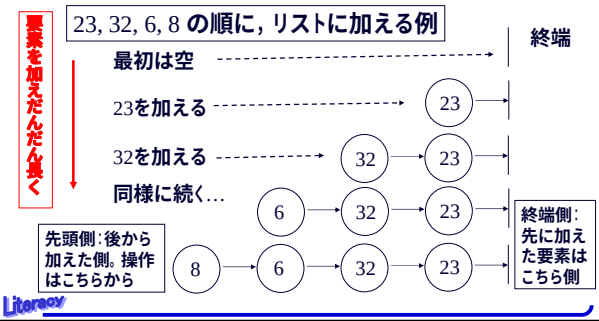
第6回
任意長の合成データ：リスト

情報処理演習II

Literacy

リストとは

- 不定数のデータの並び(合成データ)
- データの並びには順序があり操作は先頭側から



Literacy

リスト

- 要素の列挙での記述 `(list 要素の並び)`

  (例) `(list 15 8 6 32 23)`

- リストに関する他の演算子

  `(first a_list)`はリスト `a_list` の先頭の要素

  `(rest a_list)`はリスト `a_list` の先頭を除いた残り

  `(list? a_list)`はリスト `a_list` がリストかどうか判断

  `(empty? a_list)`はリスト `a_list` が空リストか判断

Literacy

例題

Literacy

例題1. リストの n 番目の要素

- リストの n 番目の要素 (先頭は 1 番目) を得る関数 `my-list-ref` を定義し，実行する。

入力は1つのリストと1つの数値 ⇒ `my-list-ref` ⇒ 出力は指定番目の要素

`(list 11 12 13 14)` `2` → `12`

Literacy

リスト処理の再帰関数のテンプレート

リスト処理関数一般のテンプレート

```
(define (fun_for_list a-list)
  (cond
    [(empty? a-list) …]
    [else … (first a-list) …
          … (fun_for_list (rest a-list)) …]))
```

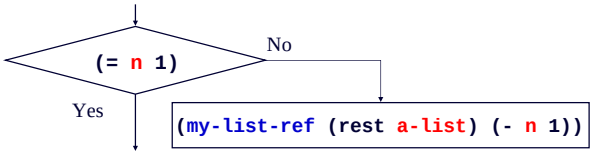
今回の関数 `my-list-ref` のテンプレート

```
(define (my-list-ref a-list n)
  (cond
    [(= n 1) …]
    [else … (my-list-ref (rest a-list)
                         (- n 1)) …]))
```

Literacy

例題1.リストの n 番目の要素

1. n = 1 ならば: (終了条件)
   解は (first a-list) (自明な解)
2. そうでなければ:
   – 「リストの rest 部」の「n−1 番目の要素」が求める解



(first a-list) が自明な解

Literacy

例題1.リストの n 番目の要素

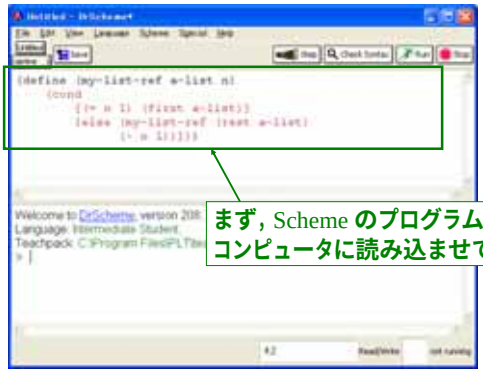
```
;; my-list-ref: list number -> anything
;; get the n-th element of a list
;; (my-list-ref (list 11 12 13 14) 2) = 12
(define (my-list-ref a-list n)
    (cond 終了条件 自明な解
        [(= n 1) (first a-list)]
        [else (my-list-ref (rest a-list)
                (- n 1))]))
```

my-list-ref の内部に my-list-ref (再帰による繰り返し)

例: (my-list-ref (list 11 12 13 14) 2)
= (my-list-ref (list 12 13 14) 1)

