第7回
リスト処理関数

情報処理演習II

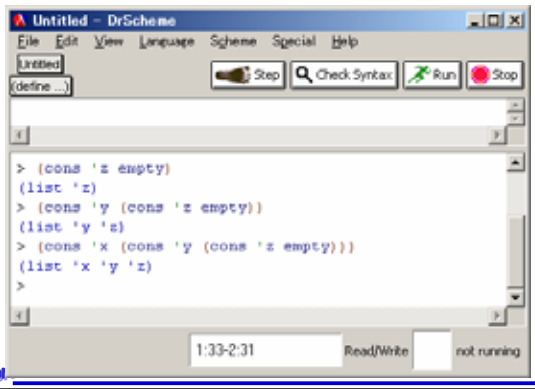
例題

例題1. cons 式

■ 下記の cons 式を Scheme インタプリタ（DrScheme）で実行し，実行結果を確認する

```
(cons 'z empty)
(cons 'y (cons 'z empty))
(cons 'x (cons 'y (cons 'z empty)))
```

実行結果の例



コンピュータが行っていること

Scheme の式

↓

インタプリタ
DrScheme

↓

式の実行結果

(cons 'z empty)
を入力すると･･･

↓

(list 'z)
が表示される

リストの表記

■ cons による表記

例）(cons 'x (cons 'y
(cons 'z empty)))

「empty」は空リストの意味．
ここでは，末尾の意味になる

■ list による表記

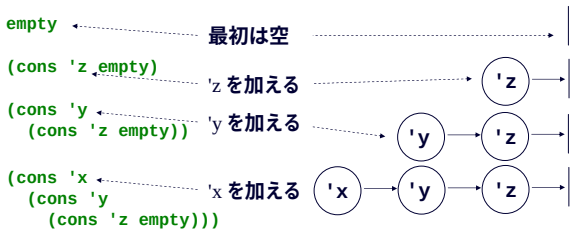
例）(list 'x 'y 'z)

2種類の表記がある

リストの作成：cons

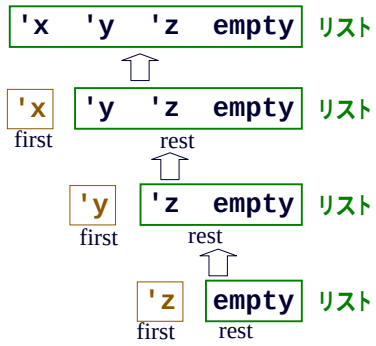
空リスト empty から始め，cons 関数を用いてリストを作成

(cons 追加する先頭要素 追加されるリスト)



Literacy

cons の意味



Literacy

例題2．勤務時間リストから賃金リストを生成

- ある賃金を求める関数 wage を定義する
  - 賃金 ＝ 12ドル×勤務時間

```
(define (wage h) (* 12 h))
```

- 全従業員の勤務時間のリスト alon から賃金のリストを生成する関数 hours->wages を定義し，実行する
  - このとき上記の関数 wage を使う

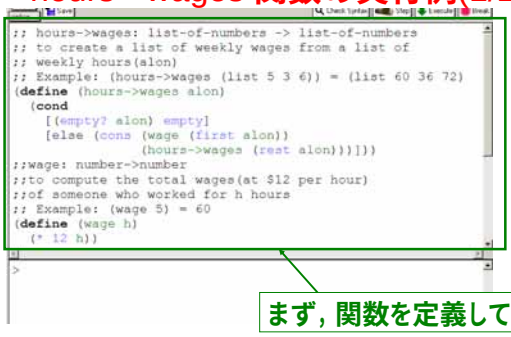
(list 5 3 6) → hours->wages → (list 60 36 72)

入力は勤務時間(数値)のリスト

出力は賃金(数値)のリスト

Literacy

hours->wages 関数の実行例(1/2)

