CS 301, Summer 2017
Lab 6
DUE: 11:59pm Friday, August 4th
60 Points Total

NOTICE: when defining functions, use the exact name given in this assignment sheet (including matching capitalization.)

In this lab assignment we will attempt to perform certain operations on a list of nested lists and atoms representing a logical expression.

Using the logical operators, **and, or, implies, not,** and **iff,** and atoms representing truth values, we can make a list that would represent proposition:

For instance: **'(A and (not A))**, or **'((B iff (A or C)) implies (not (C implies A)))** would represent the formal statements $(A \wedge \neg A)$ and $((B \equiv (A \vee C) \rightarrow \neg(C \rightarrow A)))$.

We use the following convention when we represent propositions in this way: for every operator – unary or binary – there corresponds at least – perhaps more - one pair of parentheses defining its scope. Thus, the lists **'(A and not A)** and **(B iff A or C)** are not permitted but **'((A and ((not A))))** and **'(B iff (((A or C))))** are acceptable.

Your task this week will be to define three different functions:

1. Define **collect-prop-variables**, which will take as input a list representing a proposition, and return a list, representing a set, of all the variables used in the expression. Example: **(collect-prop-variables '(A and (not A)))** should return **'(A)** and **(collect-prop-variables '((B iff (A or C)) implies (not (C implies A))))** should return **'(A B C).**

2. Define **substitute**, which will take as input a list representing a truth expression, a variable, and an element (of any type.) Substitute will replace all instances of the variable in the truth expression with that element. For example: **(substitute '(C or (D or D)) 'D #f)** should return

'(C or (#f or #f)) and **(substitute '(not ((B and A) or (A implies B))) 'B '(#t and X))**

should return

'(not (((#t and X) and A) or (A implies (#t and X))))

3. Define **Evaluate-WFF,** which will take as input a formula of logic in which there are only truth values and no variables, such as the following: **(#t or (not #t))**.

Evaluate-WFF should return **#t** if the formula under the particular truth assignment evaluates to **#t** and false otherwise.

By way of example,

(Evaluate-WFF ‘(#t or (not #t))

should return **#t** and

(Evaluate-WFF ‘(#t and (not #t)

should return **#f**.