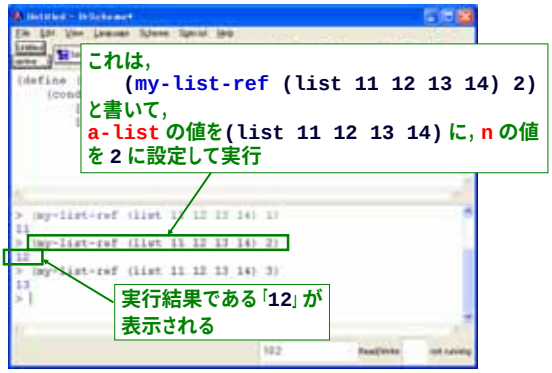
Literacy

例題1の実行例(1/2)



Literacy

例題1の実行例(2/2)



Literacy

(my-list-ref (list 11 12 13 14) 2) から 12が得られる過程

```
(my-list-ref (list 11 12 13 14) 2)  最初の式
= (cond
    [(= 2 1) (first (list 11 12 13 14))]
    [else (my-list-ref (rest (list 11 12 13 14) ) (- 2 1))])
= (cond
    [false (first (list 11 12 13 14))]
    [else (my-list-ref (rest (list 11 12 13 14) ) (- 2 1))])
= (my-list-ref (rest (list 11 12 13 14) ) (- 2 1))
= (my-list-ref (list 12 13 14) (- 2 1))
= (my-list-ref (list 12 13 14) 1)
= (cond
    [(= 1 1) (first (list 12 13 14))]
    [else (my-list-ref (rest (list 12 13 14) ) (- 1 1))])
= (cond
    [true (first (list 12 13 14))]
    [else (my-list-ref (rest (list 12 13 14) ) (- 1 1))])
= (first (list 12 13 14))                    コンピュータ内部での計算
= 12  実行結果
```

Literacy

(my-list-ref (list 11 12 13 14) 2) から 12が得られる過程

```
(my-list-ref (list 11 12 13 14) 2)
= (cond
    [(= 2 1) (first (list 11 12 13 14))]
    [else (my-list-ref (rest (list 11 12 13 14) ) (- 2 1))])
= (cond
    [false (first (list 11 12 13 14))]
    [else (my-list-ref (rest (list 12 13 14) ) (- 1 1))])
= (first (list 12 13 14))
= 12
```

これは,
(define (my-list-ref a-list n)
  (cond
    [(= n 1) (first a-list)]
    [else (my-list-ref (rest a-list) (- n 1))]))
の a-list を (list 11 12 13 14) で, n を 2 で置き換えたもの。

Literacy

(my-list-ref (list 11 12 13 14) 2) から 12が得られる過程の概略

(my-list-ref (list 11 12 13 14) 2)

リストの2番目の要素

= …

= (my-list-ref (list 12 13 14) 1)

リストの rest を取って，その1番目の要素

= …

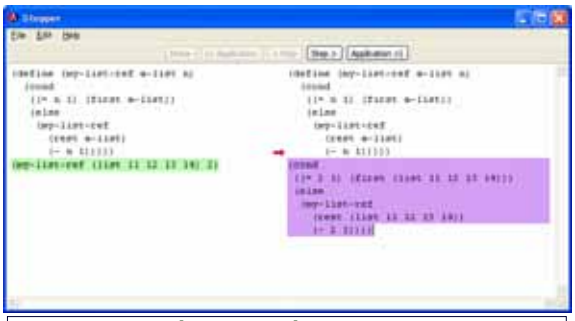
= (first (list 12 13 14))

リストの先頭

= 12

Literacy

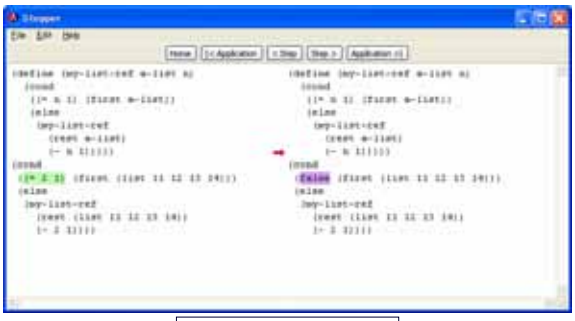
my-list-ref関数のステップ実行(1/10)



