MASSACHVSETTS INSTITVTE OF TECHNOLOGY
Department of Electrical Engineering and Computer Science
6.001—Structure and Interpretation of Computer Programs
Spring Semester, 1995

Compiled Code – Recitation Notes, May 12

1. What scheme code was compiled to give the following?

```
 (assign val (op make-compiled-procedure) (label entry73) (reg env))
 (goto (label after-cond72))
entry73
 (assign env (op compiled-procedure-env) (reg proc))
 (assign env (op extend-binding-environment) (const (x)) (reg argl) (reg env))
 (assign val (op lookup-variable-value) (const x) (reg env))
 (goto (reg continue))
after-cond72
 (perform (op define-variable!) (const f) (reg val) (reg env))
 (assign val (const (the-unspecified-value)))
```

2. What scheme code was compiled to give the following? Where is the bug (what line is missing)?

```
  (save continue)
  (assign proc (op lookup-variable-value) (const =) (reg env))
  (assign val (const 0))
  (assign argl (op list) (reg val))
  (assign val (op lookup-variable-value) (const x) (reg env))
  (assign argl (op cons) (reg val) (reg argl))
  (test (op primitive-procedure?) (reg proc))
  (branch (label primitive-branch11))
compound-branch12
  (assign continue (label after-call71))
  (assign val (op compiled-procedure-entry) (reg proc))
  (goto (reg val))
primitive-branch11
  (assign val (op apply-primitive-procedure) (reg proc) (reg argl))
after-call71
  (test (op false?) (reg val))
  (branch (label false-branch70))
true-branch70
  (assign val (const 1))
  (goto (reg continue))
false-branch70
  (assign val (const 2))
  (goto (reg continue))
after-if18
```

Consider the following code output by the compiler:

```
 (assign val (op make-compiled-procedure) (label entry24) (reg env))
 (goto (reg continue))
entry24
 (assign env (op compiled-procedure-env) (reg proc))
 (assign env (op extend-environment) (const (f)) (reg argl) (reg env))
 (assign val (op make-compiled-procedure) (label entry25) (reg env))
 (goto (reg continue))
entry25
 (assign env (op compiled-procedure-env) (reg proc))
 (assign env (op extend-environment) (const (x)) (reg argl) (reg env))
 (assign proc (op lookup-variable-value) (const f) (reg env))
 (save continue)
 (save proc)
 (assign proc (op lookup-variable-value) (const f) (reg env))
 (assign val (op lookup-variable-value) (const x) (reg env))
 (assign argl (op list) (reg val))
 (test (op primitive-procedure?) (reg proc))
 (branch (label primitive-branch28))
compiled-branch27
 (assign continue (label after-call26))
 (assign val (op compiled-procedure-entry) (reg proc))
 (goto (reg val))
primitive-branch28
 (assign val (op apply-primitive-procedure) (reg proc) (reg argl))
after-call26
 (assign argl (op list) (reg val))
 (restore proc)
 (restore continue)
 (test (op primitive-procedure?) (reg proc))
 (branch (label primitive-branch31))
compiled-branch30
 (assign val (op compiled-procedure-entry) (reg proc))
 (goto (reg val))
primitive-branch31
 (assign val (op apply-primitive-procedure) (reg proc) (reg argl))
 (goto (reg continue))
after-call29
```

A. What source code expression was compiled to produce lines 14-25 of the object code?

B. What source code expression was compiled to produce lines 11-35 of the object code?

C. What source code expression was compiled to produce lines 6-35 of the object code?

D. What is the overall expression that compiled into the object code shown?

E. This code was compiled in such a way that a tail recursive call was produced. How would it differ if a simpler compiler had ignored that optimization?

F. How can you hand optimize out the need for binding `x` in the environment?

G. Can you do the same for `f`?