演習 5.1

演習 5.1.1 省略.
演習 5.1.2 次の関数を fn を使って書き換えよ.

- 関数 padd
- 関数 smult
- 関数 pmult
- 関数 sumPairs
- 関数 merge
- 関数 comb

```
fun padd(P,nil) = P
     |   padd(nil,Q) = Q
     |   padd((p:real)::ps, q::qs) = (p+q)::padd(ps,qs);

fun smult(nil,q) = nil
     |   smult((p:real)::ps,q) = (p*q)::smult(ps,q);

fun pmult(P,nil) = nil
     |   pmult(P,q::qs) = padd(smult(P,q), 0.0::pmult(P,qs));

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

fun merge(nil,M) = M
     |   merge(L,nil) = L
     |   merge(L as x::xs, M as y::ys) =
             if x<y then x::merge(xs,M)
             else y::merge(L,ys);

fun comb(_,0) = 1
|   comb(n,m) =
       if m=n then 1
       else comb(n-1,m) + comb(n-1, m-1);
```

演習 5.1.3 西暦が 4 で割り切れる年はうるう年である．ただし，100 で割り切れる年はうるう年でない．100 で割り切れても 400 で割り切れる年はうるう年である．あたえられた年がうるう年であるかどうかを判定する case 文を書け.
演習 5.1.4 orelse, andalso は if-then-else 文で書き換えることができる．また，if-then-else 文は case 文で書き換えることができる．次の orelse, andalso 文を case 文で書き直せ.

a) E orelse F

b) E andalso F

演習 5.2

演習 5.2.1 リストの第 3 要素を返す関数を書け．この関数が定義されないような入力に対しては適切な例外を発生させること．

演習 5.2.2 階乗を計算する関数を書け．ただし，入力が 0 の時は 1, 入力が負の数の時は適切な例外を発生させること．

演習 5.2.3 省略.

演習 5.2.4 実数の行列を，実数のリストのリストで表すことができる.「主」リスト中の各リストは行列の行を表す．行列の行列式は，再帰的に第一行と第一列を消去していく方法 (pivot condensation) によって計算することができる．この方法は次のように記述することができる．

基底 もし 1 行 1 列の行列だったら，その 1 要素をリターンする．

帰納段階 もし 1 行 1 列よりも大きければ,

i) 第 1 行の各要素を第 1 要素，たとえば a で除算することにより正規化する．

ii) 第 1 行第 1 列以外の各要素 M_{ij} につき，i 番目の行の第 1 要素と第 1 行目の j 番目の要素を乗じたものを M_{ij} から減じる．

iii) 第 1 行第 1 列を消去していくことで再帰的に行列の行列式を計算する．結果は，この行列式に a を乗じたものである．a は段階 i) の定数であり，もとの行列の左上隅にあったものであることを思い出されたい．

このアルゴリズムを実現する関数群を書け．エラーを捕獲する適切な例外も定義せよ．エラーには次のものを含めよ．

1. 段階 i) で $a = 0$ である．つまり段階 ii) で a による除算ができない場合．
2. もとの行列がそもそも正方行列でない場合．つまり行と列の数が異なっていたり行の長さが均一でない等．

ヒント: この手続きをよりやさしい段階にわけることができる．与えられた定数によって各要素を除算することにより，行 (リスト) を正規化する関数から出発する．さらに定数を掛けた行を他の行から減算する関数を書く．その上で行のリストを取り，各行の尾部から行の頭部と与えられたリストの積を引く関数を書く．後者はこのアルゴリズムの中心である．与えられた行は第 1 行の正規化された尾部である．それに (ベクトルと考えて) 行の頭部を掛け，その上で同じ今日の尾部から (再びベクトルと考えて) その結果を引くと，このアルゴリズムの基本操作を行ったことになる．

演習 5.2.5 次の関数を考える．

```
fun myFavoriteException("sally") = Div
|   myFavoriteException("joe") = Match;
```

a) この関数の型は何?

b) myFavoriteException(“joe”) としたときと myFavoriteException(“zzz”) としたときは，同じ答え Match を生ずる．この二つの呼出しの違いは何か?