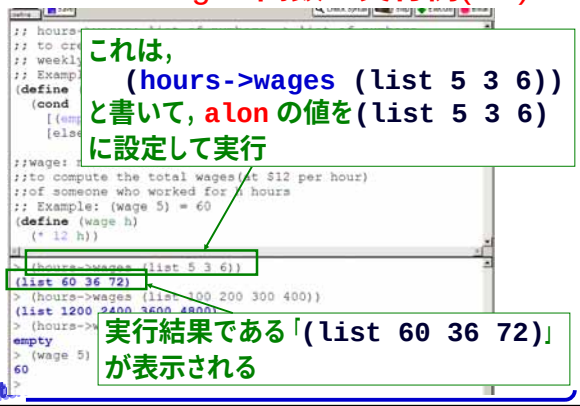


```
;; hours->wages: list-of-numbers -> list-of-numbers
;; to create a list of weekly wages from a list of
;; weekly hours(alon)
;; Example: (hours->wages (list 5 3 6)) = (list 60 36 72)
(define (hours->wages alon)
  (cond
    [(empty? alon) empty]
    [else (cons (wage (first alon))
                (hours->wages (rest alon)))]))
;;wage: number->number
;;to compute the total wages(at $12 per hour)
;;of someone who worked for h hours
;; Example: (wage 5) = 60
(define (wage h)
  (* 12 h))
```

まず，関数を定義している

Literacy

hours->wages 関数の実行例(2/2)



これは，
(hours->wages (list 5 3 6))
と書いて，alon の値を(list 5 3 6)
に設定して実行

実行結果である「(list 60 36 72)」が表示される

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)) …]))
```

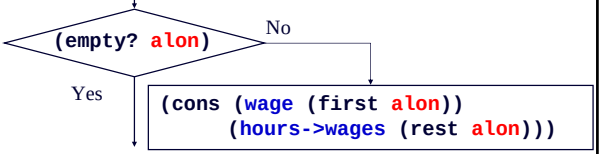
今回の関数 hours->wages のテンプレート

```
(define (hours->wages alon)
  (cond
    [(empty? alon) …]
    [else … (first alon) …
          … (hours->wages (rest alon)) …]))
```

Literacy

例題2．勤務時間リストから賃金リストを生成

1. リストが空ならば： (終了条件)
   解は empty (自明な解)
2. そうでなければ：
   – 「リストの rest 部」の賃金リストを求め，それと，(wage (first alon)) を結合したものが求める解



empty が自明な解

Literacy

例題2．勤務時間リストから賃金リストを生成

```
;; hours->wages: list-of-numbers -> list-of-numbers
;; to create a list of weekly wages from a list of
;; weekly hours(alon)
;; Example: (hours->wages (list 5 3 6)) = (list 60 36 72)
(define (hours->wages alon)
  (cond      終了条件   自明な解
    [(empty? alon) empty]
    [else (cons (wage (first alon))
                (hours->wages (rest alon)))]))
```

hours->wagesの内部に hours->wages
(再帰による繰り返し)
例：(hours->wages (list 5 3 6))
= (cons 60 (hours->wages (list 3 6))

Literacy

(hours->wages (list 5 3 6)) から (list 60 36 72) が得られる過程の概略

```
(hours->wages (list 5 3 6))   最初の式
= ...
= (cons 60 (hours->wages (list 3 6)))
= ...
= (cons 60 (cons 36 (hours->wages (list 6))))
= ...
= (cons 60 (cons 36 (cons 72 (hours->wages empty))))
= ...
= (cons 60 (cons 36 (cons 72 empty)))
= (list 60 36 72)
```

実行結果　　コンピュータ内部での計算

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

実習1．2次方程式

- 2次方程式 $ax^2 + bx + c = 0$ の解を求める関数 `quadratic-roots` を定義し，実行する
  - 2つの解を「リスト」として出力する
  - 重解を求める
  - 但し，虚数解は考えない
  - a = 0 の場合も考えない

入力は
3つの数値

`quadratic-roots`

出力は
・1つのリスト
・1つの数値
・シンボル 'None
のどれか

「実習1．2次方程式」の手順

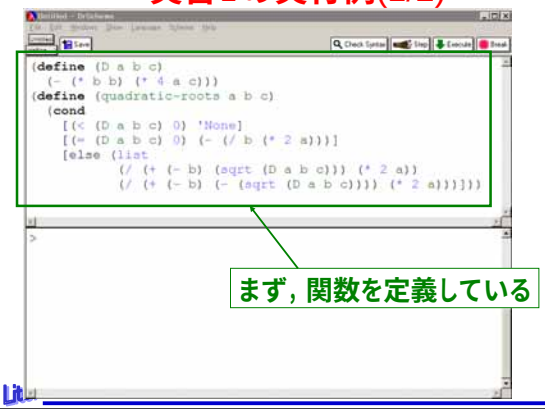
1. 次を「定義用ウインドウ」で，実行しなさい
   - 入力した後に，Run ボタンを押す