my-list-ref の「a-list」は「(list 11 12 13 14)」で，「n」は「2」で置き換わる。

Literacy

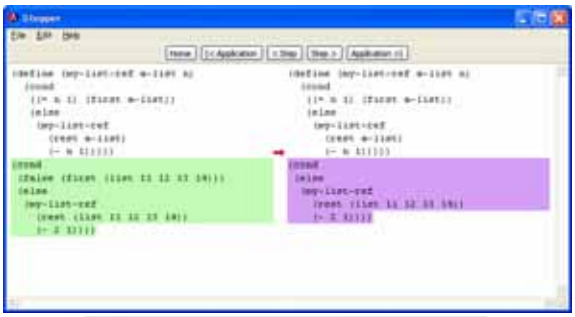
my-list-ref関数のステップ実行(2/10)



「(= 2 1)」は「false」で置き換わる。

Literacy

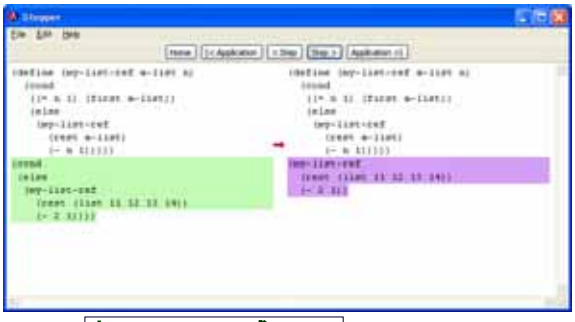
my-list-ref関数のステップ実行(3/10)



「(cond [false 式X] [else 式Y])」は「else 式Y」節が残る。

Literacy

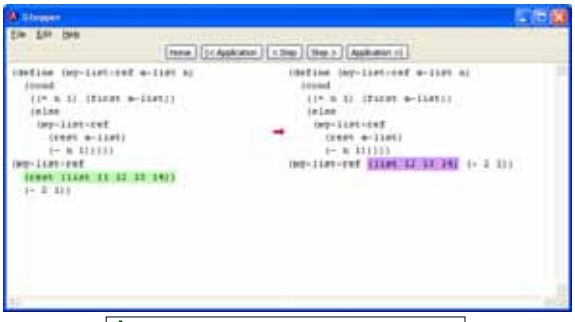
my-list-ref関数のステップ実行(4/10)



「(cond [else 式Y])」は「式Y」で置き換わる。

Literacy

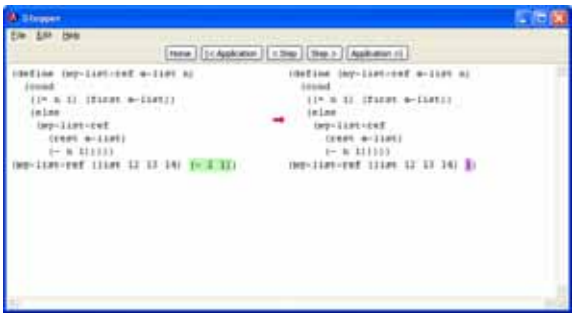
my-list-ref関数のステップ実行(5/10)



「(rest (list 11 12 13 14))」は「(list 12 13 14)」で置き換わる。

Literacy

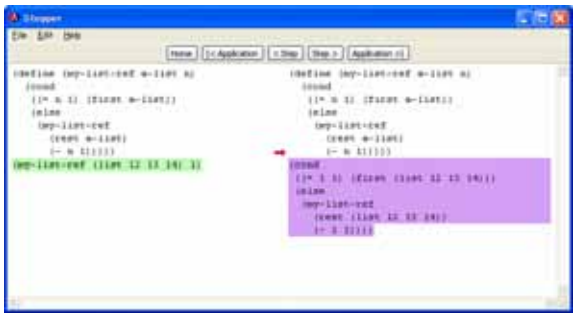
my-list-ref関数のステップ実行(6/10)



