非手続き型言語1回目課題 解答例

April 23, 2020

1 演習問題 3.1.1 e) – f)

```
(* c  -- e で使う*)
fun thirdelement (L) = hd(tl(tl(L)));

(* e *)
fun thirdchar (S) = thirdelement(explode S);

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

2 演習問題 3.1.2 a) – d)

```
(* a *)
fun minmaxpair (a, b, c) =
    if a > b then
       if a > c then
          if b > c then (c, a)
          else          (b, a)
       else             (b, c)

    else
       if b > c then
          if a > c then (c, b)
          else          (a, b)
       else             (a, c);


(* 3.1.2 a) 別解 *)
fun min3 (a, b, c) =
  if a < b then
```

```
     if a < c then a
     else  c
  else if b < c then b
     else c;

fun max3 (a, b, c) =
  if a > b then
     if a > c then a
     else  c
  else if b > c then b
     else c;

fun minmaxpair2(a,b,c) = (min3(a,b,c), max3(a,b,c));


(* b *)
fun sorted3tuple (a, b, c) =
    if a > b then
       if a > c then
          if b > c then [c, b, a]
          else          [b, c, a]
       else             [b, a, c]

    else
       if b > c then
          if a > c then [c, a, b]
          else          [a, c, b]
       else             [a, b, c];


(* 別解 *)
fun mid3(a,b,c) =
  if(a < b) then if (b < c) then b
                 else if a > c then a
                      else c
  else if (a < c) then a
       else if (b < c) then c
       else b;

fun sorted3tuple2(a,b,c)=
    [min3(a,b,c), mid3(a,b,c), max3(a,b,c)];

(* c *)
fun round10 (x) = round (x / 10.0) * 10;

(* d *)
```

```
fun deletesecond (L) = hd(L)::tl(tl(L));
```

3 実行結果

```
- use "ML1answer.ml";
[opening ML1answer.ml]
val thirdelement = fn : 'a list -> 'a
val thirdchar = fn : string -> char
ML1answer.ml:11.8 Warning: calling polyEqual
val cyclelist = fn : ''a list -> ''a list
val minmaxpair = fn : int * int * int -> int * int
val min3 = fn : int * int * int -> int
val max3 = fn : int * int * int -> int
val minmaxpair2 = fn : int * int * int -> int * int
val sorted3tuple = fn : int * int * int -> int list
val mid3 = fn : int * int * int -> int
val sorted3tuple2 = fn : int * int * int -> int list
val round10 = fn : real -> int
val deletesecond = fn : 'a list -> 'a list
val it = () : unit
- thirdchar ("abcdef");
val it = #"c" : char
- cyclelist ([1,2,3,4,5]);
val it = [2,3,4,5,1] : int list
- minmaxpair (10, 5, ~3);
val it = (~3,10) : int * int
- sorted3tuple(33, ~6, ~19);
val it = [~19,~6,33] : int list
- round10(113.7);
val it = 110 : int
- deletesecond ([1,2,3,4,5]);
val it = [1,3,4,5] : int list
-
```

非手続き型言語 2 回目課題 解答例

April 27, 2020

1　関数 comb (第 2 回のスライドの問題)

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

2　演習問題 3.2.1 d) − f)

```
(* 3.2.1 d)*)

fun ex321d (L) = if L = [] then 0
                    else 1 + ex321d(tl(L));

(* 3.2.1 e)*)

fun ex321e (x, i) = if i = 0 then 1.0
                         else x * ex321e(x, i-1);

(* 3.2.1 f) 素直な考え方*)
fun ex321f (L) = if tl(L) = [] then hd(L)
else if hd(L) > ex321f(tl(L)) then hd(L)
     else ex321f(tl(L));

(* ex321f の再帰呼び出しを減らす工夫をした解答 *)
fun maxList(L) =
    if tl(L) = nil then hd(L)
    else (* 少なくとも二つの要素がある場合 *)
       if hd(L)>hd(tl(L)) then maxList(hd(L)::tl(tl(L)))
       else maxList(tl(L));
```

3　実行結果

```
- use "ML2answer.ml";
[opening ML2answer.ml]
val comb = fn : int * int -> int
ML2answer.ml:7.23 Warning: calling polyEqual
val ex321d = fn : ''a list -> int
val ex321e = fn : real * int -> real
val ex321f = fn : int list -> int
val maxList = fn : int list -> int
val it = () : unit
- comb (5,2);
val it = 10 : int
- ex321d [];
val it = 0 : int
- ex321d (["abc", "c", "dd"]);
val it = 3 : int
- ex321e (2.0, 4);
val it = 16.0 : real
- ex321f ([4,1,5,~2]);
val it = 5 : int
-
```

非手続き型言語3回目課題 解答例

April 30, 2020

1 関数 mygcd (第3回のスライドの問題)

