Problem 1 (15pts). [Modified from Sethi]. Verify the following equality:

\[ \text{twice} \ (\text{twice}) \ f \ x = \alpha \beta \ f \ (f \ (f \ x)) \], where \( \text{twice} \) is \( \lambda f . f \ (f \ x) \)

a. Using applicative order reduction
b. Using normal order reduction

Problem 2 (7.5pts). [From Sethi]. Show that the term \( ZZ \) where \( Z = \lambda z . \lambda x . x \ (z \ z \ x) \) satisfies the requirement for fixed-point combinators that \( ZZM = \alpha \beta M(ZZM) \).

Problem 3 (7.5pts). [Modified from Scott]. 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?

```plaintext
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). [Modified from Scott]. Explain the meaning of the following C declarations. Draw the type trees as we did in class.

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

Problem 5 (10pts). [Modified from Scott]. Consider the following declaration in C:

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

Describe in English the type of \( \text{bar} \). Draw the type tree.

How about

```c
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 (EXTRA CREDIT: 10pts). Using the applied lambda calculus of slides 7-13 of Lecture 18 (April 11), define a function to compute elements of the Fibonacci sequence.