Will Christie
Lab3 Write Up

1. See survey results.
2. **JavaScripty Interpreter: Tag Testing, Recursive Function, Dynamic Scoping.**

```
Const a = 3;
Function myFunction( ) {x = a + 3; return x;}
Function myFunction2( ) {a = 2; return myFunction();}
myFunction( );
myFunction2();
```

This test case is one that does not behave the same under dynamic scoping vs static scoping.

STATIC:

```
myFunction() //returns 6
myFunction2() //returns 6
```

DYNAMIC:

```
myFunction() //returns 6
myFunction2() // returns 5
```

The difference in this case is that the environment changes when the function is called when evaluated using big-step instead of being substituted when evaluated with small steps.

   a. See implementation.
3. **JavaScripty Interpreter: Substitution and Evaluation Order.**
   a. See implementation.
   b. See implementation.
   c. The evaluation order **is deterministic** as specified by the judgment form e→e'. This is because every case in the step function that we were provided in Figure 7,8,9 is based on the judgment form e→e'. Because e→e' is a one step reduction, the evaluation order is deterministic because it shows us the exact reduction order as compared to big-step semantics in which we have no idea what the particular evaluation order may be.
4. **Evaluation Order:**
   a. In order to evaluate e1 + e2, we must first step e1 down to a value, after which e2 will be stepped down to a value. Once both e1 and e2 are values, then the **sum of values** e1 and e2 will be performed. If we were to change the rules in order to obtain the opposite evaluation order we would have reverse the order of the specifications, ensuring that we stepped e2 to a value first, then stepping e1 to value, and then performing the sum of value e2 and e1.

Ex: e2→e2'

e1 + e2 → e1 + e2'

e1→e1'

e1 + v2 = e1' + v2

5. **Short Circuit Evaluation:**
    a. An example of the usefulness of short-circuit evaluation is in examining the statement (x != null && x.get() != 42). This is a very generic example of the usefulness of short-circuiting because we do not want to throw a null pointer exception when x is in fact NULL. Because there is short-circuiting we evaluate to false when x is NULL and do not attempt to dereference a NULL pointer.
    b. Based on small step semantics, e1&&e2 will short circuit because the DoAndFalse rule has been put in place to handle the case in which e1 is false.