

Principles of Programming Languages

Small examination 2

Student ID:

Name:

Problem 1 Show the type consistency of the following program fragment, which is written in the subset of C language presented in the lecture, according to (1) and (2).

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

- (1) Rewrite the variable declarations `int *p;` and `int x[3];` in the postfix notation presented in the lecture.

- (2) Show the type consistency of the assignment expression `p=x` by applying the inference rules to the declarations of `p` and `x` in the postfix notation obtained in (1).

Problem 2 A lambda expression $(\lambda x. \lambda y. x) ((\lambda z. z) w)$ can be transformed to $(\lambda y. w)$ by applying the β reductions. Write the each step of the β reductions. (Although there are more than one sequences of β reductions, write one of them.)

Problem 3 Write the output to the display when executing the following program in C++.

```
#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;
}
```

Problem 4

Show the meaning of the following programs (1) and (2) by using the rules presented in the lecture. Note that the programs are in the small subset of C presented in the lecture. Let the states before executing the programs both to be $\sigma = \{(X, 3), (Y, 1), (Z, 0)\}$.

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

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

Rules presented in the lecture Typing rules

- Rules for function calls, pointers, arrays

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

- Rule for assignment operator =, where e is an l-value expression and not a constant.

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

- Rule for the & operator where the outermost part of τ is not &.

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

Rules for lambda calculus

- β reductions

$$\begin{array}{c} (\lambda x.M) N \xrightarrow{\beta} M[N/x] \\ \\ \frac{M \xrightarrow{\beta} N}{\lambda x.M \xrightarrow{\beta} \lambda x.N} \quad \frac{M \xrightarrow{\beta} N}{MP \xrightarrow{\beta} NP} \quad \frac{M \xrightarrow{\beta} N}{PM \xrightarrow{\beta} PN} \end{array}$$

- Substitutions

$$\begin{aligned} c[N/x] &= c \\ x[N/x] &= N \\ x[N/y] &= x \quad (x \neq y) \\ (\lambda y.M)[N/x] &= \begin{cases} \lambda y.M & \text{if } x = y \\ \lambda y.(M[N/x]) & \text{if } x \neq y, y \notin FV(N) \\ \lambda z.((M[z/y])[N/x]) & \text{if } x \neq y, z \neq x, y \in FV(N), \\ & z \notin FV(M), z \notin FV(N) \end{cases} \\ (M_1M_2)[N/x] &= (M_1[N/x])(M_2[N/x]) \end{aligned}$$

- Free variables

$$\begin{aligned} FV(c) &= \{\} \\ FV(x) &= \{x\} \\ FV(\lambda x.M) &= FV(M) \setminus \{x\} \\ FV(M_1M_2) &= FV(M_1) \cup FV(M_2) \end{aligned}$$

Operational semantics for the small subset of C

- Rules for arithmetic expressions

- Sequences of numbers: $\langle n, \sigma \rangle \rightarrow m$ where m is an integer represented by the sequence of numbers n in the decimal representation.

- Variables: $\langle x, \sigma \rangle \rightarrow \sigma(x)$

- Addition:

$$\frac{\langle a_1, \sigma \rangle \rightarrow m_1 \quad \langle a_2, \sigma \rangle \rightarrow m_2}{\langle (a_1 + a_2), \sigma \rangle \rightarrow m} \quad (m \text{ is the sum of } m_1 \text{ and } m_2.)$$

- Subtraction:

$$\frac{\langle a_1, \sigma \rangle \rightarrow m_1 \quad \langle a_2, \sigma \rangle \rightarrow m_2}{\langle (a_1 - a_2), \sigma \rangle \rightarrow m} \quad (m \text{ is the difference of } m_1 \text{ and } m_2.)$$

- Multiplication:

$$\frac{\langle a_1, \sigma \rangle \rightarrow m_1 \quad \langle a_2, \sigma \rangle \rightarrow m_2}{\langle (a_1 * a_2), \sigma \rangle \rightarrow m} \quad (m \text{ is the product of } m_1 \text{ and } m_2.)$$

- Rules for statements

- Assignments:

$$\frac{\langle a, \sigma \rangle \rightarrow m}{\langle x = a; , \sigma \rangle \rightarrow \sigma[m/x]}$$

where $\sigma[m/x]$ is defined as follows.

$$(\sigma[m/x])(y) = \begin{cases} m & \text{if } y = x \\ \sigma(y) & \text{if } y \neq x \end{cases}$$

- Sequences:

$$\frac{\langle c_1, \sigma \rangle \rightarrow \sigma_1 \quad \langle c_2, \sigma_1 \rangle \rightarrow \sigma_2}{\langle c_1 \ c_2, \sigma \rangle \rightarrow \sigma_2}$$

- **while** statements:

$$\frac{\langle a, \sigma \rangle \rightarrow 0 \quad \langle \mathbf{while} (a) \{c\}, \sigma \rangle \rightarrow \sigma}{\langle a, \sigma \rangle \rightarrow m \quad \langle c, \sigma \rangle \rightarrow \sigma_1 \quad \langle \mathbf{while} (a) \{c\}, \sigma_1 \rangle \rightarrow \sigma_2} \quad (\text{if } m \neq 0)$$

$$\langle \mathbf{while} (a) \{c\}, \sigma \rangle \rightarrow \sigma_2$$