```
#|

 YOU ARE NOT PERMITTED TO USE THE STEPPER AT ANY POINT IN THIS EXAM.

 This document describes the questions for problem 3.  You read the
 questions here, but ALL of your answers must go into the
 mt1-p3-starter.rkt file, following the instructions there.

 The entire problem concerns the evaluation of expressions involving
 locals.

 Given the following two definitions, we want you to consider evaluation
 of the call to doit that follows them.

|#

(define FOO 10)

(define (doit a lon)
  (local [(define (fn x) (if (odd? x) (add1 x) (sub1 x)))
          (define b (foldr + 0 lon))
          (define (bar x)
            (local [(define (foo y)
                      (if (zero? x)
                          (+ x y (fn (+ b y FOO)))
                          (+ x y (fn (+ a b)))))]
              (foo a)))]

    (map bar lon)))

(doit 3 (list 1 2 3))
```

```
34
35  ;;
36  ;; PART 1
37  ;;
38  ;; Which of the following is an actual step in the evaluation of
39  ;; the expression?
40  ;;
41
```

```
42
43  ;; A
44
45  (define (fn_0 3)
46    (if (odd? 3) (add1 3) (sub1 3)))
47  (define b_0 (foldr + 0 (list 1 2 3)))
48  (define (bar_0 x)
49    (local
50     ((define (foo y)
51        (if (zero? x)
52          (+ x y (fn_0 (+ b_0 y 10)))
53          (+ x y (fn_0 (+ 3 b_0))))))
54     (foo 3)))
55
56  (map bar_0 (list 1 2 3))
57
58
59
60
61
62  ;; B
63
64  (define (fn_0 3)
65    (if (odd? 3) (add1 x) (sub1 x)))
66  (define b_0 (foldr + 0 lon))
67  (define (bar_0 x)
68    (local
69     ((define (foo y)
70        (if (zero? x)
71          (+ x y (fn_0 (+ b_0 y FOO)))
72          (+ x y (fn_0 (+ a b_0))))))
73     (foo a)))
74
75  (map bar_0 (list 1 2 3))
76
77
```

```
 77
 78  ;; C
 79
 80  (define (fn_0 x)
 81    (if (odd? x) (add1 x) (sub1 x)))
 82  (define b_0 (foldr + 0 (list 1 2 3)))
 83  (define (bar_0 x)
 84    (local
 85     ((define (foo_0 a)
 86        (if (zero? x)
 87          (+ x a (fn_0 (+ b_0 y 10)))
 88          (+ x a (fn_0 (+ a b_0))))))
 89     (foo_0 a)))
 90
 91  (map bar_0 (list 1 2 3))
 92
 93
 94  ;; D
 95
 96  (define (fn_0 x)
 97    (if (odd? x) (add1 x) (sub1 x)))
 98  (define b_0 (foldr + 0 (list 1 2 3)))
 99  (define (bar_0 x)
100    (local
101     ((define (foo y)
102        (if (zero? x)
103          (+ x y (fn_0 (+ b_0 y FOO)))
104          (+ x y (fn_0 (+ 3 b_0))))))
105
106     (foo 3)))
107
108  (map bar_0 (list 1 2 3))
109
110
111
```

```
119 |
120 | ;; PART 2
121 | ;;
122 | ;; Which of the following expressions never gets evaluated?
123 | ;;
124 | ;;
125 | ;; A. (add1 x)
126 | ;; B. (sub1 x)
127 | ;; C. (foldr + 0 lon)
128 | ;; D. (odd? x)
129 | ;; E. (zero? x)
130 |
131 |
132 | ;; PART 3
133 | ;;
134 | ;; Which of the following is a closure?
135 | ;;
136 | ;;
137 | ;; A. doit
138 | ;; B. fn
139 | ;; C. b
140 | ;; D. bar
141 |
142 |
143 | ;; PART 4
144 | ;;
145 | ;;
146 | ;; How many definitions are lifted during the evaluation?
147 | ;;
148 | ;; 3, 4, 5, 6, 7, or 8
149 |
150 |
151 | ;; PART 5
152 | ;;
153 | ;;
154 | ;; How many times is foo lifted during the evalation?
155 | ;;
156 | ;; 1, 2, 3, 4, or 5
157 |
```