「(- 2 1)」は「1」で置き換わる。

Literacy

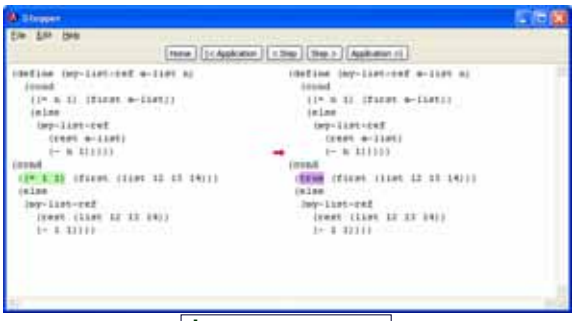
my-list-ref関数のステップ実行(7/10)



my-list-refの「a-list」は「(list 12 13 14)」で，「n」は「1」で置き換わる。

Literacy

my-list-ref関数のステップ実行(8/10)



「(= 1 1)」は「true」で置き換わる。

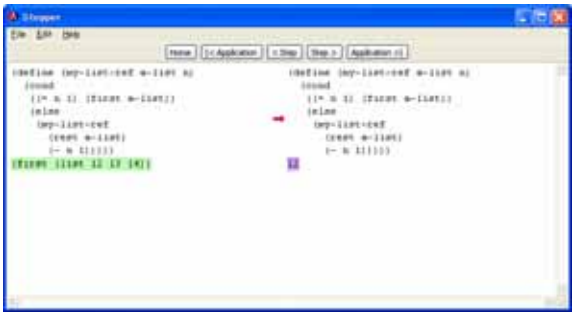
Literacy

my-list-ref関数のステップ実行(9/10)

「(cond [true 式X] [else 式Y])」は「式X」で置き換わる。

Literacy

my-list-ref関数のステップ実行(10/10)



「(first (list 12 13 14))」は「12」で置き換わる。

Literacy

実習

Literacy

実習の進め方

- 資料を見ながら，「実習」を行ってみる
- その後，各自「課題」に挑戦する
  - 各自で自習 ＋ 巡回指導
  - 遠慮なく質問してください
- 自分のペースで先に進んで構いません

Literacy

DrScheme の使用

- DrScheme の起動
  - プログラム → PLT Scheme → DrScheme
- 今日の実習では「Intermediate Student」に設定
  - Language
  - → Choose Language
  - → Intermediate Student
  - → OK ボタン

Literacy

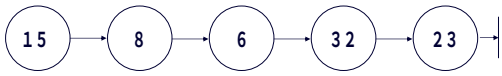
DrScheme の使用 / ステップ実行

- プログラム実行の振る舞いを観察するツール
- 「定義用ウインドウ」のみを使用
  （通常の実行と異なる）
- 「Intermediate Student」に設定する必要あり
  - Language → Choose Language
  - → Intermediate Student
  - → Run ボタン

