

## Program 4 corrected

*C++ Funcadelia.*

**I.** Write a STL-style algorithm **for\_each\_pair** that takes two sequences and a binary functor, and calls the functor on each corresponding pair of elements in turn. That is, it takes:

$$\begin{array}{l} (a_0, a_1, \dots, a_{n-1}) \\ (b_0, b_1, \dots, b_{n-1}, \dots) \\ f \end{array}$$

and calls  $f(a_0, b_0), \dots, f(a_{n-1}, b_{n-1})$  in turn.

**II.** Write an STL style functional **mem\_cartesian\_product** that takes two no-parameter member functions  $C_1::f_1()$  and  $C_2::f_2()$  and returns a functor that does the following: Given two parameters ( $o_1$  of class  $C_1$ , and  $o_2$  of class  $C_2$ ), return the **std::pair** object consisting of ( $o_1.f_1()$ ,  $o_2.f_2()$ ).

Here is how it could be used with the algorithm in part I above:

```
for_each_pair( itFirstA, itLastA, itFirstB, compose( do_something_with_a_pair,
                                                    mem_cartesian_product( C1::f1, C2::f2 ) ) ) ;
```

Here **itFirstA**, **itLastA** are iterators into a sequence of **C1** objects, and **itFirstB** is an iterator into a sequence of **C2** objects.

See the source for **std::mem\_fun1\_t** in **<functional>** for some ideas.

**Due:** Thursday, November 14. This is an individual assignment. Submit these in one source file named **funcadelia.cpp**.