CSC421 Assignment 2. Spring 2015 (10pts)

Birds can fly, unless they are penguins and ostriches, or if they happen to be dead, or have broken wings, or are confined to cages, or have their feet stuck in cement, or have undergone experiences so dreadful as to render them psychologically incapable of flight

Marvin Minsky

`Student Name:` `Student Number:`

`Instructor George Tzanetakis`

Question	Value	Mark
1	2	
2	2	
3	2	
4	2	
5	2	
Total	10	

1 Overview

The goal of this assignment is to familiarize you with Propositional and First-order Logic as well as automated inference. Be aware that some of the questions require much more work than others. Your deliverable will be a report documenting your work. You can use any programming language to implement the programming parts of the assignment. Don't hesitate to contact the instructor via email or the through ConneX with any questions/clarifications you might need.

2 Propositional Logic 1 (2pts)

Design a syntax for propositional formulas using regular ASCII characters. You might find it easier to use prefix rather than infix notation and only support boolean operators that take only two arguments (e.g., ((A $\wedge$ B) $\wedge$ C) rather than (A $\wedge$ B $\wedge$ C). In prefix notation the operator precedes the operands so for example $5 + 2$ would be written $+52$. This makes parsing much easier as a simple recursive function can be used. To make use of your design, write a recursive function to eliminate implication in a formulate by replacing subexpressions of the form $(A \rightarrow B)$ with ($\neg A \vee B$). Your implementation should also expand equivalences so that expressions of the form $A \leftrightarrow B$ are replaced by $(A \rightarrow B) \wedge (B \rightarrow A)$.

3 Propositional Logic 2 (2pts)

Using the syntax you defined for input in the previous question, implement a truth evaluator eval(formula, truthAssignemnt) which evaluates whether a formula hold for a particular truth assignment. Use your evaluator to answer the following question:

Let A be the formula:

$$((p_1 \rightarrow (p2 \wedge p_3)) \wedge ((\neg p_1) \rightarrow (p_3 \wedge p_4))) \tag{1}$$

Let B be the formula:

$$((p_3 \rightarrow (\neg p_6)) \wedge ((\neg p_3) \rightarrow (p_4 \rightarrow p_1))) \tag{2}$$

Let C be the formula:

$$((\neg(p2 \wedge p_5)) \wedge (p2 \rightarrow p_5)) \tag{3}$$

Let D be the formula:

$$(\neg(p_3 \rightarrow p_6)) \tag{4}$$

Evaluate the formulate E:

$$((A \wedge (B \wedge C)) \rightarrow D) \tag{5}$$

under the true assignment I_1, where $I_1(p_1) = I_1(p_3) = I_1(p_5) = false$ and $I_1(p2) = I_1(p_4) = I_1(p_6) = true$ as well as under the truth assignment I_2, where $I_2(p_1) = I_2(p_3) = I_2(p_5) = true$ and $I_2(p_2) = I_2(p_4) = I_2(p_6) = false$.

Your code should either read the input from a file or provide some form of interactive loop for entering the formulas. Provide 10 test cases for your code including the two examples of truth evaluation.

4 English to FOL (2pts)

Represent the following English sentences in First Order Logic:

- Every cruise ship was accompanied by at least one tug.
- At least one tanker was accompanied by more than one tug.
- All the fishing boats but one returned safely to port.
- There are exactly two students with grade less than B.

5 Matching (2pts)

In this exercise you are asked to implement matching which is a limited form of unification. We say that two formulas *match* if we can find substitutions for the variables appearing in the formulas such that the two are syntactically equivalent. You will need to write a function that determines whether a constant corresponding to a ground term such as Brother(George) and a pattern corresponding to a quantified formula such as Brother(x) match. If they do match the function returns a set of substitutions called *bindings* that map variables to terms. A constant matches another constant if they are equal. An unbound variable (one currently without a binding) matches any formula. A bound variable matches a constant if the constant and the value to which the variable is bound are equal.

You can choose whatever syntax you like for representing the formulas but make sure you document it clearly (for example variables should start with small-case letters, etc..). For example (in pseudocode):

```
match( Loves(Dog(Fred), Fred)
       Loves(x,y))
```

```
is true with x = Dog(Fred) and y = Fred

match( Loves(Dog(Fred), Fred)
       Loves(x,x)
fails
```

Your code should either read the input from a file or provide some form of interactive loop for entering the formulas. Provide 10 test cases for your code including the examples above.

6 Prolog (2pts)

Write a database of Prolog rules and facts to create sublists which contain any consecutive duplicates of elements. For example:

```
?- make_sub([a,a,a,a,b,c,c,a,a,t,e,e,e,e], X)
X = [[a,a,a,a],[b],[c,c],[a,a],[t],[e,e,e,e]]
```

You can use any implementation of Prolog.

7 Deliverables

Your deliverable is a report with your answers to the questions. For the questions requiring programming you must include the source code and test cases and provide enough documentation to make it easy to understand and read.