Literacy

実習1．リスト

「list」を使ってリストを書く



手順：次の式を「実行用ウインドウ」で実行

```
(list 15 8 6 32 23)
```

Literacy

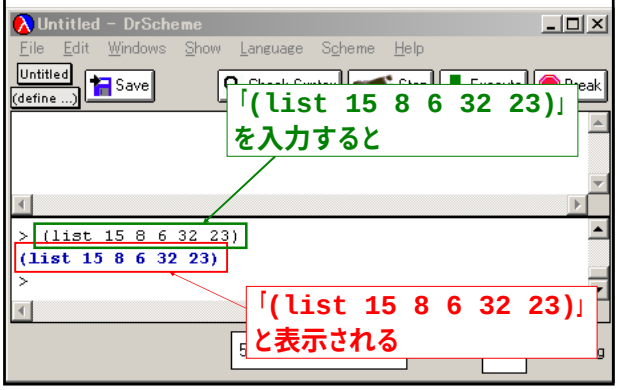
「実習1．リスト」の手順

1. 次を「実行用ウインドウ」で実行しなさい

```
(list 15 8 6 32 23)
```

Literacy

実行結果の例



コンピュータが行っていること

Scheme の式

(list 15 8 6 32 23)
を入力すると・・・

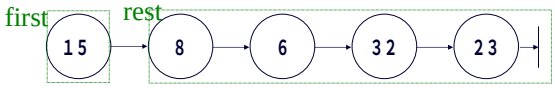
インタプリタ
DrScheme

(list 15 8 6 32 23)
がそのまま表示される

式の実行結果

実習2．リストの first と rest

■リスト (list 15 8 6 32 23) に対して，first と rest を実行する



手順：次の式を「実行用ウインドウ」で実行

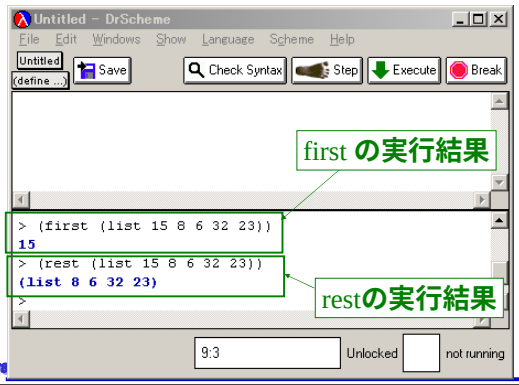
```
(first (list 15 8 6 32 23))
(rest (list 15 8 6 32 23))
```

「実習2．リストの first と rest」の手順

1. 次を「実行用ウインドウ」で実行しなさい

```
(first (list 15 8 6 32 23))
(rest (list 15 8 6 32 23))
```

実行結果の例



関数 first

Scheme の式

(first (list 15 8 6 32 23))
を入力すると・・・

インタプリタ
DrScheme

15
が表示される

式の実行結果

リストの先頭の要素

関数 rest

Scheme の式

(rest (list 15 8 6 32 23))
を入力すると・・・

インタプリタ
DrScheme

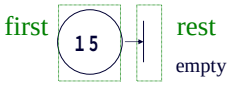
(list 8 6 32 23)
が表示される

式の実行結果

リストから先頭の要素を
除いた残り（長さは1短い）

実習3. リストの first と rest

■ 要素が1つしか無いリスト (list 15)に対して, first と rest を実行する



手順:次の式を「実行用ウインドウ」で実行

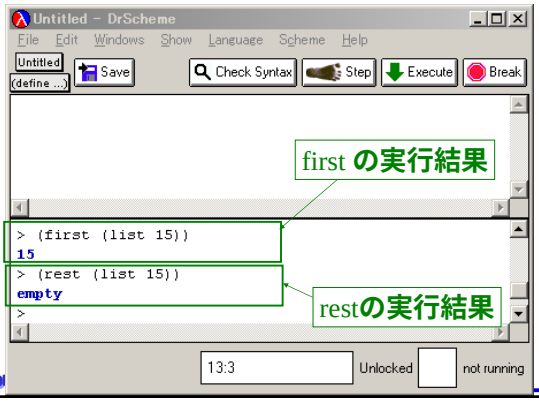
```
(first (list 15))
(rest (list 15))
```

「実習3. リストの first と rest」の手順

1. 次を「実行用ウインドウ」で実行しなさい

```
(first (list 15))
(rest (list 15))
```

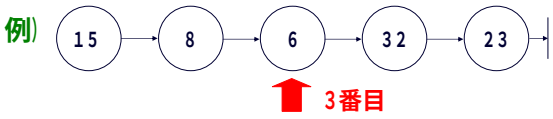
実行結果の例



実習4. リストの基本操作

リストの第三要素を得る関数 element3 を定義し, 実行する

例)



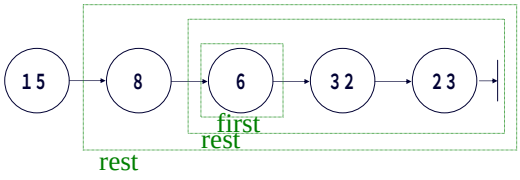
入力 ⇒ element3 ⇒ 出力

(list 15 8 6 32 23) 6

rest関数とfirst関数による第三要素の抽出

first と rest を使い第三要素を求める関数

＝ リストの rest の rest の first



第三要素を求める element3 関数

「関数である」ことを示すキーワード

関数の名前

値を1つ受け取る (入力)

```
(define (element3 a-list)
  (first (rest (rest a-list)))
)
```

a-list の値から
3番目の要素を求める (出力)