```
(define (D a b c)
  (- (* b b) (* 4 a c)))
(define (quadratic-roots a b c)
  (cond
    [(< (D a b c) 0) 'None]
    [(= (D a b c) 0) (- (/ b (* 2 a)))]
    [else (list
           (/ (+ (- b) (sqrt (D a b c))) (* 2 a))
           (/ (+ (- b) (- (sqrt (D a b c)))) (* 2 a)))]))
```

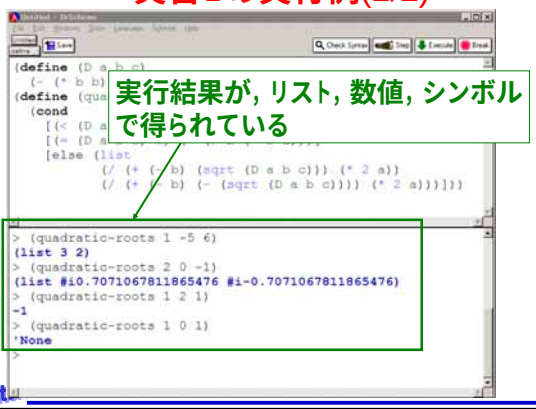
2. その後，次を「実行用ウインドウ」で実行しなさい

```
(quadratic-roots 1 -5 6)
(quadratic-roots 2 0 -1)
(quadratic-roots 1 2 1)
(quadratic-roots 1 0 1)
```

実習1の実行例(1/2)



実習1の実行例(2/2)



二次方程式の解法(1/2)

- 変数 x の二次方程式の一般形:
  $ax^2 + bx + c = 0$
- 二次方程式の解の数:
  係数 a，b，cの値に依存
  (1) $a = 0$ ⇒ 方程式は degenerate
  (2) $a \neq 0$ ⇒ proper な二次方程式
  1. もし $b^2 > 4ac$ なら　二つの解
  2. もし $b^2 = 4ac$ なら　一つの解
  3. もし $b^2 < 4ac$ なら　解無し

二次方程式の解法(2/2)

判別式 $D = b^2 - 4ac$ とする

1) $D > 0$ のとき

$$x = \frac{-b+\sqrt{D}}{2a}, \frac{-b-\sqrt{D}}{2a}$$ 異なる2実数解

2) $D = 0$ のとき

$$x = -\frac{b}{2a},$$ 重解（解の個数は1）

3) $D < 0$ のとき

解なし

Literacy

quadraric-roots 関数

```
(define (D a b c)
  (- (* b b) (* 4 a c)))

(define (quadratic-roots a b c)
  (cond
    [(< (D a b c) 0) 'None]
    [(= (D a b c) 0) (- (/ b (* 2 a)))]
    [else (list
           (/ (+ (- b) (sqrt (D a b c))) (* 2 a))
           (/ (+ (- b) (- (sqrt (D a b c)))) (* 2 a)))]))
```

解なし　重解（解の個数は1）　異なる2実数解

Literacy

実習2．リストの生成

- n 個のシンボル「'*」を要素とするリストを生成する関数 `astlist` を定義し，実行する
  - cons を使用する

入力は数値　　出力はリスト

3　　(list '* '* '*)

Literacy

「実習2．リストの生成」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

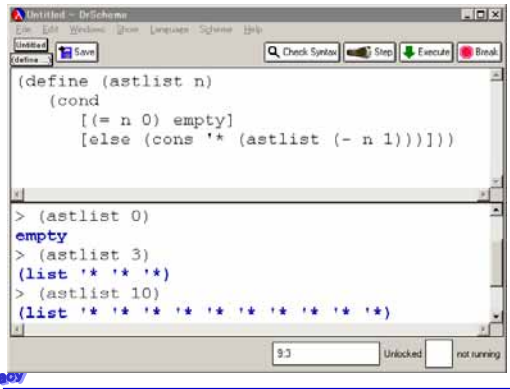
```
;; astlist: number -> list of symbols
;; to create a list of n copies of '*
;; (astlist 0) = empty
;; (astlist 3) = (list '* '* '*)
(define (astlist n)
   (cond
      [(= n 0) empty]
      [else (cons '* (astlist (- n 1)))]))
```

2. その後，次を「実行用ウインドウ」で実行しなさい

