Hairong Chen, Jingwen Qian, Xinhao Luo
Homework 7, Programming Languages **December 13, 2020**

Problem 1.

Answer.

$$f(f(f(f\ x)))$$

Problem 2.

Proof. Since Z is $\lambda z.\lambda x.x(z\ z\ x)$, we have

$$\begin{aligned} ZZM &= (\lambda z.\lambda x.x(z\ z\ x))ZM \\ &=_\beta (\lambda x.x(Z\ Z\ x))M \\ &=_\beta M(ZZM) \end{aligned}$$

Problem 3.

- For structural equivalence:

$$(a, b, c, d)$$

- For strict name equivalence:

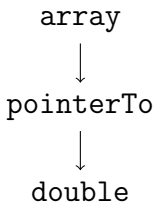
$$\begin{aligned} &(a, b) \\ &(c) \\ &(d) \end{aligned}$$

- For loose name equivalence:

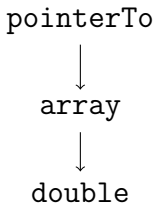
$$\begin{aligned} &(a, b, c) \\ &(d) \end{aligned}$$

Problem 4.

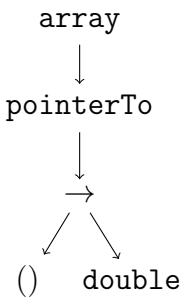
- For `double *x[n];`
  - Meaning: `x` is an array of `n` pointers to doubles.
  - Type tree for `x`:



- For `double (*y)[n];`
  - Meaning: `y` is a pointer to an array of `n` doubles.
  - Type tree for `y`:

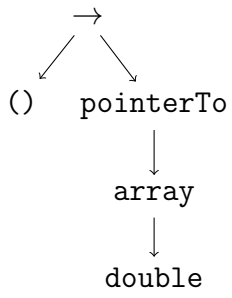


- For `double (*z[n])();`
  - Meaning: `z` is an array of `n` pointers to functions that take void as argument and return double.
  - Type tree for `z`:



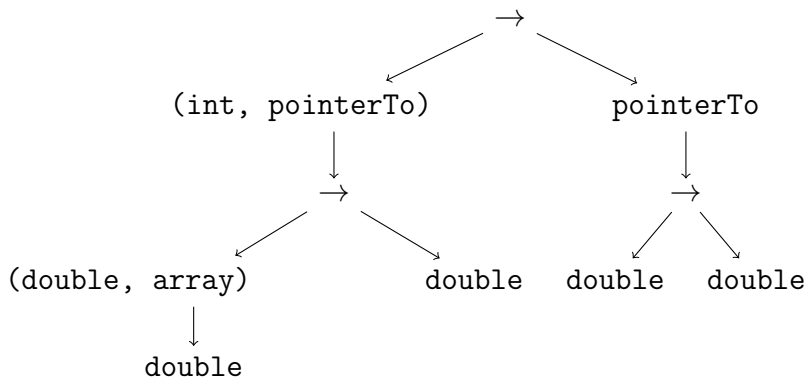
- For `double (*w())[n];`
  - Meaning: `w` is a function that takes void as argument and returns a pointer to an array of `n` doubles.

– Type tree for `w`:



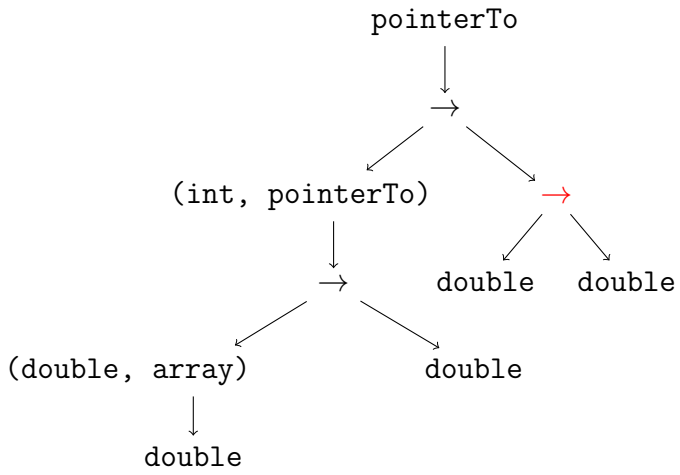
Problem 5.

- For `double (*bar(int, double(*)(double, double[])))(double);`
  - Type of `bar`: a function that takes an int and a pointer to a function, that takes a double and an array of doubles as argument and returns double, as argument and returns a pointer to a function, that takes double as argument and returns double.
  - Type tree for `bar`:



- For `double ((*bar)(int, double(*)(double, double[])))(double);`
  - Type of `bar`: a pointer to a function that takes an int and a pointer to a function, that takes a double and an array of doubles as argument and returns double, as argument and returns a function, that takes double as argument and returns double.

– Type tree for `bar`:



- **Answer.** No, this is not a valid declaration in C.

  Explanation. Since function in C is not a first-class object, it cannot be returned by the other function, which is marked as red in the type tree.

Problem 6.

Answer. Since `int` is 4 bytes, and `char` is 1 byte, totally $4 + 1 = 5$ bytes, and by applying word-aligned structure fields, each entry of `A` would occupy 8 bytes. Thus we have

$$\begin{aligned}\text{address of } \texttt{A[3][7]} &= \text{addr}(\texttt{A[0,0]}) + 10 * 8 * (3 - 0) + (7 - 0) * 8 \\ &= \text{addr}(\texttt{A[0,0]}) + 240 + 56 \\ &= 1000 + 240 + 56 \\ &= 1296\end{aligned}$$

Problem 7.

a. **Answer.**

$$\sum_{\texttt{i}=1}^{10} \texttt{A[i]}$$

b. **Answer.**

$$\frac{1}{\texttt{A[1]}} + \frac{1}{\texttt{A[3]}} + \frac{1}{\texttt{A[5]}}$$

c. **Explanation.** The Jensen's device enables us to pass expressions in the funtion call which can be re-evaluated for multiple times during executions, and thus it provides convenience and flexibility for coding.

d. **Answer.** The `max` is as follow:

```
double max(first:integer /*by value*/, last:integer /*by value*/,
    incr:integer /*by value*/, i:integer /*by name*/, term:double /*by name*/)

    result:double := term
    i := first
    while i <= last do
        if term > result then
            result := term
        endif
        i := i + incr
    endwhile
    return result
```