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

演習 5.3

演習 5.3.1

関数 rev1 と関数 rev2 が以下のようなものであるとする．

```
fun rev1(L) =
    if L = nil then nil
    else rev1(tl(L)) @ [hd(L)];

fun rev2(nil) = nil
|   rev2(x::xs) = rev2(xs) @ [x];
```

このとき，次の結果はどうなるか．

a) rev1([(rev1: int list → int list), rev1])

b) rev2([(rev1: int list → int list), rev1])

c) rev1([rev1, rev1])

d) rev2([rev2, rev2])

e) rev1([chr, chr])

f) rev2([chr, chr])

g) rev1([chr, ord])

h) rev2([chr, ord])

演習 5.3.2

多相型には次のような制限がある．i) equality type による制限 ii) 型変数を決まった型で置き換えることによる制限 iii) 型変数を決まった型でないもので置き換えることによる制限．次の型表現にそれぞれの制限を加えたものの例を示せ．

a) 'a * 'b * int

b) ('a list) * ('b list)

演習 5.3.3

関数 f(x,y,z) を考える． f の引数が以下の型となるような例を考えよ．(関数の式を考えるということ)

a) 'a * ''b * ('a → ''b)

b) 'a * 'a * int

c) 'a list * 'b * 'a

d) ('a * 'b) * 'a list * 'b list

演習 5.3.4

以下の型が equality 型 (同値型) であるかどうかを答えよ.

a) int * string list

b) (int → char) * string

c) int → string → unit

d) real * (string * string) list

演習 5.3.5

L の値は [(1,2), (3,4)], M の値は (1,2), N の値は (3,4) である．以下のどの式が true になるか?

a) L = M::[N]

b) M::L = L@[N]

c) [(1,2)]@[N] = L@nil

d) N::L = (3,4)::M::N::nil

演習 5.3.6

もし ML が実行時 (runtime) にリストの中身をみないでコンス演算子を実行するなら，どうやってそのリストの頭部 (head) と尾部 (tail) の要素の型が同じだということがわかるのか?

演習 5.3.7

次の三つの関数がある.

```
fun f(nil) = nil
|   f([x]) = [x]
|   f(x::y::zs) = [x,y];

fun g(x,y) = (f(x), f(y));

fun h(x,y) =
    let
       val v = f(nil)
    in
       (x::v, y::v)
    end;
```

関数 f は長さが 3 以上あるリストの場合それを長さ 2 にする関数である．長さが 3 に満たないときはそのリストそのままを返す．関数 g は二つのリストを引数として取り，それぞれに関数 f を適用してその結果を対として返す．関数 h は，関数 f に nil を適用した結果，すなわち空リストを引数として受け取った二つの値の後ろにくっつけ，できあがった二つのリストを対として返す．以下の式のうちで正しいのはどれか．正しい場合にその答えは何となるかまたエラーとなるものは何が原因か.

a) g([1,2,3], [”a”])

b) g([1,2,3], nil)

c) g([f,f],[1])

d) g([1], [1.0])

e) h(1,2)

f) h(1, “a”)

g) h(nil, nil)

h) h([1], nil)

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

演習 5.4

演習 5.4.1

初期値 a, 増分 δ, 点の数 n, 実数を引数にとり結果として実数を返す関数 F の四つを引数とする関数 tabulate を書け．この関数は各行に x と $F(x)$ の値を記した表を出力する．ここで x は初期値 a で増分 δ で増えていく．すなわち，

$$x = a, a+\delta, a+2\delta, \ldots, a+(n-1)\delta$$

に対応する $F(x)$ を計算して出力する．

演習 5.4.2

シンプソン法とは，関数を数値的に積分するより正確な手法である．次のような等間隔の $2n+1$ 個の点において関数 F を評価するとしよう．

$$a, a+\delta, a+2\delta, \ldots, a+2n\delta$$

そのとき，積分

$$\int_a^{a+2n\delta} F(x)dx$$

は次のように近似できる．

$$\begin{aligned}&\delta(F(a)+4F(a+\delta)+2F(a+2\delta)+4F(a+3\delta)+2F(a+4\delta)+\ldots\\&+2F(a+(2n-2)\delta)+4F(a+(2n-1)\delta)+F(a+2n\delta))/3\end{aligned}$$

すなわち，偶数番目の項は係数 4 を，奇数番目の項は係数 2 を持つ．ただし最初と最後の項は例外でそれぞれ 1 である．開始点 a と終了点 b, 整数 n そしてシンプソン法で積分される関数 F を取る関数 simpson を書け．作成した関数を多項式 x^2, x^3 等で試験せよ．x^i の正確な積分を計算するのに，$a = 0.0, b = 1.0, n = 1$ でも失敗する最小の整数 i を求めよ．