```
(astlist 0)
(astlist 3)
(astlist 10)
```

Literacy

実行結果の例



Literacy

リストの生成

1. 入力 n = 0 ならば：　→ 終了条件

   empty　　→ 自明な解

2. そうで無ければ：
   - 長さ n-1 のリストを定義し，その先頭に「'*」をつなげる
   - リストの先頭に「'*」をつなげるために，cons を使う

Literacy

astlist 関数

```
;; astlist: number -> list of symbols
;; to create a list of n copies of '*
;; (astlist 0) = empty
;; (astlist 3) = (list '* '* '*)
(define (astlist n)
   (cond 終了条件  自明な解
      [(= n 0) empty]
      [else (cons '* (astlist (- n 1)))]))
```

astlist の内部に astlist (再帰による繰り返し)
例: (astlist 3)
= (cons '* (astlist 2)

Literacy

(astlist 3) から (list '* '* '*) が得られる過程の概略

```
(astlist 3)  最初の式
= ...
= (cons '* (astlist 2))
= ...
= (cons '* (cons '* (astlist 1)))
= ...
= (cons '* (cons '* (cons '* (astlist 0))))
= ...
= (cons '* (cons '* (cons '* empty)))
= (list '* '* '*)                    コンピュータ内部での計算
```

実行結果

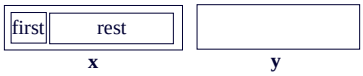
Literacy

実習3．リストの連結

2つのリスト(x と y)を連結する関数 my-append を定義し，実行する．

- x が空ならば: y
- x が空でなければ:

  「x の first 部」と「x の rest と y とを連結したもの(これは，x の rest 部を使った再帰呼出の結果)」を cons で結合



Literacy

「実習3．リストの連結」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

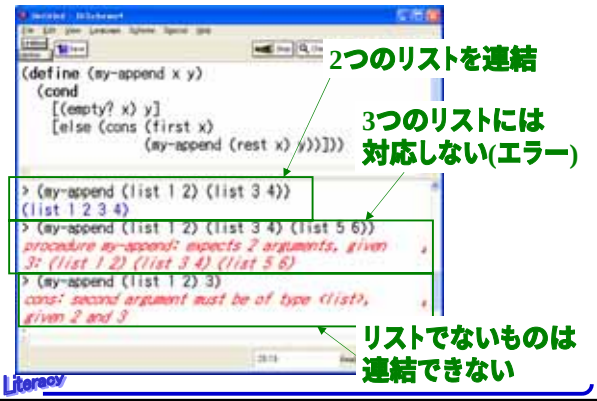
```
(define (my-append x y)
  (cond
    [(empty? x) y]
    [else (cons (first x)
                (my-append (rest x) y))]))
```

2. その後，次を「実行用ウインドウ」で実行しなさい

```
(my-append (list 1 2) (list 3 4))
```

Literacy

リストの連結の実行例



2つのリストを連結

3つのリストには対応しない(エラー)

リストでないものは連結できない

Literacy

my-append 関数

```
(define (my-append x y)
  (cond   終了条件  自明な解
    [(empty? x) y]
    [else (cons (first x)
                (my-append (rest x) y))]))
```

my-append の内部に my-append (再帰による繰り返し)
例: (my-append (list 1 2) (list 3 4))
=(cons 1
(my-append (list 2) (list 3 4))

Literacy

実習4．かけ算の表

- 2つの数 m と n から，m 行n 列のかけ算の表を出力する関数 hyou を定義し，実行する
  - かけ算の表の「1行分」を出力する gyou も定義し，実行する
  - かけ算の表は，リストのリストの形で得る

入力は，2つの数値

