = 1, 2, \ldots, n$ の積で表される.

演習 3.6.5 二つの変数 x, y を含む多項式をリストで表すことを考える．表し方は，変数 s の係数を変数 y を含む多項式で表すことにする．つまり，このような多項式はリストのリストとして表すことができる．例えば，多項式 $1 + 2xy + 3xy^2 + 4x^3y$ は，$1 + (2y + 3y^2)x + (4y)x^3)$ である．多項式 $2y + 3y^2$ は リスト [0.0, 2.0, 3.0] で表され，多項式 $4y$ は [0.0, 4.0] で表される．したがってこの多項式は

[[1.0], [0.0, 2.0, 3.0], [], [0.0, 4.0]]

と表せる．この 2 変数の多項式の和，スカラー倍，積を計算する関数を書け.

Jeffrey D. Ullman 著
Elements of ML Programming ML97edition
1997 より

Jeffrey D. Ullman 著
Elements of ML Programming ML97edition
1997 より