# CSCI 3155 Problem Set 3

Anthony Tracy

02/24/2018

1. ***Feedback: Complete survey linked from the moodle after completing this assignment. Any non-empty answer will receive full credit.***

   This has been done.

2. ***JavaScripty Interpreter: Tag Testing, Recursive Functions, and Dynamic Scoping.***

   The code to be implemented was done. As for the write-up for this section, the following is a test case that would behave differently under dynamic scoping vs static scoping.

```
 1 x = 0;
 2 function f1(){
 3    var x = 10;
 4    f2();
 5 }
 6
 7 function f2(){
 8    y = x+10;
 9 }
10
11 f1();
12 console.log(y);
13 console.log(x);
```

   If this were to be dynamically scoped then it would print 20 on line 12 and 0 on line 13. This is due to f2() getting the environment from when it was called in f1(). Which would then override the global x=0 for x=10, then y would be equal to 20.

However, if this were statically scoped then it would print 10 on line 12 and print 0 on line 13. This is because statically scoped means that each function has its own work space. I.e - function f1() uses the var x that is defined on line 3, then when f2() is called the environment isn't updated with the current var x, and so f2() doesn't have a declaration of x so it looks to the global declaration at line 1.

3. ***JavaScripty Interpreter: Substitution and Evaluation Order.***

   The code to be implemented was done. as for the write-up portion of this, this would be deterministic. Considering $SearchBinary_1$ and $SearchBinary_2$, the first step is to look at expression 1, $e_1$ and to step it to $e_1'$ and return another binary operator with the new $e_1' bope_2$. Then repeat until $e_1$ is just a value, then it moves to evaluating $e_2$.

4. ***Evaluation Order.*** Consider the small-step operational semantics for JAVASCRIPTY shown in Figures 7, 8, and 9. What is the evaluation order for e 1 + e 2 ? Explain. How do we change the rules obtain the opposite evaluation order? It will first look to make sure that $e_1$ is not a value, if it is not it will take another step and return the $e_1' bope_2$ which will continue to be evaluated until the check for $e_1$ until it returns true, in which case it will then begin small-step evaluation of $e_2$ where it will continue to check if not a value until it is a value. If we wanted to change these rules to have it evaluate right to left instead, then we would first look at $e_2$ and basically reverse the process.

5. ***Short-Circuit Evaluation.***

   (a) ***Concept.*** Give an example that illustrates the usefulness of short-circuit evaluation. Explain your example.

```
// Make some object with stats:
var someObj = {
  stat1: 'A word',
  stat2: 'Another word',
  stat3: 10
}
```

```
33
34
35 // Case where the stat exists:
36 console.log(someObj.stat1 || 'no stat here')
37 // Case where it doesn't:
38 console.log(someObj.stat4 || 'no stat here')
```

In the above code it shows that short circuiting can help us avoid errors but instead print the default response that we want. For example in line 38 someObj.stat4 hasn't been added to the object yet so instead of breaking the code we can short circuit to a 'no stat here' response.

(b) ***JAVASCRIPTY.*** Consider the small-step operational semantics for JAVASCRIPTY shown in Figures 7, 8, and 9. Does $e_1$ AND $e_2$ short circuit? Explain.

Given our small-step semantics, we would not always see it short circuit. This is because when $e_1$ is not a value $v_1$ it will first make a small-step operation to reduce $e_1$, which it will first do before attempting to evaluate $e_1$ with a short circuit.