

**Honors Programming Languages
G22.3110 Spring 2008**

**Final Exam
Take-Home Section**

Take this sheet home with you to answer the questions below. Your answers must be returned by Monday (before midnight). You can 1) email me the answers or 2) write them in a blue book, put it in an envelope, and slide it under my door (509 WWH).

1. 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`.
2. (a) ML's type system extends the predicative polymorphic lambda calculus with the `LET` construct. Even though it supports polymorphism, why isn't the predicative polymorphic lambda calculus alone sufficient to express ML-style polymorphism?
(b) Using the notation of the predicative polymorphic lambda calculus (with explicit type parameters) extended with the `LET`, explain precisely why in ML, if the identity function is defined by

```
fun id x = x
```

the expression

```
let val f = id id
in (f 3, f true)
end
```

generates a type error, but

```
(id id 3, id id true)
```

does not generate a type error.
- (c) Give the derivation, using the predicative polymorphic type rules and the `LET` rule, that shows that `(id id 3, id id true)` is valid. Also, show where the derivation for the `let` expression, above, fails.
3. Suppose two additional statements – a repeat-until loop and a break statement – were added to the language with assignment, pass-by-reference, and goto whose denotational semantics were given in class. That is, the syntax for statements is:

```
s ::= ... | repeat s until e | break
```

where the ... are the other forms of statements as discussed in the notes.

- (a) Using the framework given in class, give the denotational semantics of the repeat-until loop. In the repeat-until loop, the statement `s` is repeatedly executed until the value of expression `e` is true.

- (b) Using the framework given in class, give the denotational semantics of the break statement. The break statement, as in C, causes the innermost surrounding repeat-until loop to exit. Be sure that your semantic definition of the repeat-until loop works with your semantic definition of the break statement.