SI 413 Fall 2012: Homework 8

Your name:

Due: Friday, 26 October, before class

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 more concise.
- **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.

Problem	Self-assessment	Final assessment
1		
2		
3		
4		

Comments or suggestions about this homework:

Comments or suggestions about the course so far:

Citations (other students, websites, ...):

Use a separate sheet of paper for your answers! Everything should be submitted in one packet, all printed out for me to see.

1 Scope Tree

a) Draw the scope tree for the following program:

```
def account(a):
    def withdraw(x):
        if a < x:
            return False
        else:
            a = a - x
            return True
    return withdraw # Notice a function is being returned!

var A = account(10)
var B = account(12)

print A(11)
print B(11)
```

b) Draw the scope tree for the following program:

```
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;
  }
}

cout << foo(3) << endl;
```

2 Lexical scope

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

```
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.

In particular, 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.)

3 Pass by what?

Here is a small SPL program with a single function that gets called twice. (Not that it matters, but you may assume lexical scoping here.)

```
new a := 20;
new b := 10

new foo := lambda x {
  x := x + x;
  ret := x * b;
}

write foo@a;
write a;
write foo@b;
write b;
```

Clearly four numbers will get printed by this piece of code. Tell me what those four numbers will be under:

a) Pass by value

b) Pass by reference

c) Pass by value/result

4 Value/result example

In C++, function parameters that are specified with an ampersand, like the `a` in

```
void foo(int& a);
```

are passed by reference. In class, we saw a different kind of parameter passing mode called pass by value/result. Write a small C++ program demonstrating that reference parameters (with ampersands) really are passed by reference and not by value/result. That is, your program should do something different in each of these parameter passing modes.

As always, I want you to come up with your own examples! You can work together on homeworks, but the examples you turn in should be unique.