演習 5.4.3 台形公式やシンプソンの公式において，δ を再帰のたびに計算するのではなく (もっとも本文で示したように毎回計算する方が誤差が少なくなる場合もあるが), 1 度だけ計算するようにできる．1 度だけ計算するようにして以下のものを書き換えよ．

a) 以下の関数 trap

b) 演習 5.4.2 の問題.

```
fun trap(a,b,n,F) =
            if n<=0 orelse b-a<=0.0 then 0.0
            else
                let
                    val delta = (b-a)/real(n)
                in
                    delta*(F(a)+F(a+delta))/2.0 +
                        trap(a+delta,b,n-1,F)
                end;
```

演習 5.4.4 関数 trap (演習 5.4.3 で示した) を入力が不適切な場合に例外処理を行うように修正せよ．具体的には一行目の条件に対応する例外処理を行うようにせよ．

演習 5.4.5 関数 simpleMap(F,L) を用いて以下の操作を行う式を書け.

```
fun simpleMap(F, nil) = nil
|   simpleMap(F, x::xs) = F(x)::simpleMap(F, xs);
```

a) 実数のリスト中の負の数を 0 で置き換え，非負の数はそのままにせよ．

b) 整数のリストの全ての要素に 1 を加えよ．

c) 文字のリストのうち，小文字を対応する大文字に変換せよ．文字のリストは小文字だけからなるわけではないことに注意せよ．

d) 文字列のリストの各要素の 5 文字より長い部分を切捨てよ．すなわち，文字列の長さが 5 文字より長いものは 6 文字目以降を切捨て，5 文字以下のものはそのままにする．

演習 5.4.6 関数 reduce を使って以下の操作を行え．

```
exception EmptyList;

fun reduce(F, nil) = raise EmptyList
|   reduce(F, [a]) = a
|   reduce(F, x::xs) = F(x, reduce(F,xs));
```

a) 実数のリストの最大値を求めよ．

b) 実数のリストの最小値を求めよ．

c) 文字列のリストのすべての要素を連結してひとつの文字列にせよ．

d) ブール型のリストの論理和を求めよ．

演習 5.4.7 関数 filter を使って以下の操作を行え．

```
fun filter(P, nil) = nil
|   filter(P, x::xs) =
      if P(x) then x::filter(P,xs)
      else filter(P,xs);
```

a) 実数のリストから 0 より大きい数だけを集めたリストを作れ．

b) 実数のリストから，値が 1 と 2 の間のものだけを集めたリストを作れ．

c) 文字列のリストから，#”a” で始まる文字列だけを集めたリストを作れ．

d) 文字列のリストから，少なくとも 3 以上の長さの文字列だけを集めたリストを作れ．

g **演習 5.4.8** リスト L に対して reduce(op -, L) を行うとどうなるか? 何が計算されるか?

演習 5.4.9 関数 F とリスト $[a_1, a_2, \ldots, a_n]$ の二つを引数に取り，

$$F(\ldots F(F(a_1, a_2), a_3) \ldots, a_n)$$

を計算する関数 lreduce を書け．この関数は reduce と同じように働くが reduce がリストの後ろの方から計算していくのに対し，リストの前から計算していく．

演習 5.4.10 リスト L に対して lreduce(op -, L) を行うとどうなるか? 何が計算されるか?
演習 5.4.11 reduce の別バージョンは，基底として型 'b の定数 g と型 'a * 'b →'b の関数 F そして型 'a の要素のリストを取る．リスト $[a_1, a_2, \ldots, a_n]$ に適用すると，結果は以下のようになる．

$$F(a_1 \ldots F(a_{n-1}, F(a_n, g)) \ldots)$$

この操作を実行する関数 reduceB を書け.
演習 5.4.12 演習 5.4.11 で書いた関数 reduceB を使って以下の操作を行え．

a) リストの長さを計算せよ．

b) リストの接尾辞のリストを計算せよ．例えば，リスト [1,2,3] に対しては，[[1,2,3],[2,3],[3],nil] を生じる．

演習 5.4.13 多相型の別の使用法として，+や*のように多重定義される記号の遅延束縛 (late binding) がある．つまりこれらの記号を関数 f の中で使用するかわりに plus や times という名前をつけておいて，これらの名前を関数 f のパラメータとするのである．そうするとパラメータの適当な定義を用いて f を呼び出すことができるので，名前を正しい意味に束縛するのをできる限り遅くすることができる．

a) 加算と乗算を表す関数をパラメータとして取り，また多項式 (を表すリスト) とその多項式を評価するための値を取るような関数 eval を作成せよ．

b) 整数多項式 $4x^3 + 3x^2 + 2x + 1$ を $x = 5$ で評価するために，a) で作った関数をどのように呼び出すかを示せ．

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

演習 5.5

演習 5.5.1 関数のリストと一つの値を引数として取り，それらの関数に値を適用したときの結果をリストとして出力する関数 applyList をカリー化形式で書け.

