HW 7
50pts, no extra credit
This is a team assignment.

Posted Tuesday, November 24, 2020

Due Friday, December 11, 2020

Problem 1 (5pts). Show the answer in *normal form* for expression

$$\mathit{twice}\ \mathit{twice}\ f\ x$$

where

$$\mathit{twice} \text{ is } \lambda f.\lambda x.f(f\ x)$$

Note: you do not need to show reduction steps to receive full credit. Write the normal form expression if it exists. Write "Normal form does not exist" otherwise.

Problem 2 (5pts). Show that the term ZZ where Z is $\lambda z.\lambda x.x(z\ z\ x)$ satisfies the requirement for fixed-point combinators that $ZZM =_\beta M(ZZM)$.

Problem 3 (5pts). In the following code, which of the variables will a compiler consider to have compatible types under structural equivalence? Under strict name equivalence? Under loose name equivalence?

```
type A = array [1..10] of integer
type B = A
a : A
b : A
c : B
d : array [1..10] of integer
```

Problem 4 (10pts). Explain the meaning of the following C declarations. Draw the type trees as we did in class.

```
double *x[n];
double (*y)[n];
double (*z[n])();
double (*w())[n];
```

Problem 5 (10pts). Consider the following declaration in C:

```
double (*bar(int, double(*)(double, double[])))(double);
```

Describe in English the type of `bar`. Draw the type tree.

How about

```
double ((*bar)(int, double(*)(double, double[])))(double);
```

Describe and draw the type tree. Is this a valid declaration in C? Explain your answer.

Problem 6 (5pts). Consider the following C declaration, compiled on a 32-bit Pentium machine:

```
struct {
     int n;
     char c;
} A[10][10];
```

If the address of `A[0][0]` is 1000 (decimal), what is the address of `A[3][7]`? Note: you may assume 32-bit integers and word-aligned structure fields.

Problem 7 (10pts). Consider the Pascal-like code for function `compute`. Assume that the programming language allows a mixture of parameter passing mechanisms as shown in the definition.

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

   result : double := 0.0
   i := first
   while i <= last do
       result := result + term
       i := i + incr
   endwhile
   return result
```

a. (2pts) What is returned by call `compute(1, 10, 1, i, A[i])`?
b. (2pts) What is returned by call `compute(1, 5, 2, j, 1/A[j])`?
c. (2pts) `compute` is a classic example of *Jensen's device*, a technique that exploits call by name and side effects. In one sentence, explain what is the benefit of Jensen's device.
d. (4pts) Write `max`, which uses Jensen's device to compute the maximum value in a set of values based off of an array `A`.