プログラミング言語論 2015年度 第2回小テスト

学籍番号:　　　　　　　　　　　　氏名:

問 1 以下のプログラム断片の型の整合性を、手順 (1)-(2) に従って示せ。ただし、これは講義中に提示した、C のサブセットによるプログラムの断片である。

```
int *p;
int x[3];
p = x;
```

(1) 変数宣言部分 `int *p; int x[3];` を、講義中に紹介した型の後置記法による宣言に直せ。

(2) (1) で得られた、p, x の型の後置記法による宣言から、講義中に提示した規則に従って、代入式 `p=x` が型に関して整合性を持つことを示せ。

型付け規則

- 関数、ポインタ、配列に関する規則

$$\frac{e : \tau[n]}{e[i] : \tau} \qquad \frac{e : \tau()}{e() : \tau} \qquad \frac{e : \tau*}{*\,e : \tau} \qquad \frac{e : \tau[n]}{e : \tau\&}$$

- 代入演算子 $=$ に関する規則 (e は左辺値を持つ式であり、かつ定数ではない)

$$\frac{e : \tau \quad e' : \tau}{e = e' : \tau}$$

- &演算子に関する規則 (τ の一番右側（外側）は&ではない)

$$\frac{e : \tau}{\&e : \tau\&} \qquad \frac{e : \tau\&}{*\,e : \tau} \qquad \frac{e : \tau* \quad e' : \tau\&}{e = e' : \tau\&}$$

問 2 ラムダ式 $(\lambda x.\ \lambda y.\ x)\ ((\lambda z.\ z)\ w)$ は何度か β 変換を行うことによってラムダ式 $(\lambda y.\ w)$ に変換できるが、その変換過程を示せ。(変換過程は複数あるが、そのうちの一つでよい。)

ラムダ計算に関する規則

- β 変換

$$(\lambda x.M)\ N \ \xrightarrow[\beta]{} \ M[N/x]$$

$$\frac{M \xrightarrow[\beta]{} N}{\lambda x.M \xrightarrow[\beta]{} \lambda x.N} \qquad \frac{M \xrightarrow[\beta]{} N}{MP \xrightarrow[\beta]{} NP} \qquad \frac{M \xrightarrow[\beta]{} N}{PM \xrightarrow[\beta]{} PN}$$

- 置換

$$\begin{array}{rcl}
c[N/x] & = & c \\
x[N/x] & = & N \\
x[N/y] & = & x \quad (x \neq y) \\
(\lambda y.M)[N/x] & = & \begin{cases} \lambda y.M & \textbf{if } x = y \\ \lambda y.(M[N/x]) & \textbf{if } x \neq y,\ y \notin FV(N) \\ \lambda z.((M[z/y])[N/x]) & \textbf{if } x \neq y,\ z \neq x,\ y \in FV(N), \\ & \quad z \notin FV(M),\ z \notin FV(N) \end{cases} \\
(M_1 M_2)[N/x] & = & (M_1[N/x])(M_2[N/x])
\end{array}$$

- 自由変数

$$\begin{array}{rcl}
FV(c) & = & \{\} \\
FV(x) & = & \{x\} \\
FV(\lambda x.M) & = & FV(M) \setminus \{x\} \\
FV(M_1 M_2) & = & FV(M_1) \cup FV(M_2)
\end{array}$$

問 3　以下の C++言語で書かれたプログラムを実行したときの画面への出力結果を示せ。

```
#include <stdio.h>

class B {
public:
  virtual char f() { return 'B';}
  char g() { return 'B'; }
  char testF(B *b) { return b->f();}
  char testG(B *b) { return b->g();}
};

class D : public B {
public:
  char f() { return 'D';}
  char g() { return 'D';}
};

int main(void) {
  D *d = new D;
  printf("%c%c\n", d->testF(d), d->testG(d));
  return 0;
}
```

問 4　Prolog で以下のように `a`, `b`, `c`, `d`, `e` を宣言した時、`a(X).` という query に対する解（`X` に対する置換）を書け。

```
a(1) :- b.
a(2) :- e.
b :- !, c.
b :- d.
c :- fail.
d.
e.
```

問 5　以下の (1), (2) のプログラムの意味 (状態の変化) を、講義中に提示した規則にしたがって示せ。ただし、これらのプログラムは、講義中で意味の定義を紹介するときに定義した、C の非常に小さなサブセットによるプログラムである。(1)、(2) のプログラムの実行前の状態は、いずれも $\sigma = \{(\mathtt{X}, 3), (\mathtt{Y}, 1), (\mathtt{Z}, 0)\}$ とする。

(1) `Z=(X+4);`

(2) `while(Y){Y=(Y-1);}`

実行の規則

- 式の評価規則
  - 数字列の場合　　$< n, \sigma > \to m$　(m は数字列 n に対応する整数)
  - 変数の場合　　$< x, \sigma > \to \sigma(x)$
  - 足し算の場合
    $$\frac{< a_1, \sigma > \to m_1 \quad < a_2, \sigma > \to m_2}{< (a_1 + a_2), \sigma > \to m} \ (m \text{ は } m_1 \text{ と } m_2 \text{の和})$$
  - 引き算の場合
    $$\frac{< a_1, \sigma > \to m_1 \quad < a_2, \sigma > \to m_2}{< (a_1 - a_2), \sigma > \to m} \ (m \text{ は } m_1 \text{ と } m_2 \text{の差})$$
  - 掛け算の場合
    $$\frac{< a_1, \sigma > \to m_1 \quad < a_2, \sigma > \to m_2}{< (a_1 * a_2), \sigma > \to m} \ (m \text{ は } m_1 \text{ と } m_2 \text{の積})$$
- 文の実行に関する規則
  - 代入文の場合
    $$\frac{< a, \sigma > \to m}{< x = a; , \sigma > \to \sigma[m/x]}$$
    ただし、$\sigma[m/x]$ は以下で定義される。
    $$(\sigma[m/x])(y) = \begin{cases} m & \textbf{if } y = x \\ \sigma(y) & \textbf{if } y \neq x \end{cases}$$
  - 文の並びの場合
    $$\frac{< c_1, \sigma > \to \sigma_1 \quad < c_2, \sigma_1 > \to \sigma_2}{< c_1 \ c_2, \sigma > \to \sigma_2}$$
  - while 文の場合
    $$\frac{< a, \sigma > \to 0}{< \textbf{while } (a) \ \{c\}, \sigma > \to \sigma}$$
    $$\frac{< a, \sigma > \to m \quad < c, \sigma > \to \sigma_1 \quad < \textbf{while } (a) \ \{c\}, \sigma_1 > \to \sigma_2}{< \textbf{while } (a) \ \{c\}, \sigma > \to \sigma_2} \ (\textbf{if } m \neq 0)$$