

# プログラミング言語特論(2008年度)・テスト問題用紙

(09年7月29日(木)・16:20～17:50)

## 解答上、その他の注意事項

- I. 問題は、問 I～V までである。
- II. 解答用紙の右上の欄に学籍番号・名前を記入すること。
- III. ノート・プリント・参考書などは持ち込み可である。
- IV. 携帯電話などの通信機能を持つものは 持ち込み不可 である。
- V. 問 I を解答するときのみ、ノート PC を使用して良い。ネットワークに接続して WWW を閲覧しても良いが、掲示板、チャット、メールなどで生身の人間と通信することは禁じる。
- VI. テストの配点は 50 点 (+ ボーナス 20 点) である。合格はレポートの得点を加えて、100 点満点中 60 点以上とする。

- (1) 引数として与えられる整数のリスト中の 3 の倍数の要素の和を返す関数:

```
foo :: [Integer] -> Integer
```

を定義せよ。

例えば、foo [1,2,3,4,5,6] は 9、foo [4,6,-1,0,-4,-3] は 3 となる。

この問では map, filter, foldl, foldr などのリストに関するライブラリ関数や内包表記を使わず、if ~ then ~ else ~ 式や論理演算子、等号・不等号、パターンマッチ、再帰などを使って定義せよ。

ヒント:

- 剰余を求める ( C の % 演算子に相当する ) Haskell の演算子は 'mod' である。

```
Prelude> 7 'mod' 3
1
Prelude> (-4) 'mod' 3
2
```

- (2) 正の整数  $n$  を引数として受け取り、 $2 \leq i < j \leq n$  を満たす整数の組  $(i, j)$  で、互いに素となる (つまり、 $i$  と  $j$  の最大公約数が 1 となる) 組を列挙する関数:

```
bar :: Integer -> [(Integer,Integer)]
```

を ( リストの内包表記を用いて ) 定義せよ。

例えば、bar 3 は [(2,3)]、bar 6 は [(2,3),(2,5),(3,4),(3,5),(4,5),(5,6)] となる。

( リストの要素の順番はこのとおりでなくても良い。 )

ヒント:

- $m$  から  $n$  までの整数のリストは  $[m..n]$  という式で得られる。

```
Prelude> [1..9]
[1,2,3,4,5,6,7,8,9]
```

- 2 つの引数の最大公約数を求める関数 gcd は Haskell の標準ライブラリに用意されているので、利用して良い。

```
Prelude> gcd 12 18
6
Prelude> gcd 11 30
1
```

II. (ラムダ計算) (6点×2)

次のλ式が正規形に到達するまでの、最左変換による1ステップずつのβ変換の列を書け。ただし、10回以内の最左変換で正規形に到達しない式については、それが判別できる時点(以前と同じ式が出現した時点) または10回最左変換した時点で止めてよい。

解答例 1:

$$\begin{aligned} & (\lambda f x. f(fx))((\lambda f x. f(fx))g)y \\ \xrightarrow{\beta} & (\lambda x. ((\lambda f x. f(fx))g)((\lambda f x. f(fx))g)x)y \\ \xrightarrow{\beta} & ((\lambda f x. f(fx))g)((\lambda f x. f(fx))g)y \\ \xrightarrow{\beta} & (\lambda x. g(gx))((\lambda f x. f(fx))g)y \\ \xrightarrow{\beta} & g(g((\lambda f x. f(fx))g)y)) \\ \xrightarrow{\beta} & g(g((\lambda x. g(gx))y)) \\ \xrightarrow{\beta} & g(g(g(y))) \end{aligned}$$

解答例 2:

$$\begin{aligned} & (\lambda x. xx)(\lambda x. xx) \\ \xrightarrow{\beta} & (\lambda x. xx)(\lambda x. xx) \\ \xrightarrow{\beta} & (\text{停止しない}) \end{aligned}$$

- (1)  $(\lambda f g x. f x(gx))(\lambda xy. x)(\lambda xz. z)$   
 (2)  $(\lambda x. f x)((\lambda p. p(\lambda xy. x))((\lambda xyf. fxy)xx))$

なお、必要に応じて  $I \equiv \lambda x. x$  など適宜、定数を定義しても良い。

III. (Haskell)

(7点×2)

次の例にならって、下のHaskellのプログラム(1)~(2)を評価した結果を書け。

例: `take 5 (from 1)` ⇒ 評価結果: `[1,2,3,4,5]`

ただし、`take` と `from` は講義プリントに定義されているとおりの関数である。

```
from :: Integer -> [Integer]
from n = n : from (n+1)

take :: Integer -> [a] -> [a]
take 0 _ = []
take _ [] = []
take n (x:xs) = x : take (n-1) xs
```

- (1) `foldr (\ x y -> 2 * y - x) 0 [1,3,5]`

この問で使用されている関数 `foldr` の定義は次のとおりである。

```
foldr :: (a -> b -> b) -> b -> [a] -> b
foldr f z [] = z
foldr f z (x:xs) = f x (foldr f z xs)
```

- (2) `[ (x,y) | x <- [1,2,3], y <- [2,3,4], x/=y ]`

(この問に関してはリスト内の順番のみの間違いは、減点はしない。)

なお、`/=` は “not equal” である。(C言語の `!=` 演算子に相当する。)

IV. ( 語句 )

( 6 点 × 2 )

プログラミング言語 ( やその処理系 ) で用いられる次の 6 つの語句のうち 2 つを選択し、具体的な例を挙げて説明せよ。ただし、講義プリントにのっている例ではなく、オリジナルの例を考えること。

- 動的束縛 ( dynamic binding )
- 遅延評価 ( lazy evaluation )
- 非決定性 ( nondeterminism )
- 接続 ( continuation )
- CPS ( continuation passing style )
- 多重定義 ( overloading )

V. ( 自由記述 — ボーナス問題 )

( 最高 20 点 )

プログラミング言語の構文 ( 文法 ) の理解を補助するためには、エディタでのキーワードの色分けや、構文木の表示などいくつかの手法が実用的に使われている。

では、プログラミング言語の ( 構文ではなく ) 意味の理解を補助するために、どのような手法やツールがあれば良いと思うか?。( 実現性は気にせず ) 自由に、できるだけ具体的に考述せよ。