```
3 4
```

⇒ hyou ⇒ 出力は リストのリスト

```
(list
 (list 12 9 6 3)
 (list 8 6 4 2)
 (list 4 3 2 1))
```

Literacy

「実習4．かけ算の表」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

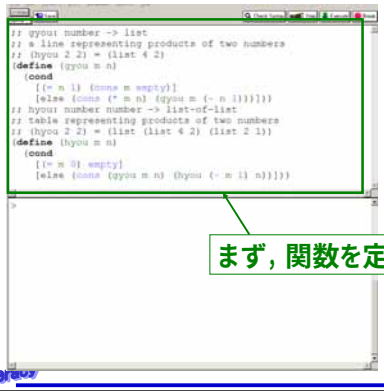
```
;; gyou: number -> list
;; a line representing products of two numbers
;; (gyou 3 4) = (list 12 9 6 3)
(define (gyou m n)
    (cond
        [(= n 1) (cons m empty)]
        [else (cons (* m n) (gyou m (- n 1)))]))
;; hyou: number number -> list-of-list
;; table representing products of two numbers
;; (hyou 2 2) = (list (list 4 2) (list 2 1))
(define (hyou m n)
    (cond
        [(= m 0) empty]
        [else (cons (gyou m n) (hyou (- m 1) n))]))
```

2. その後，次を「実行用ウインドウ」で実行しなさい

```
(hyou 3 4)
```

Literacy

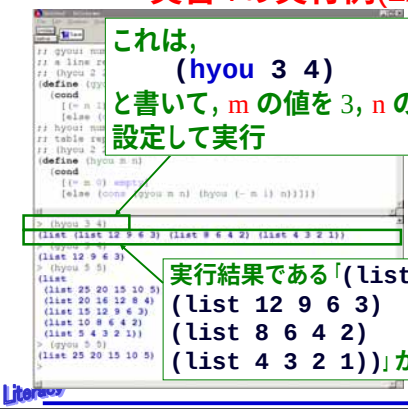
実習4の実行例(1/2)



まず，関数を定義している

Literacy

実習4の実行例(2/2)



これは，
(hyou 3 4)
と書いて，m の値を 3，n の値を 4 に設定して実行

実行結果である「(list
(list 12 9 6 3)
(list 8 6 4 2)
(list 4 3 2 1))」が表示される

Literacy

hyou 関数と gyou 関数の関係

- **hyou 関数**
  - 1つの表を定義し，実行する
    例） (list (list 12 9 6 3)
    (list 8 6 4 2)
    (list 4 3 2 1))
  - gyou 関数を使用
- **gyou 関数**
  - 1行分を定義し，実行する
    例） (list 12 9 6 3)

Literacy

hyou 関数

```
;; hyou: number number -> list-of-list
;; table representing products of two numbers
;; (hyou 2 2) = (list (list 4 2) (list 2 1))
(define (hyou m n)
  (cond 終了条件　自明な解
    [(= m 0) empty]
    [else (cons (gyou m n) (hyou (- m 1) n))]))
```

hyou の内部に hyou (再帰による繰り返し)

```
例:  (hyou 2 2)
     = (cons (gyou 2 2)
             (hyou (- 2 1) 2))
```

Literacy

gyou 関数

```
;; gyou: number -> list
;; a line representing products of two numbers
;; (hyou 3 4) = (list 12 9 6 3)
(define (gyou m n)
    (cond   終了条件      自明な解
        [(= n 1) (cons m empty)]
        [else (cons (* m n) (gyou m (- n 1)))]))
```

gyou の内部に gyou (再帰による繰り返し)
例: (gyou 3 4)
= (cons 12 (gyou 3 3))

Literacy

実習4b. ステップ実行

- 関数 hyou (実習4)について，実行結果に至る過程を見る
  - (gyou 3 4) から (list 12 6 9 3) に至る過程と，(hyou 5 5) から実行結果に至る過程を見る
  - DrScheme のステップ実行機能を使用する

Literacy

「実習4b. ステップ実行」の手順 (1/2)

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

実習4と同じ

```
;; gyou: number -> list
;; a line representing products of two numbers
;; (gyou 3 4) = (list 12 9 6 3)
(define (gyou m n)
    (cond
        [(= n 1) (cons m empty)]
        [else (cons (* m n) (gyou m (- n 1)))]))
;; hyou: number number -> list-of-list
;; table representing products of two numbers
;; (hyou 2 2) = (list (list 4 2) (list 2 1))
(define (hyou m n)
    (cond
        [(= m 0) empty]
        [else (cons (gyou m n) (hyou (- m 1) n))]))
(gyou 3 4)
```

2. DrScheme を使って，ステップ実行の様子を確認しなさい (Step ボタン，Next ボタンを使用)
   - 理解しながら進むこと

☆ 次ページも行ってください

Literacy

「実習4b. ステップ実行」の手順 (2/2)

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

実習4と同じ

