Today’s pair-programming partnerships are:

- Mr. Boateng Asante and Mr. Dewey
- Mr. Zahid and Mr. Schlager
- Ms. Byrne and Ms. Kyaruzi
- Mr. Nordin and Ms. Liddell
- Ms. DeWitt and Ms. Jervis
- Mr. Mataire and Mr. Champagne
- Mr. Mulhall

In each pair, the partner listed first should log in and manage the keyboard and screen.

**Exercises**

**A.** Create a subdirectory called **EXPLICIT-REFS** in the directory you’re using for labs in this course and copy my implementation of the EXPLICIT-REFS programming language into it. The `/home/stone/courses/languages/code/EXPLICIT-REFS` directory contains seven language-specific modules:

- `expvals-and-environments.scm`
- `stores.scm`
- `tokens.scm`
- `syntax-trees.scm`
- `scanner.scm`
- `parser.scm`
- `interpreter.scm`

**B.** (1) Write a **proc**-expression in EXPLICIT-REFS whose value is a procedure that swaps the contents of two given references. Since EXPLICIT-REFS supports only unary procedures, you’ll need to use currying: The procedure you write should take the first reference as an argument and return a procedure that takes the second reference as an argument.

   (2) Test your **proc**-expression by embedding it in a program that creates and initializes two references, calls the swapping procedure, and then constructs a value that proves that contents of the two references were indeed swapped.

**C.** Many programming languages that explicitly support references also provide a “null value,” a one-of-a-kind expressed value that is not a number, not a boolean, not a procedure. The null value has no content and its most common uses are to serve as a sentinel marking the end points or leaves of data structures and to serve as a placeholder when it is convenient to create a new reference that will not be initialized until later.

Change your copy of the implementation of EXPLICIT-REFS to add the null value as a new kind of expressed value. You’ll need to add an `expval->null` procedure, which will involve making a design decision: You could either say that it’s always an error to try to pull a conventional value out of the null expressed value, or you could choose some conventional value in Scheme such as the empty list or the false Boolean value to serve,
more or less arbitrarily, as the avatar of the null value. Consider the pros and cons of this decision and implement your choice.

Also add a binding to the initial environment, associating the identifier `null` with the null value.

Test your changes by writing and running an EXPLICIT-REFS program that creates a reference containing the null value, then uses a `setref`-expression to overwrite that value with the number 13, and finally uses a `deref`-expression to recover the contents of the reference as the value of the program.

Incidentally, what happens if you write a program that has the null value as its value? Try it and see what your interpreter does, and make sure that you can explain the result you get.