「実習４．リストの基本操作」の手順

1. 次を「定義用ウインドウ」で，実行しなさい
   - 入力した後に，Run ボタンを押す

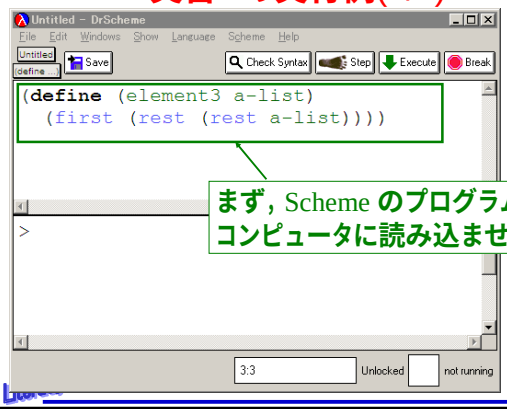
```
;; element3: list of anything -> anything
;; Get the 3rd element of a list
;; (list-sum (list 15 8 6 32 23)) = 6
(define (element3 a-list)
    (first (rest (rest a-list))))
```

2. その後，次を「実行用ウインドウ」で実行しなさい

```
(element3 (list 1 2 3 4))
(element3 (list 15 8 6 32 23))
```

Literacy

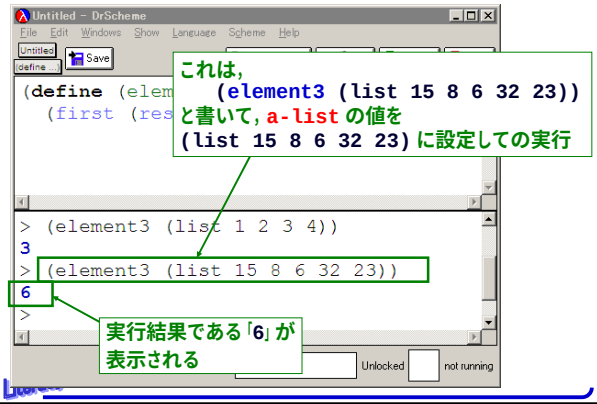
実習４の実行例(1/2)



まず，Scheme のプログラムを
コンピュータに読み込ませている

Literacy

実習４の実行例(2/2)



これは，
(element3 (list 15 8 6 32 23))
と書いて，a-list の値を
(list 15 8 6 32 23) に設定しての実行

実行結果である「6」が
表示される

Literacy

(element3 (list 15 8 6 32 23)) から 6が得られる過程

(element3 (list 15 8 6 32 23)) 最初の式

```
= (first (rest (rest (list 15 8 6 32 23))))
= (first (rest (list 8 6 32 23)))
= (first (list 6 32 23))
```

コンピュータ内部での計算

= 6 実行結果

Literacy

(element3 (list 15 8 6 32 23)) から 6が得られる過程

```
(element3 (list 15 8 6 32 23))
= (first (rest (rest (list 15 8 6 32 23))))
= (first (rest (list 8 6 32 23)))
= (
= 6
```

これは，
(define (element3 a-list)
(first (rest (rest a-list))))
の a-list を (list 15 8 6 32 23) で
置き換えたもの。

Literacy

実習４の関数 element3 について

- リストの長さが2以下の時には，「エラーメッセージ」が表示される

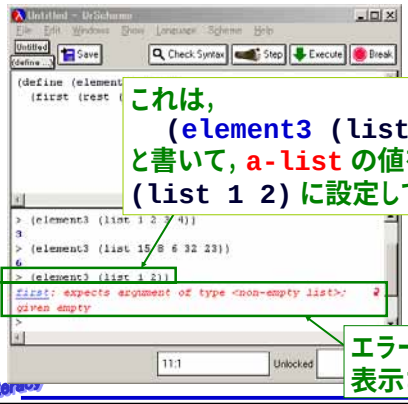
例)

```
(element3 (list 1 2))
```

→ エラーメッセージが表示される

Literacy

element3関数で起こるエラーの例



Literacy

(element3 (list 1 2)) から実行エラーに至る過程

(element3 (list 1 2)) 最初の式

(first (rest (rest a-list)))
に a-list = (list 1 2) が代入される

= (first (rest (rest (list 1 2))))