```
;; gyou: number -> list
;; a line representing products of two numbers
;; (gyou 3 4) = (list 12 9 6 3)
(define (gyou m n)
    (cond
        [(= n 1) (cons m empty)]
        [else (cons (* m n) (gyou m (- n 1)))]))
;; hyou: number number -> list-of-list
;; table representing products of two numbers
;; (hyou 2 2) = (list (list 4 2) (list 2 1))
(define (hyou m n)
    (cond
        [(= m 0) empty]
        [else (cons (gyou m n) (hyou (- m 1) n))]))
(hyou 5 5)
```

2. DrScheme を使って，ステップ実行の様子を確認しなさい (Step ボタン，Next ボタンを使用)
   - 理解しながら進むこと

Literacy

(gyou 3 4)から (list 12 9 6 3) が得られる過程の概略

(gyou 3 4) 最初の式

```
= ...
= (cons 12 (gyou 3 3))
= ...
= (cons 12 (cons 9 (gyou 3 2)))
= ...
= (cons 12 (cons 9 (cons 6 (gyou 3 1))))
= ...
= (cons 12 (cons 9 (cons 6 (cons 3 empty))))
```

= (list 12 9 6 3) 実行結果

コンピュータ内部での計算

Literacy

(gyou 3 4)から (cons 12 (gyou 3 3))が得られる過程

```
(gyou 3 4)
= ...
= (cons 12 (gyou 3 3))
= ...
= (cons
= ...
= (cons
= ...
= (cons
= (list
```

```
(gyou 3 4)
= (cond
        [(= 4 1) (cons 3 empty)]
        [else (cons (* 3 4) (gyou 3 (- 4 1)))])
= (cond
        [false (cons 3 empty)]
        [else (cons (* 3 4) (gyou 3 (- 4 1)))])
= (cons (* 3 4) (gyou 3 (- 4 1)))
= (cons 12 (gyou 3 (- 4 1)))
= (cons 12 (gyou 3 3))
```

Literacy

(hyou 5 5)から結果が得られる過程の概略

(hyou 5 5) 最初の式

コンピュータ内部での計算

```
= ...
= (cons (list 25 20 15 10 5) (hyou 4 5))
= ...
= (cons (list 25 20 15 10 5) (cons (list 20 16 12 8 4) (hyou 3 5)))
= ...
= (cons (list 25 20 15 10 5) (cons (list 20 16 12 8 4) (cons (list 15
  12 9 6 3) (hyou 2 5))))
= ...
= (cons (list 25 20 15 10 5) (cons (list 20 16 12 8 4) (cons (list 15
  12 9 6 3) (cons (list 10 8 6 4 2) (hyou 1 5)))))
= ...
= (cons (list 25 20 15 10 5) (cons (list 20 16 12 8 4) (cons (list 15
  12 9 6 3) (cons (list 10 8 6 4 2) (cons (list 5 4 3 2 1) (hyou 0
  5))))))
= ...
= (cons (list 25 20 15 10 5) (cons (list 20 16 12 8 4) (cons (list 15
  12 9 6 3) (cons (list 10 8 6 4 2) (cons (list 5 4 3 2 1) empty)))))
```

```
= (list (list 25 20 15 10 5) (list 20 16 12 8 4) (list 15 12 9 6 3)
  (list 10 8 6 4 2) (list 5 4 3 2 1))
```

実行結果

課題

課題1

- 関数 `1list` についての問題
  - `(1list 2)` から `(list 1 1)` が得られる過程の概略を数行程度で説明しなさい

```
;; 1list: number -> list-of-numbers
;; to create a list of n copies of 1
;; (1list 0) = empty
;; (1list 3) = (list 1 1 1)
(define (1list n)
   (cond
      [(= n 0) empty]
      [else (cons 1 (1list (- n 1)))]))
```

課題2

- 次の関数 `insert` について，「`(insert 4 (list 5 1))`」から「`(list 5 4 1)`」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利用しなさい

