工学部専門科目「プログラミング言語」2012 年度 期末試験

実施日時: 2012 年 8 月 1 日 8:45〜10:15，担当: 五十嵐 (淳)

注意: 穴を埋めて手続きの定義を完成させる問題については，穴の部分だけでなく手続きの定義全体を解答せよ．

問 1(解答用紙 1 枚目表に解答) メタ・サーキュラー・インタプリタで `let*` を derived expression として実装することを考える．以下は手続き `eval` の一部である．

```
(define (eval exp env)
  (cond ...
        ((let*? exp) (eval (let*->nested-lets exp) env))
         ...))
```

ここで使われている，(引用された)`let*` 式を引数として，それと同等な入れ子になった `let` 式を返す手続き `let*->nested-lets` を与えよ．

問 2(解答用紙 1 枚目裏に解答) 遅延評価 Scheme を考える．手続き呼び出しにおいて各引数の評価を遅延させるかどうかを，手続きの構文を拡張することで遅延評価 Scheme のプログラマが指定できるようにする．具体的な構文拡張は，手続のパラメータを `(lazy ...)` で囲むことで，そのパラメータに対応する実引数の評価が遅延されることを示すこととする．例えば

`(define (f a (lazy b))` 〈 本体 〉`)`

であれば，`(f` 〈式 1〉〈式 2〉`)` という呼び出しで，〈式 1〉のみが手続きの 〈本体〉 の評価の前に評価され，〈式 2〉は値が必要になった時に評価される．

2-1) 拡張された `lambda` 式 (の引用) を受け取り，パラメータ名のリストを返す手続き `procedure-parameters` と，各パラメータに `lazy` がついているかどうかの真偽値のリストを返す手続き `procedure-param-spec` を以下の ?? を埋めて定義せよ．

```
(define (procedure-parameters p) ??)
(define (procedure-param-spec p) ??)

;; 実行例
> (procedure-parameters '(lambda (a (lazy b)) (+ a b)))
(a b)
> (procedure-param-spec '(lambda (a (lazy b)) (+ a b)))
(#f #t)
```

2-2) 図 1 に，インタプリタの手続き `eval` と `my-apply` の一部を示す．`my-apply` から呼ばれている手続き `list-of-arg-possibly-delayed-args` の定義を与えよ．この時，手続き `no-operands?`, `first-operand`, `rest-operands`, `actual-value`, `delay-it` は定義せずに使ってよい．

```
(define (eval exp env)
  (cond ...
        ((application? exp)
         (my-apply (actual-value (operator exp) env) (operands exp) env))
        ...))

(define (my-apply procedure arguments env)
  (cond ...
        ((compound-procedure? procedure)
         (eval-sequence
           (procedure-body procedure)
           (extend-environment
             (procedure-parameters procedure)
             (list-of-possibly-delayed-args (procedure-param-spec procedure) arguments env)
             (procedure-environment procedure))))
        ...))
```

図 1: 問 2 のプログラム

問 3(解答用紙 2 枚目表に解答) catch/throw 機構を持つ拡張 Scheme を考える．以下のふたつの命題が正しいかそれぞれ判断し，正しければその理由を説明し，正しくなければ反例を挙げ反例である理由の説明をせよ．

1. 任意の拡張 Scheme 式 e について，(catch 'a (catch 'b e)) と (catch 'b (catch 'a e)) の実行結果は等しい
2. 任意の拡張 Scheme 式 e_1, e_2, e_3 について，(catch e_1 (catch e_2 e_3)) と (catch e_2 (catch e_1 e_3)) の実行結果は等しい

問 4(解答用紙 2 枚目表に解答) コインのリスト (非負整数リスト) coins と整数で与えられる合計金額 total から，以下の条件 1,2 を満たすコインのリスト (非負整数リスト) answer を**非決定的に返す**手続き change を，amb を使って下の ?? を埋めて完成させよ．

```
(define (change coins amount)
  (if (= 0 amount) '() ??))
```

条件 1 answer の各要素は coins の要素である．

条件 2 answer の要素の合計は total に等しい．