演習 5.5.2 定義域の型が D で値域の型が R である関数 F を考える．ここで型 R は関数で $T_1 \to T_2$ で表せるとする．すなわち，関数 F の型は D $\to (T_1 \to T_2)$ である．この関数 F と型 D のリストを引数として取り，関数 G を結果とする関数 makeFnList をカリー化形式で書け．ただし，関数 G は型 D のリスト $[d_1, d_2, \ldots, d_n]$ を取り，型 R のリスト $[f_1, f_2, \ldots, f_n]$ を結果とし，$f_i = F(d_i)$ である.

演習 5.5.3 二つの文字列を引数として取り，最初の文字列が 2 番目の文字列の部分文字列であるかどうかを判定する関数 substring を書け．カリー化形式であってもなくても構わない．ここで文字列 x が文字列 y の「部分文字列」であるとは，文字列 y が，ある文字列 w, z を使って w x z の連結で表すことができるときに言う．ここで w, x, z のどれも空の文字列であってよい．例えば，文字列 “abc” の部分文字列には，“”, “b”, “ab” が含まれる．演習 5.5.2 で作成した関数 makeFnList とここで作成した substring を用いて文字列のリスト $[s_1, \ldots, s_n]$ を引数として取り，s_i が文字列 x の部分配列であるかどうかを判定する関数 $F_i(x)$ のリスト $[F_1, \ldots, F_n]$ を生じる関数 f を作成せよ.

演習 5.5.4 演習 5.5.3 で作った関数 f を使い，“he”, “she”, “her”, “his” がある文字列の部分文字列であるかどうかを判定するための関数のリストを作成せよ.

演習 5.5.5 演習 5.5.4 で作ったリストと演習 5.5.1 で作った関数 applyList を用いて “he”, “she”, “her”, “his” が “hershey” の部分文字列かどうかを判定せよ.

演習 5.5.6 演習 5.5.3 を部分文字列 (substring) ではなく部分並び (subsequences) について判定する関数を作成せよ．文字列 x が文字列 y の部分並びであるとは，文字列 y からゼロ個以上の文字を取り除いて文字列 x ができるということである．例えば，文字列 “ac” は文字列 “abc” の部分並びであるが，部分文字列ではない．この関数を作成した後，演習 5.5.4 のように，文字列のリスト [”ear”, “part”, “trap”, “seat”] がある文字列の部分並びであるかどうかを判定するための関数リストを作り，最後にそれを文字列 “separate” に対して適用せよ.

演習 5.5.7 カリー化された関数をカリー化されない形に変換するのは簡単である．次の変換を行う高階関数を書け.

a) n 個の要素の組の型である 1 個のパラメータを取る関数 F が与えられたとき，関数 curry を適用することで n 個 の引数をカリー化された形式で取る関数 G を生じる．$G\ x_1\ x_2\ \ldots\ x_n$ は $F(x_1, x_2, \ldots, x_n)$ と同じ値を生じる.

b) n 個のパラメータを取るカリー化された関数 F が与えられたとき，関数 uncurry を適用することで n 個 の要素の組を 1 個のパラメータとして取る関数 G を生じる．$G(x_1, x_2, \ldots, x_n)$ は $F\ x_1\ x_2\ \ldots\ x_n$ と同じ値を生じる.

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

演習 5.6

演習 5.6.1

以下の map, foldr, foldl を用いてリストに関する操作を行え．リストの要素への操作に関しては関数の定義が必要な場合は匿名関数を使うこと．

```
fun map F =
    let
        fun M nil = nil
        |   M(x::xs) = F x :: M xs
    in
        M
    end;

fun foldr F y nil = y
|   foldr F y (x::xs) = F(x, foldr F y xs);

fun foldl F y nil = y
     |   foldl F y (x::xs) = foldl F (F(x,y)) xs;
```

a) 整数のリストを実数のリストに変換する関数

b) 整数のリスト L を L の各要素の絶対値でかつ実数のリストに変換する関数

c) 関数 implode, すなわち、文字のリストを一つの文字列に変換する関数

d) 関数 concat, すなわち、文字列のリストを一つの文字列に連結する関数

e) 整数の列 $[a_1, a_2, \ldots, a_n]$ を $a_1 - a_2 + a_3 - a_4 + \ldots$ に変換する関数

f) boolean のリストからその論理積 (AND) を計算する関数

g) boolean のリストからその論理和 (OR) を計算する関数

h) boolean のリストからその排他的論理和 (exclusive or) を計算する関数．ただし exclusive or はリスト中，true であるものが奇数であるときに true, そうでないときに false となる．