```
(define (insert n alon)
  (cond
    [(empty? alon) (cons n empty)]
    [else (cond
            [(<= (first alon) n) (cons n alon)]
            [(> (first alon) n)
             (cons (first alon)
                   (insert n (rest alon)))])]))
```

課題3

- 次の関数 `relative-to-absolute` について，「`(relative-to-absolute (list 1 2 3))`」から「`(list 1 3 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利用しなさい

```
;; relative-to-absolute : list-of-numbers -> list-of-numbers
(define (relative-to-absolute alon)
  (cond
    [(empty? alon) empty]
    [else (cons (first alon)
                (add-to-each (first alon)
                             (relative-to-absolute (rest alon))))]))
;
;; add-to-each : number (listof number) -> (listof number)
(define (add-to-each n alon)
  (cond
    [(empty? alon) empty]
    [else (cons (+ (first alon) n) (add-to-each n (rest alon)))]))
```

課題4．2次方程式の解

- 関数 `quadratic-roots`（実習1）についての問題
  - 係数 a = 0 のときにも正しく計算ができるように書き直しを行い，書き直した結果の関数と，実行結果を報告しなさい

a = 0 かつ b = 0 かつ c = 0 のとき
すべての x が解である
a = 0 かつ b = 0 かつ c ≠0 のとき
解なし
a = 0 かつ b ≠ 0 のとき
x = - c / b

課題5

- 数値 n から，「1番目から n 番目の奇数のリスト」を作る関数 series-odd-list を作成し，実行結果を報告しなさい
  - 例：(series-odd-list 4) = (7 5 3 1)
  - ヒント： 次の空欄を埋めなさい

```
(define (series-odd-list n)
  (cond
    [                          ]
    [                          ]))
```

Literacy

課題6

- 構造体 AddressNote に関する問題
  - AddressNote 構造体のリスト a-list と年齢 an-age から，年齢が an-age 以下である人のリストを出力する関数 selection-by-age を定義し，実行結果を報告しなさい

AddressNote

name
age
address

例えば

```
(selection-by-age
   (list (make-address-note “Kunihiko Kaneko” 35 “Hakozaki”)
         (make-address-note “Taro Tanaka” 36 “Kaizuka”)
         (make-address-note “Hanako Saito” 33 “Tenjin”)) 35)

 = (list (make-address-note “Kunihiko Kaneko” 35 “Hakozaki”)
         (make-address-note “Hanako Saito” 33 “Tenjin”))
```

Literacy

課題7．1週間分のカレンダー

- ある年 y のある月 m のある日 d の曜日 youbi から，その「1週間分のリスト」を出力する関数を定義し，実行結果を報告しなさい
  - youbi は 0 ～ 6 までの数値で，0なら日曜，1なら月曜…
  - 参考：youbi は y 年 m 月 d 日から計算することもできる (ツエラーの公式，「条件式と条件関数」の回を参照)
  - 「1週間分のリスト」とは，日曜日から土曜日までの日付のリストで，数値あるいは「'*」で表される

例えば

y = 2005, m = 6, d = 4, youbi = 6 のとき
⇒ 「(list '* '* '* 1 2 3 4)」が出力される

y = 2005, m = 5, d = 31, youbi = 2 のとき
⇒ 「(list 29 30 31 '* '* '* '*)」が出力される

Literacy

課題8．1ヶ月分のカレンダー

- ある年 y のある月 m から，その「月のカレンダー」を出力する関数を定義し，実行結果を報告しなさい
  - 「月のカレンダー」は，リストのリスト
  - 1週間で1つのリストとなり，5週間あれば，5つのリストからなる大きな1つのリスト

例えば

```
(list
  (list '* '* '* '* 1 2 3)
  (list 4 5 6 7 8 9 10)
  (list 11 12 13 14 15 16 17)
  (list 18 19 20 21 22 23 24)
  (list 25 26 27 28 29 30 31))
```

Literacy

課題9．1年分のカレンダー

- ある年 y から，その「年のカレンダー」を出力する関数を定義し，実行結果を報告しなさい
  - 「年のカレンダー」は，リストのリストのリスト

Literacy