Name SOLUTIONS

Thursday, September 24

Email 6.818-www@mit.edu

6.818 Fall 2020 Miniquiz #11 5 Minutes

1. Consider the extension to IMP that we saw last lecture, which added block scopes to the language. Suppose we redefine variable declarations as follows, with changes highlighted in bold (where · denotes the empty stack):

$$\frac{\neg\big(x \in \boldsymbol{visible}(\gamma :: \sigma)\big) \quad (\gamma :: \sigma,\ h,\ e) \to v \quad \neg(a \in dom(h)) \quad \sigma[x : a] = \sigma' \quad h[a : v] = h'}{(\gamma :: \sigma, h, \text{var } x = e) \to (\gamma :: \sigma',\ h')}$$

$$\frac{\boldsymbol{dom}(\sigma) = \boldsymbol{d}}{\boldsymbol{visible}(\cdot :: \sigma) = \boldsymbol{d}} \qquad \frac{\boldsymbol{visible}(\gamma) = \boldsymbol{d_1} \qquad \boldsymbol{dom}(\sigma) = \boldsymbol{d_2} \qquad \boldsymbol{d_1} \cup \boldsymbol{d_2} = \boldsymbol{d}}{\boldsymbol{visible}(\gamma :: \sigma) = \boldsymbol{d}}$$

In plain English, how does this modification change the semantics of variable declarations?

The modification above disallows variable shadowing (i.e., declaring a variable with the same name as another variable in an outer scope would result in the program getting stuck).

2. What is the output of each program below?

```
var g = 3;
var x = 2;
var z = 4;
var f = fun(x) {
    var g = fun(y) {
        return x * y;
    }
    return g(z);
}
print(f(g));
```

Output: 12

```
var g = fun() {
    var x = 0;
    var f = fun() {
        x = x + 1;
        return x;
    };
    return f;
};
var a = g();
var b = g();
a();
a();
print(b());
```

Output: 1