```
fun mygcd(a, 0) = a
| mygcd(a, b) = mygcd(b, a mod b);
```

2 演習問題3.3.2

```
fun swapElement ([]) = []
|   swapElement ([x]) = [x]
|   swapElement  (x::y::zs) = y::x::swapElement(zs);
```

3 演習問題3.3.3

```
(* 先頭が 1 番目 *)
fun deleteIElement ([], i) = []
|   deleteIElement (x::xs, 1) = xs
|   deleteIElement (x::xs, i) = x::deleteIElement(xs, i-1);
```

4 演習問題3.3.11

```
(* a *)
fun member(x,[]) = false
 |  member(x, y::ys) = if x = y then true
                       else member(x, ys);
(*
Warning: calling polyEqual が出るが、これは無視してよい
*)

(* orelse を使ったこういうのもできる これも  Warning: calling polyEqual が
でる*)
fun member2(x,[]) = false
|   member2(x,y::ys) =
        x=y orelse member(x,ys);
```

```
(* b *)
fun delete(x,[]) = nil
 |  delete(x,y::ys) = if x = y then ys
                      else y::delete(x,ys);
(*
Warning: calling polyEqual が出るが、これは無視してよい
*)
(* c *)
Fun insert(x,S) = if member(x, S) then S
                  else x::S;

(* member を使わない 以下のものでもよい*)
(* この解答例で as は別名をつけるというもの. S as y::ys は, S として
も y::ys としても同じリストを意味するという意味 *)

fun insert2(x,[]) = [x]
|   insert2(x,S as y::ys) =
        if x=y then S else y::insert2(x,ys);
```

5 実行結果

```
- use "ML3answer.ml";
[opening ML3answer.ml]
val mygcd = fn : int * int -> int
val swapElement = fn : 'a list -> 'a list
val deleteIElement = fn : 'a list * int -> 'a list
ML3answer.ml:33.29 Warning: calling polyEqual
val member = fn : ''a * ''a list -> bool
ML3answer.ml:42.10 Warning: calling polyEqual
val member2 = fn : ''a * ''a list -> bool
ML3answer.ml:47.28 Warning: calling polyEqual
val delete = fn : ''a * ''a list -> ''a list
val insert = fn : ''a * ''a list -> ''a list
ML3answer.ml:62.13 Warning: calling polyEqual
val insert2 = fn : ''a * ''a list -> ''a list
val it = () : unit
- mygcd (100, 55);
val it = 5 : int
- swapElement([2,3,4,5,6]);
val it = [3,2,5,4,6] : int list
- deleteIElement ([1,2,3,4,5], 3);
val it = [1,2,4,5] : int list
- member (3, [1,2,3,4,5]);
```

```
val it = true : bool
- delete (3, [1,2,3,4,5]);
val it = [1,2,4,5] : int list
- insert (7, [1,2,3,4,5]);
val it = [7,1,2,3,4,5] : int list
- insert2 (7, [1,2,3,4,5]);
val it = [1,2,3,4,5,7] : int list
```

非手続き型言語4回目課題 解答例

May 7, 2020

1　演習問題 3.3.13

```
fun inserteach(_,nil) = nil  (* これは演習問題 3.3.12 の解答 *)
 |  inserteach(a,x::xs) = (a::x)::inserteach(a,xs);

fun powerset(nil) = [nil]
 |  powerset(x::xs) = powerset(xs)@inserteach(x,powerset(xs));
```

2　演習問題 3.4.3

```
(* Ex. 3.3.13 の局所環境版 *)
fun PowerSet(nil) = [nil]
 |  PowerSet(x::xs) =
    let
        val s = PowerSet(xs)
    in
        s@inserteach(x,s)
    end;
```

3　演習問題 3.4.4

```
(* 演習問題 3.2.1 f のときはこうしたはず
fun maxnumber2(x::nil) = x:real
 |  maxnumber2(x::xs) = if x > maxnumber2(xs) then x
                      else maxnumber2(xs);
*)

fun maxnumber2(x::nil) = x:real
 |  maxnumber2(x::xs) =
     let
       val m = maxnumber2(xs)
     in
       if x > m then x
       else m
     end;

(* こういう解答でもいい *)
fun maxnumber3(x::nil) = x:real
|   maxnumber3(x::xs) =
   let
     fun max(a,b) = if a > b then a else b
   in
     max(x, maxnumber3(xs))
   end;

