Darsamian Lab 3 Write-Up

Sam Bennetts, D’Artagnan Wake

2. Give one test case that behaves differently under dynamic scoping versus static scoping (and does not crash). Explain the test case and how they behave differently.

```
const x = 1;

const g = function g(y) { const c = x; const x = c-1; return y === 0 ? 1 : c * g(x) };

const h = function h(x) { return g(2) };

h(3)
```

Static Scoping: Returns 1.0

Dynamic Scoping: Returns 6.0

With static scoping the binding “const x = 1” replaces all free variables x with the value 1. Thus, in the body of the function g(y), the values for x are replaced with a 1 and the function performs a factorial on 1. Even though the function h(x) is called with a 3, the value of x will still be replaced with a 1 because of the const binding at the beginning of the code.

Under dynamic scoping conditions the value of x gets updated. Although we initially extend the environment so x = 1, the function call h(3) re-extends the environment so x now = 3. Thus, the body of the function g performs a factorial on x = 3 and returns a value of 6.

3. Explain whether the evaluation order is deterministic as specified by the judgment form e → e’.

This judgement form is deterministic. Small step semantics always evaluates expressions one step at a time, and must step on the first expression before the second. This is an important property that allows expressions to always evaluate in the same way. Such evaluation allows us to do things such as short circuit code, which can save a great amount of time and space.

4. Give an example that illustrates the usefulness of short-circuit evaluation. Explain your example.

```
If( a == 3 && b == 5) { do something}

Else {do something else}
```

Short-circuit evaluation can be extremely useful in terms of saving time and space while running code. In this example we can see two short circuit evaluations. The first is within the if statement where ‘a’ and ‘b’ must both be true in order to proceed with the code. By checking ‘a’ first we are able to short circuit because, if a is false, we can proceed to the else statement without having to check ‘b’ thus saving us the time it would take to see if b is true. The second short circuit evaluation is the if-else statement itself. By placing some body of a code within an if statement we are able to save time (and space!!) of evaluating code if conditions ‘a’ and ‘b’ are not met (i.e. we can skip right to the body of code in the else statement).