Lab 3: Write-Up

2a. An example of a test case in which static scoping produces a different result than dynamic scoping:

```
const x = 5;
const add = function(x){return function(y){return x + y;}};
jsy.print(add(3)(3));
```

With static scoping, the result is 6, but with dynamic scoping, the result would be 8. With a dynamic scope, the compiler will use the earlier assignment of x.

3c. The evaluation order specified by the judgment form *e → e'* is deterministic. A set of inference rules is called deterministic if there is at most one rule that specifies how to get to the "next" step of evaluation. All of the Do rules in the small-step semantics defining *e → e'* follow this principle because they all specify left to right evaluation.

4. The evaluation order for *e1 + e2* is left to right. This is shown using the Search inference rules of the small-step semantics. For example, if we look at SearchBinary, we see that if e1 is evaluated to e1', then the expression *e1 bop e2* evaluates to *e1' bop e2* in one step. Now looking at SearchBinaryArith, we see that *e2* is evaluated to *e2'* only after the left operand (e1) has evaluated to a value v1. To change the inference so that the evaluation order is right to left:

SearchBinary:

$$\frac{e_1 \rightarrow e_1'}{e_1 bop\ e_2 \rightarrow e_1'\ bop\ e_2}$$

SearchBinaryArith:

$$\frac{e_2 \rightarrow e_2'\ bop \in [....]}{v_1 bop\ e_2 \rightarrow v_1\ bop\ e_2'}$$

5.

a)

Short-circuit evaluation is useful in evaluating conditionals. For example, consider this while loop:

```
while(amountOfMoney != 0) && (numberOfItems <= bagCapacity){
     //Do stuff
}
```

There is no point in evaluating whether the number of items in the shopping bag is acceptable if all of the money is gone. By evaluating that condition first, if it evaluates to false, we know to exit the loop without checking the other conditional.

b)

e1 && e2 short-circuits if e1 evaluates to **false.** This can be seen using the two inference rules DoAndFalse and DoAndTrue. With DoAndFalse, if v1 evaluated as a Boolean is false, the v1 && e2 will evaluated to v1, which is false. This short-circuits the evaluation of e2 as it is never evaluated to a value. However, with DoAndTrue, if v1 evaluated as a Boolean is true, then e2 will be evaluated instead of short-circuited.