(* 上の解答例では Warning : match nonexhaustive が出る.
   例外を用いてそれが出ないようにした改良版*)

exception NullList;
fun maxnumber4(nil) = raise NullList
 | maxnumber4(x::nil) = x:real
 |  maxnumber4(x::xs) =
     let
       val m = maxnumber4(xs)
     in
       if x > m then x
       else m
       end;
```

4 演習問題 5.2.1

```
exception TooShort;
fun thirdelement(nil) = raise TooShort
|   thirdelement(x::nil) = raise TooShort
|   thirdelement(x::y::nil) = raise TooShort
|   thirdelement(x::y::z::zs) = z;

(*スマートな解答*)
exception TooShort;
fun   thirdelement(x::y::z::zs) = z
|     thirdelement(_) = raise TooShort;
```

5 演習問題 5.2.2

```
exception Negative;
fun fact(0) = 1
|   fact(n) =
     if n > 0 then n * fact(n-1)
     else raise Negative;
```

6 実行結果

```
- use "ML4answer.ml";
[opening ML4answer.ml]
val inserteach = fn : 'a * 'a list list -> 'a list list
val powerset = fn : 'a list -> 'a list list
val PowerSet = fn : 'a list -> 'a list list
ML4answer.ml:26.5-33.9 Warning: match nonexhaustive
          x :: nil => ...
          x :: xs => ...

val maxnumber2 = fn : real list -> real
ML4answer.ml:36.5-42.7 Warning: match nonexhaustive
          x :: nil => ...
          x :: xs => ...

val maxnumber3 = fn : real list -> real
exception NullList
val maxnumber4 = fn : real list -> real
exception TooShort
val thirdelement = fn : 'a list -> 'a
exception TooShort
val thirdelement = fn : 'a list -> 'a
exception Negative
val fact = fn : int -> int
val it = () : unit
- powerset ([1,2,3]);
val it = [[],[3],[2],[2,3],[1],[1,3],[1,2],[1,2,3]] : int list list
- PowerSet ([1,2,3]);
val it = [[],[3],[2],[2,3],[1],[1,3],[1,2],[1,2,3]] : int list list
- maxnumber2 ([3.0, 1.2, ~1.1, 4.0]);
val it = 4.0 : real
- maxnumber4 ([]);

uncaught exception NullList
  raised at: ML4answer.ml:48.29-48.37
- thirdelement ([1,2,3,4,5]);
val it = 3 : int
- thirdelement ([1,2]);

uncaught exception TooShort
  raised at: ML4answer.ml:79.31-79.39
```

```
- fact (5);
val it = 120 : int
- fact (~3);

uncaught exception Negative
  raised at: ML4answer.ml:88.17-88.25
-
```

非手続き型言語5回目課題 解答例

May 11, 2020

1　局所環境を使った関数 rev

```
fun rev(L) =
    let
        fun rev1(M, []) = M
        | rev1(M, x::xs) = rev1(x::M, xs)
    in
        rev1([],L)
    end;
```

2　演習問題 3.5.2

ここでは局所環境を使わない解答例を示すが，局所環境を用いた方がさらによい．

```
fun cat(nil,M) = M
|   cat(x::xs,M) = x::cat(xs,M);

fun cycle1(L, M, 0) = cat(L, rev(M))
|   cycle1(x::xs, M, i) = cycle1(xs, x::M, i-1);

fun cycle(L,i) = cycle1(L, nil, i);
```

関数 rev を使わない別解 (問題文のヒントはこちらのことを言っていると思われる)．

```
fun split (nil, ys, i) = (nil, ys)
| split (x::xs, ys, 0) = (x::xs, ys)
|  split (x::xs, ys, i) =
     let
       val (M, N) = split(xs, ys, i-1)
     in
       (M, x::N)
     end;

fun cat(nil,M) = M
```

```
|   cat(x::xs,M) = x::cat(xs,M);

fun cycle2(L, i) =
     let
        val (M, N) = split(L, nil, i)
     in
      cat(M, N)
     end;
```

3　1 から n までの総和を求める末尾再帰の関数

```
fun sum(n) =
    let
        fun f1(0, s) = s
        | f1(n, s) = f1(n-1,s+n)
    in
        f1(n,0)
    end;
```

4　実行結果

```
- use "ML5answer.ml";
[opening ML5answer.ml]
val rev = fn : 'a list -> 'a list
val cat = fn : 'a list * 'a list -> 'a list
ML5answer.ml:15.5-16.48 Warning: match nonexhaustive
          (L,M,0) => ...
          (x :: xs,M,i) => ...