(rest (list 1 2)) → (list 2)

= (first (rest (list 2)))

(rest (list 1)) → empty

= (first empty)

コンピュータ内部での計算

→ 「空リスト empty に対して first を実行できない」という決まりがあるので, 実行エラーになる。

Literacy

実習5. 全ての要素が10以上か？

数値リスト a-list の要素を調べ, 全ての要素が10以上のとき (＝10未満の要素が無いとき) にのみ true を返す関数 all-are-large を定義し, 実行する

入力 → all-are-large → 出力

数値のリスト
例: (list 10 11 12)

ブール値
例: true

Literacy

リスト処理の再帰関数のテンプレート

リスト処理関数一般のテンプレート

```
(define (fun_for_list a-list)
  (cond
    [(empty? a-list) …]
    [else … (first a-list) …
          … (fun_for_list (rest a-list)) …]))
```

今回の関数 all-are-large のテンプレート

```
(define (all-are-large alon)
  (cond
    [(empty? alon) …]
    [else … (first alon) …
          … (all-are-large (rest alon)) …]))
```

Literacy

実習5. 全ての要素が10以上か？

```
;; all-are-large: list number -> boolean
;; If some elements are "<10",
;; it returns false
;; (all-are-large (list 10 11 12) = true
(define (all-are-large alon)
  (cond    終了条件      自明な解
    [(empty? alon) true]
    [else (and (<= 10 (first alon))
               (all-are-large (rest alon)))]))
```

all-are-large の内部に all-are-large (再帰による繰り返し)
例: (all-are-large (list 10 11 12))
= (all-are-large (list 11 12))

Literacy

「数値リストの要素の和の手順

1. 次を「定義用ウインドウ」で, 実行しなさい
   - 入力した後に, Run ボタンを押す

```
;; all-are-large: list number -> boolean
;; If some elements are "<10", it returns false
;; (all-are-large (list 10 11 12) = true
(define (all-are-large alon)
  (cond
    [(empty? alon) true]
    [else (and (<= 10 (first alon))
               (all-are-large (rest alon)))]))
```

2. その後, 次を「実行用ウインドウ」で実行しなさい

```
(all-are-large (list 10 11 12))
(all-are-large (list 7 8 9))
```

Literacy

実習6. 構造体のリスト

AddressNote 構造体のリストから，年齢が全て12歳以下であるかを調べる関数 all_are_child を作る

- 12 歳以上の人がいなければ true
- 構造体1つで1人分 → 構造体のリストで複数人分

AddressNote
name
age
address

入力は AddressNote 構造体のリスト

```
(list
 (make-AddressNote "Ken" 11 "Fukuoka")
 (make-AddressNote "Bill" 30 "Saga")
 (make-AddressNote "Mike" 8 "Nagasaki"))
```

all_are_child

出力は boolean

```
false
```

Literacy

リスト処理の再帰関数のテンプレート

リスト処理関数一般のテンプレート

```
(define (fun_for_list a_list)
  (cond
    [(empty? a_list) …]
    [else … (first a_list) …
            … (fun_for_list (rest a_list)) …]))
```

今回の関数 all_are_child のテンプレート

先頭要素が構造体データなのでさらにセレクタを使いフィールドデータを参照

```
(define (all_are_child a_list)
  (cond
    [(empty? a_list) …]
    [else … (AddressNote-age (first a_list))
          … (all_are_child (rest a_list)) …]))
```

Literacy

実習6. 構造体のリスト

```
(define-struct AddressNote
                (name age address))
```

構造体定義

```
;; all_are_child:
;;   a list of AddressNote ->boolean
;; If some AddressNotes whose age is more
;; than 12, it returns false
(define (all_are_child a_list)
  (cond
    [(empty? a_list) true]
    [else (and
           (> 12 (AddressNote-age (first a_list)))
           (all_are_child (rest a_list)))]))
```

