プログラミング言語**特**論（2008年度）・テスト問題用紙

('08年7月24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（偶数の要素は無視し）奇数の要素の積を返す関数:

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

を定義せよ。

例えば、`foo [1,2,3,4,5]` は 15、`foo [4,2,-1,0,-4,-3]` は 3 となる。

この問では `map`, `filter`, `foldl`, `foldr` などの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、不等号、パターンマッチ、再帰などのみを使って定義せよ。

(2) 正の整数 `n` を引数として受け取り、$0 \leq i \leq n$ かつ $0 \leq j \leq n$ を満たす整数の組 (i, j) で、和が奇数となる（つまり、$odd\ (i + j)$ が成り立つ）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 2` は `[(0,1),(1,0),(1,2),(2,1)]`、`bar 4` は `[(0,1),(0,3),(1,0),(1,2),(1,4),(2,1),(2,3),(3,0),(3,2),(3,4),(4,1),(4,3)]` 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: 0 から n までの整数のリストは `[0..n]` という式で得られる。

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、正規形が存在しない式については、それが判別できる時点（ただし少なくとも 3 回以上 β 変換したあと）に「停止しない」と記入せよ。

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:

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

(1) $(\lambda xyz.xz(yz))((\lambda xy.x)(\lambda xyz.xz(yz)))(\lambda xy.x)$

(2) $(\lambda xyz.xz(yz))(\lambda xy.x)(\lambda xy.x)$

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) `take 6 (iterate (\ (x,y) -> (y,x+y)) (1,1))`

この問で使用されている関数 `iterate` の定義は次のとおりである。

```
iterate     :: (a -> a) -> a -> [a]
iterate f x = x : iterate f (f x)
```

(2) `[ (x,y) | x <- [1,2,3], y <- [2,4,6], y == x+1]`

（この問に関してはリスト内の順番の間違いの減点は 1 点のみとする。）

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。

- 遅延評価（lazy evaluation）
- カプセル化（encapsulation）
- 例外（exception）
- 接続（continuation）
- CPS（continuation passing style）
- 多相（polymorphism）

V. （自由記述 — ボーナス問題） （最高 20 点）

これまでにいろいろなプログラミング言語を学習して、理解しにくかった点、間違いやすいと感じた点、あるいは初心者がつまづきやすいと感じた点は何か? できるだけ具体的に述べよ。また、それに対して、オリジナルの改良のアイデアや抜本的な解決策を（実現性は気にせず）自由に考述せよ。

プログラミング言語**特**論（2008年度）・テスト解答用紙('08年7月24日)

学籍番号		氏名	

学籍番号		氏名	