プログラミング言語特論（2010年度）・テスト問題用紙

(2011年2月23日（水）・ 13:00 〜 14:30)

解答上、その他の注意事項

I. 問題は、問I〜Vまである。

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点して、100点満点中60点以上とする。

I. （Haskell 実習問題） （6 点 × 2）

(1) 引数として与えられる整数のリスト中の要素がすべて 3 で割って 1 余る数かどうかを判定する関数

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

を定義せよ。

例えば、`foo [1, 4, 7]`は `True` であり、`foo [1, 5]`は `False` である。

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、パターンマッチング、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) 2 つの整数リスト `xs`, `ys` を引数として受け取り、$i \in xs$, $j \in ys$ を満たす整数の組 (i, j) で、積 $i \times j$ が 6 の倍数となるものを列挙する関数:

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

を（リストの内包表記を用いて）定義せよ。

例えば、`bar [1,2,3] [3,4,5,6]` は `[(1,6),(2,3),(2,6),(3,4),(3,6)]` で、
`bar [1,3,5] [2,4,6]` は `[(1,6),(3,2),(3,4),(3,6),(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。）

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:

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

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) $(\lambda xy.xyy)(\lambda xy.y)(\lambda xy.x)$

(2) $(\lambda mnf.m(nf))(\lambda sz.s(sz))(\lambda sz.z)$

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

この問で使用されている関数 `foldl` の定義は次のとおりである。

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

(2) `[ (x,y) | x <- [1,3,5], 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。）

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。

- 高階関数（higher-order function）
- 遅延評価（lazy evaluation）
- カプセル化（encapsulation）
- 非決定性（nondeterminism）
- 接続（continuation）
- 多相（polymorphism）

V. （自由記述 — ボーナス問題） （最高 20 点）

「楽しいプログラミング」とは何か？

プログラミングの学習を楽しくするアイデアを、自由に、できるだけ具体的に考述せよ。

プログラミング言語**特**論（2010年度）・テスト解答用紙('11年2月23日)

学籍番号		氏名	

学籍番号		氏名	