Your name:

**Due:** Wednesday, October 30

**Instructions:** Review the course honor policy for written homeworks.

This cover sheet must be the front page of what you hand in. Fill out the left column in the table to the right after we go over each problem in class, according to the rubric below.

This rubric is also on the website, in more detail, under “Other Stuff”→“Grading Rubrics”.

Make sure all problems are submitted IN ORDER.

- **5:** Solution is completely correct, concisely presented, and neatly written.
- **4:** The solution is mostly correct, but one or two minor details were missed, or the presentation could be better.
- **3:** The main idea is correct, but there are some significant mistakes. The presentation is somewhat sloppy or confused.
- **2:** A complete effort was made, but the result is mostly incorrect. There may be some basic misunderstandings of the topic or the problem.
- **1:** The beginning of an attempt was made, but the work is clearly incomplete.
- **0:** Not submitted.

**Comments or suggestions for the instructor:**

**What other students did you work with?**

**Citations** (be specific about websites):
1 CRT 1

Consider the following program in a Scheme-like syntax:

(define b 10)

(define (f a)
  (if (eqv? a 'second)
      b
      (g (* b 2)))))

(define (g a)
  (let ((b (+ a 5)))
    (f 'second)))

(f 'first)

Use a Central Reference Table to determine how this program would behave under dynamic scoping. Draw the state of the complete CRT at the beginning of the call to (f 'second).

(Remember that a CRT includes the stack of sets of variable names for “names in scope” as well as a stack of bindings for every variable in the program.)
Consider the following program in a C-like syntax:

```c
int x = 10;
int i = 5;

int foo(x) {
    if (x == i) {
        return 3;
    }
    else {
        int i = x - 1;
        int j = foo(i);
        return 3 * j;
    }
}

print foo(3);
```

Do like before: trace the execution of this program using a CRT. To turn in, draw the complete CRT at the moment that there are 3 bindings in the stack for \( x \). Also indicate clearly what is finally printed by this program.
3 Scope Tree

Draw the scope tree for the program from the previous problem. Then indicate what the final printed value would be using *lexical* scoping instead of dynamic scope as before.

4 Frames and closures

Consider the following SPL code, which we will imagine is lexically scoped:

```splat
new counter := lambda start {
    new increment := lambda by {
        start := start + by;
        ret := start;
    };
    ret := increment;
};
new A := counter @ 0;
new B := counter @ 5;
write A@0;
write B@0;
write A@6;
```

Draw all the frames and links that result after executing this program. See the reading assigned from Unit 6 for exactly how these should be drawn, particularly Section 3.2.3 of SICP.

Specifically, every variable name that refers to a function should point to a closure, which is represented by a pair of circles, pointing to the referencing environment and the function definition, respectively. (You do NOT have to write out the complete body of every function.)

(Use the back of the page.)