CSC344 – Assignment 2

12 Points Due ~~3/6/17~~ 3/8/17

Write a set of Clojure functions that perform symbolic simplification and evaluation of boolean expressions using “and”, “or”, and “not”. “not” will be assumed to take one argument, while “and” and “or” will take one or more. You should use false for False and true for True.

Expressions are created as unevaluated lists. For example:

```
(def p1 '(and x (or x (and y (not z)))))
(def p2 '(and (and z false) (or x true)))
(def p3 '(or true a))
```

set p1, p2, and p3 to the given unevaluated expressions. Start off with three functions that build (unevaluated) expressions:

```
(defn andexp [e1 e2] (list 'and e1 e2))
(defn orexp  [e1 e2] (list 'or e1 e2))
(defn notexp [e1] (list 'not e1))
```

For example, p3 could have been created using

```
(def p3 (orexp true 'a))
```

You will need to modify andexp and orexp to allow for one or more arguments.

The main function to write, “evalexp”, entails functions that simplify, bind, and evaluate these expressions.

Simplification consists of replacing particular forms with equivalent functions, including the following (you may add others too):

```
    (or true) =>; true;
    (or false) => false;
    (and true) => true;
    (and false) => false;
    (or x false) => x;
    (or false x) => x;
    (or true x) => true;
    (or x true) => true;
    (and x false) => false;
    (and false x) => false;
    (and x true) => x;
    (and true x) => x;
    (not false) => true;
    (not true) => false;
    (or x y z) => (or x (or y z));
    (and x y z) => (and x (and y z));
    (not (and x y)) => (or (not x) (not y));
    (not (or x y)) => (and (not x) (not y));
```

Binding consists of replacing some or all of the variables in expressions with constants (true or false), and then returning the partially evaluated form.

The evalexp function should take a symbolic expression and a binding map and return the simplest form (that might just be a constant). One way to define this is

```
  (defn evalexp [exp bindings] (simplify (bind-values bindings exp)))
```

Example:

```
(evalexp p1 '{x false, z true})
```

binds x and z (but not y) in p1, leading to `(and false (or false (and y (not true))))` and then further simplifies to just false.

For full credit, you should demo your project to me by the end of the day on ~~3/6/17~~ 3/8/17. Each day late will result in a 5% penalty.