val cycle1 = fn : 'a list * 'a list * int -> 'a list
val cycle = fn : 'a list * int -> 'a list
val split = fn : 'a list * 'a list * int -> 'a list * 'a list
val cat = fn : 'a list * 'a list -> 'a list
val cycle2 = fn : 'a list * int -> 'a list
val sum = fn : int -> int
val it = () : unit
- rev ([1,2,3,4,5]);
val it = [5,4,3,2,1] : int list
- cycle ([1,2,3,4,5], 4);
val it = [5,1,2,3,4] : int list
- cycle2 ([1,2,3,4,5], 4);
val it = [5,1,2,3,4] : int list
- sum (10);
val it = 55 : int
```

非手続き型言語6回目課題 解答例

May 14, 2020

1 演習問題3.6.3

```
fun eval(nil, _) = 0.0
|  eval(x::xs, a) = x + a * eval(xs, a);
```

2 演習問題3.6.4

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

fun smult(nil,(q:real)) = nil
|   smult(p::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 getPoly(nil) = [1.0]
|   getPoly(x::xs) = pmult([~x, 1.0], getPoly(xs));

(* これは間違い
fun getPoly2(nil) = nil  <- ここが違う．掛け算していくので基底は1.0で
ないとダメ
|   getPoly2(x::xs) = pmult([~x, 1.0], getPoly2(xs));
*)
```

3 演習問題3.6.5

```
(* 加算 *)
fun mpadd(PP, nil) = PP
|   mpadd(nil, QQ) = QQ
|   mpadd(P::PS, Q::QS) = padd(P,Q)::mpadd(PS, QS);
```

```
(*スカラー倍 *)
fun mpsmult(nil, q) = nil
| mpsmult(P::PS, q) = smult(P, q)::mpsmult(PS,q);

(* 積 *)
(* 補助関数としてリストのリストとリストをかける関数 mppmult(PP, Q)*)
(*
PP = P + PSx, Q
PP * Q = P* Q + PS * Q x
PP はリストのリスト。x と y がまざっている
P, Q はリスト.ここでいうなら y のみ.
*)

fun mppmult(nil, Q) = nil
|   mppmult(P::PS, Q) = mpadd([pmult(P,Q)], [0.0]::mppmult(PS,Q));


(* 二つの変数を含む多項式の乗算 *)
(* 二つの多項式を  PP と QQ = Q+QSx とすると *)
(* PP * QQ = PP * Q + PP * QS x *)
(* つまり *)

fun mpmult(PP, []) = []
|   mpmult(PP, Q::QS) = mpadd(mppmult(PP, Q), [0.0]::mpmult(PP, QS));
```

4 実行結果

```
- use "ML6answer.ml";
[opening ML6answer.ml]
val eval = fn : real list * real -> real
val padd = fn : real list * real list -> real list
val smult = fn : real list * real -> real list
val pmult = fn : real list * real list -> real list
val getPoly = fn : real list -> real list
val mpadd = fn : real list list * real list list -> real list list
val mpsmult = fn : real list list * real -> real list list
val mppmult = fn : real list list * real list -> real list list
val mpmult = fn : real list list * real list list -> real list list
val it = () : unit
- val P = [[1.0,1.0]];
val P = [[1.0,1.0]] : real list list
- val Q = [[1.0,1.0]];
val Q = [[1.0],[1.0]] : real list list
```

```
- eval ([1.0, 1.0], 5.0);
val it = 6.0 : real
- getPoly ([1.0, 2.0, 3.0]);
val it = [~6.0,11.0,~6.0,1.0] : real list
- mpadd (P, Q);
val it = [[2.0,1.0],[1.0]] : real list list
- mpsmult (P, 2.0);
val it = [[2.0,2.0]] : real list list
- mpmult (P, Q);
val it = [[1.0,1.0],[1.0,1.0]] : real list list
```

非手続き型言語7回目課題 解答例

May 18, 2020

(今回の課題は第 7 回講義のスライドにある通りに解答すればよい．独自に調べたことを付け加えればさらによいことはいうまでもない．しかし，調べ始めるときりがないほどラムダ計算は奥が深いのでこの課題の解答としてはほどほどのところで止めるのがよいだろう．)

1　ラムダ計算とは何か

(スライド 2, 3, 4 枚目の内容を適宜まとめる)

ラムダ計算 (lambda calculus) とは計算の実行を関数への引数の評価 (evaluation) と適用 (application) としてモデル化・抽象化した計算体系である．関数と関数の計算結果とを明確に区別するラムダ記法を用いて記述されたラムダ式を用いる．関数型言語の理論的基礎である．

2　ラムダ式の定義を述べよ

ラムダ式 (λ 式) の定義は以下である．(スライド 4 ページ目)

1. 変数 $x_0, x_1, \ldots$ は λ 式.
2. M が λ 式で x が変数のとき $(\lambda x.M)$ は λ 式.
3. M と N が λ 式のとき (MN) は λ 式.