Andrew Rutherford

CSCI 3155

2a.

A case where static and dynamic scoping would provide two different outputs:

```
const x = 100
const product = function(x) { return function(y) { return x * y} };
product(5)(5)
```

With static scoping, product would return 25; the previously declared x = 100 is out of scope and would not be evaluated. With dynamic scoping, product would return 500, because the x would be defined as 100.

3c.

The evaluation order is deterministic as specified by the judgement form e -> e', because there is no way an expression can match two unique cases in any given step.

4.

The evaluation order for e1 + e2 is e1 first, e2 second, and the plus operator last. In order to change the evaluation order you would need to redesign the SEARCH*** cases. SEARCHBINARY would need to evaluate the right side to a value, and SEARCHBINARYARITH would need to evaluate the left side.

5a.

(false && (7 / 0))

For &&, the second expression is not ever evaluated if the first expression is “falsey.” This is useful in this example because the second expression would throw a “divide by zero” error. Never evaluating the second expression when the first one fails also results in faster run time and better performance.

5b.

e1 && e2 does short circuit because according to the inference rule DOANDFALSE, if v1 evaluates to false, e2 is never evaluated.