

**Honors Programming Languages  
G22.3110 Fall 2005**

**Final Exam**

1. (a) Briefly explain why an Ada package is not an abstract data type (ADT).  
(b) Implement in Ada a simple `automobile` ADT, which supports the operations `drive`, `turn`, and `stop`.  
(c) Ada95 added the concept of *protected type*. Describe what was used when programming Ada83 to support the kind of functionality provided by Ada95 protected types. Why are protected types more desirable than Ada83's solution?
2. In Scheme, write definitions for `cons`, `car`, and `cdr` that don't use any of the built-in Scheme list operations (such as the built-in `cons`, `car`, `cdr`, `list`, etc.). That is, the new `cons`, `car`, and `cdr` should appear to the user as if they behave just like the built-in versions, but are in fact implemented completely differently.
3. Algol60 used *pass by name* parameter passing. Consider the following Algol60 (more or less) program:

```
integer x,y;

procedure f(a,b);
integer a,b;
begin
    integer i;
    for i := 1 to 10 do
    begin
        b := b + 1;
        print(a);
    end;
end;

begin
    x := 0; y := 1;
    f(x+y, x);
end;
```

Write in C a program whose run-time behavior is identical (in terms of its implementation) to the above program. It is not sufficient to write a C program that happens to produce the same output. Think about how pass by name is actually implemented (it's not textual substitution).

4. Suppose you wanted to write a program that constructed several directed graphs. In each graph, all nodes were labelled with values of the same type, but the label types of the different graphs were different. Each node in a graph can have an arbitrary number of arcs emanating from it.
  - (a) Give an ML type definition, using only the purely functional subset of ML, for a graph in which the label type is left unspecified (to allow graphs with different label types to be declared using the type you define).

- (b) Using the above type definition, could your program construct a tree? Could it construct a DAG? Could it construct a graph with cycles? Justify your answer in each of the three cases by giving either the code (only a couple of lines, please) or explaining why it can't be done.
  - (c) Can a tree traversal procedure be written in ML, if the tree is defined using the type definition in part a? What about a procedure for traversing a DAG (printing out the label of each node only once)? How about traversing a graph with cycles? Note: These are yes/no answers, with a line or two of explanation. No code please.
  - (d) If any of your answers above indicated that the type definition in part a didn't allow the definition of a certain type of graph or a traversal of a certain type of graph, then fix the type definition and write the code for a procedure that can traverse any kind of graph, printing out the label of each node only once.
5. In the context of the language with assignment, aliasing, and goto that we discussed in class, give the denotational semantics of the expression

`call/cc e`

where `e` is itself an expression. You might remember in Scheme that, for example,

`(call/cc (lambda (k) ...))`

applies the function defined by `(lambda (k) ...)` to the current continuation. Be sure to describe any necessary changes to the semantic domains and semantic functions.

6. Write in SETL, using the expressive power of the language, a procedure `BIN` that solves the one dimensional Bin Packing problem exactly. The Bin Packing problem can be stated as follows: Given a bin size and a set of objects of varying sizes, what is the minimum number of bins necessary to hold all the objects, such that the sum of the sizes of the objects in each bin is no greater than the size of the bin? Assume the input to `BIN` is an integer bin size and a set of integers representing the sizes of the objects. Do not worry about the computational complexity of `BIN`.
7. (a) What subtyping relationships must hold among the types  $\tau_1, \dots, \tau_{12}$  in order for the following to hold in  $\lambda_{<}^{\rightarrow, \text{record}}$ ?

$$\vdash (\lambda x : \{b : \tau_1 \rightarrow \tau_2\} \rightarrow \tau_3. \lambda y : \{a : \tau_4, b : \tau_5 \rightarrow \tau_6\}. x y) : \{b : \tau_7 \rightarrow \tau_8\} \rightarrow \{a : \tau_9, b : \tau_{10} \rightarrow \tau_{11}\} \rightarrow \tau_{12}$$

(note that my transparencies used  $\triangleright$  instead of  $\vdash$ , but assume they mean the same thing).

- (b) Assuming your answer from part (a) is true, show the type derivation that proves that the expression is well typed.