終了条件　自明な解

Literacy

「実習6. 構造体のリスト」の手順

1. 次を「定義用ウインドウ」で，実行しなさい
   - 入力した後に，Run ボタンを押す

```
(define-struct AddressNote
                (name age address))
;; all_are_child:
;;   a list of AddressNote ->boolean
;; If some AddressNotes whose age is more
;; than 12, it returns false
(define (all_are_child a_list)
  (cond
    [(empty? a_list) true]
    [else (and (> 12 (AddressNote-age (first a_list)))
               (all_are_child (rest a_list)))]))
```

2. その後，次を「実行用ウインドウ」で実行しなさい

```
(all_are_child (list
                (make-AddressNote "Ken" 11 "Fukuoka")
                (make-AddressNote "Bill" 30 "Saga")
                (make-AddressNote "Mike" 8 "Nagasaki")))
```

Literacy

課題

Literacy

課題1

- 関数first，rest についての問題
  - 次の式の実行結果はどうなるか。解答せよ。

1. (rest (list 1 2 3))
2. (first (rest (list 1 2 3)))
3. (rest (rest (list 1 2 3)))
4. (first (rest (rest (list 1 2 3))))
5. (rest (rest (rest (list 1 2 3))))

Literacy

課題2

以下の組み合わせのリスト演算を実行するとどうなるか

- 「エラー」の場合には，「エラー」と記入すること
- DrScheme では「実行用ウインドウ」を使い確認

	リストが (list 1) の場合	リストが (list 1 2) の場合	リストが (list 1 2 3) の場合
(first リスト) の実行結果			
(rest リスト) の実行結果			
(first (rest リスト)) の実行結果			
(first (rest (rest リスト))) の実行結果			

Literacy

課題3

- 関数 `list-sum` についての問題
  - `(list-sum (list 1 2 3))` から 6 が得られる過程の概略を数行程度で説明しなさい
  - DrScheme のステップ実行機能を使いなさい

```
(define (list-sum a-list)
  (cond
    [(empty? a-list) 0]
    [else (+ (first a-list)
             (list-sum (rest a-list)))]
    ))
```

Literacy

課題4

- DrScheme のステップ実行機能を利用した実行エラーの解決に関する問題
  1. まずは，次を「定義用ウインドウ」で，実行しなさい
     - 入力した後に，Run ボタンを押す
       ⇒ すると，実行用ウインドウに (赤い文字で) エラーメッセージが表示される (これは実行エラー)

```
(define (list-length a-list)
  (cond
    [(empty? a-list) 0]
    [else (+ 1
             (list-length (rest a-list)))]))
(list-length 100)
```

  2. DrScheme で ステップ実行機能 を使って，エラーの箇所を特定しなさい。エラーの原因を分析して，報告しなさい。

Literacy

課題5

- リストの要素の中に「偶数」を含むかどうか調べる関数 `all_are_even` を定義し，実行結果を報告しなさい
  - 偶数を1つでも含めば `true`
  - 偶数を1つも含まなければ `false`
  - 例えば，
    `(all_are_even (list 1 3 4))` = `true`
    `(all_are_evne empty)` = `false`
  - 関数 `even?` を使うこと

Literacy

課題6

- `AddressNote` 構造体のリストから，年齢 (age)の平均を求める関数 `average-age` を定義し，実行結果を報告しなさい

Literacy

課題7

- n次の多項式
  $f(x) = a_0 + a_1 \cdot x + a_2 \cdot x^2 + \cdots + a_n \cdot x^n$
  について，次数 n と，係数 $a_0 \cdots a_n$ から，f(x) を計算するプログラムを定義しなさい
  - 次ページ以降で説明する Horner 法を使うこと
  - 次数 n は入力として与えなくても実現できる

Literacy

多項式の計算

■ n次の多項式

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

について，次数 n，係数 a_0 から a_n と x から f(x) を計算する

Literacy

Horner法による多項式の計算

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

$= a_0 + (a_1 + (a_2 + \cdots + (a_{n-1} + a_n \cdot x) x \cdots) x) x$

例えば，　$5 + 6x + 3x^2$

$= 5 + (6 + 3x) x$

計算手順

① a_n

② $a_{n-1} + a_n \cdot x$

③ $a_{n-2} + (a_{n-1} + a_n \cdot x) x$

$\cdots$ (a_0 まで続ける)

Literacy