Exercises 11 (Procedural abstraction)

11A. *(C function declarations)*

The syntax of C function definitions is (in outline):

decl = …
| *type id* ‘`(`’ *formals*$^?$ ‘`)`’ *block-com*

Modify the syntax so that a function’s body is an expression rather than a block command. Show that this change, in isolation, would reduce the expressive power of functions. What other changes to the language would be needed to compensate?

11B. *(C functions and proper procedures)*

Modify the syntax of C function definitions and function calls to make a clear distinction between proper procedures and function procedures.

11C. *(Copy vs reference parameters)*

Consider the following Ada procedure:

```
type Vector is array [1 .. 3] of Float;
procedure add (v1, v2: in Vector; s: out Vector) is
begin
  for i in 1 .. 3 loop
    s[i] := v1[i] + v2[i];
  end loop;
end;
```

Here the keyword “**`in`**” signifies data flow *into* the procedure, whilst “**`out`**” signifies data flow *out of* the procedure.

The Ada compiler is free to adopt *either* copy *or* reference mechanisms for array parameters.

Consider the procedure call “`add(a, b, c);`”.

(a) Assuming that *copy* mechanisms are used, what happens to the three parameters, (i) on entry to the procedure, and (ii) on exit from the procedure?

(b) Assuming that *reference* mechanisms are used, what happens to the three parameters, (i) on entry to the procedure, and (ii) on exit from the procedure?

(c) Show that the procedure call has the same outcome, whichever parameter mechanisms are used.

Exercises 12 (Data abstraction)

12A. *(Programming without data abstraction)*

C does not support packages, abstract data types, or classes.

(a) Consider the dictionary module of slide 12-3. How can we program this module in C, in such a way as to hide the data structure?

(b) What are the disadvantages of programming in this way?

12B. *(Abstract data type for complex numbers)*

Design a `Complex` abstract data type, whose values are complex numbers. Equip your abstract type with operations such as magnitude, complex addition, complex subtraction, and complex multiplication.

Implement the abstract data type in Java, using a suitable representation for `Complex`.

12C. *(Classes for geometric shapes)*

Consider the Java class declarations `Shape`, `Circle`, and `Box` starting at slide 12-14.

(a) Define classes for straight lines and text boxes, placing them in an appropriate class hierarchy. Each of these classes must define its own `draw` operation.

(b) Define a `Picture` class, such that each `Picture` object represents a collection of shapes. Provide an operation that draws the whole picture, by drawing all of its component shapes.

Exercises 13 (Generic abstraction)

13A. *(Generic class for lists)*

Consider the Java generic class `List<T>` from the course notes (starting at slide 13-4). Assuming that you are given a suitable class `Airport`, show how to instantiate `List<T>` to declare a variable `itinerary` that will contain a list of airports.

13B. *(Generic class for priority queues)*

Consider the Java generic class `PQueue<T>` from the course notes (starting at slide 13-11). Assume that you are given the class `Fault` outlined below:

```
class Fault {
// A Fault object is a date-stamped fault report.
  public Date date;
  public String description;
  …
}
```

(a) Show how the `Fault` class declaration must be modified if you want to instantiate `PQueue<T>` with type `Fault`.

(b) Show how to instantiate the generic class to declare a variable `log`, which will contain a priority queue of fault reports ordered by date.

13C. *(Generic class for binary relations)*

A *binary relation* is a unordered collection of pairs (x, y) in which no pair is duplicated. Assume that the relation is homogeneous, i.e., all the x values have the same type, and all the y values have the same type, but the two types may be different.

Typical operations are to construct an empty relation, to add a given pair of values to a relation, to test whether a relation contains a given pair of values, and to retrieve the set of all y values paired with a given x value.

(a) Design a Java generic class that implements homogeneous binary relations. The class must be parameterized with respect to the type of the x values and the type of the y values. Provide all the above operations. Assume as little as possible about the types of the x and y values. What assumption(s) are you absolutely forced to make about these types?

(b) Suppose now that an operation is to be added to display the contents of the relation, sorted by x. What assumption(s) are you now forced to make about these the type of the x values?