演習 5.6.2 foldl の定義を foldr を参考にして書け．(でもこれは演習 5.6.1 の問題文に書いてしまった．)

演習 5.6.3 以下のように関数 Comp をカリー化関数として定義する．

```
fun comp F G =
        let
            fun C x = G(F(x))
        in
            C
        end;
```

すると，最初の関数 F と次の関数 G の合成 G o F と同じものを生じる．しかし，

```
val I = comp (fn x => x+3);
```

と書くと，“nongeneralizable type variable” という警告がでる．これは I の定義域および関数の型がわからない，という警告である．

a) 上の式は，適切な型の情報を与えてやれば定義できる． I の値域を string としたとすると，I の型はどうなるか．

b) a) のように I を定義したとき，I に整数 を与えて x + 3 の結果を文字列で得るにはどのようにすればよいか．

c) b) の答えを使って整数 x に対し x + 3 の答えを出力 (print) する関数を書け．

演習 5.6.4 演習 5.6.3 のように定義された関数 comp と以下のような関数 add1 を考える．

```
fun add1 x = x + 1;
```

以下の式で，関数の場合には型，定数の場合にはその値を答えよ．

a) val compA1 = comp add1;

b) val compCompA1 = comp compA1;

c) val f = compA1 add1;

d) f(2);

e) val g = compCompA1 compA1;

f) val h = g add1;

g) h(2);

演習 5.6.5 compA1, compCompA1 は演習 5.6.4 で定義したものとする．次の型を答えよ．

a) val f = compA1 real; だたし real は整数を実数に変換する組込み関数．

b) val compT = comp trunc; ただし trunc は実数を整数にする組込み関数．

c) val g = compCompA1 compT;

d) val h = g real;

e) f(2);

f) h(3.5);

g) h(3.5)

演習 5.6.6 以下の関数 filter を P だけを引数として書き換えよ．すなわちリストを受け取って P を満たすものだけのリストを作る関数を生成するものとして filter を書き直せ．

```
fun filter(P, nil) = nil
|   filter(P, x::xs) =
       if P(x) then x::filter(P, xs)
       else filter(P,xs);
```

演習 5.6.7 foldr と匿名関数を用いて実数のリスト $[a_0, a_1, \ldots, a_{n-1}]$ を取り，引数 b によって以下の多項式を計算する関数を生成する関数を書け.

$$a_0 + a_1 x + a_2 x^2 + \ldots + a_{n-1} x^{n-1}$$

$x = b$ で，これは $\Sigma_{i=0}^{n-1} a_i b^i$ を計算する.

演習 5.6.8 以下の二引数の関数 simpleMap をカリー化形式で書け．この動作が一引数の関数 map とまったく同じであることを示せ.

二引数の simpleMap のコード

```
fun simpleMap (F,nil) = nil
| simpleMap(F, x::xs) = F(x)::simpleMap(F, xs);
```

関数 map のコード

```
fun map F =
    let
        fun M nil = nil
        |   M (x::xs) = F x :: M xs
    in
        M
    end;
```

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

演習 5.7

演習 5.7.1 2.1.1 節で説明した ML の実数定数の文法図を示せ.
演習 5.7.2 ML では，われわれは整数を角括弧を使ってリスト表現したり，丸括弧を使って組表現したりできる．たとえば 、[(1,2),(3,4)] と書ける.

a) この二つのルールからなる値の集合の文法図を示せ．ML が，リストの場合には要素は同じ型でなければならないことは，含まれなくて構わない.

b) このクラスのパーサを実装せよ．入力は instream IN から読み込む．パーサの結果はブール型とする．すなわち，入力が正しい形式になっていれば true, そうでなければ false を出力とする．整数，括弧，カンマの間に空白があるのは許されるが，一つの整数の間に空白があるのは許されない．入力の最後は $ で終わるものとする.

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