CSC 480 – Artificial Intelligence I – Fall 2022
Assignment 3

PART 1: Propositional Logic and Proofs

Solve the following problems and write out your reasoning using propositional logic. **For each step, record which facts you are operating on and which rules of inference you are using.** See the slides titled "Exercise: The Island of Knights & Knaves" for an example of how to formulate your answer.

Q1: On the Island of Knights and Knaves, you meet two people. A says, "We are the same kind." B says, "We are of different kinds." Who is what?

Q2: On the Island of Knights and Knaves, you meet two people. A says, "B is a knave." B says, "Neither A nor I are knaves." Who is what?

Q3: You meet inhabitants A, B, and C. You walk up to A and ask: "are you a knight or a knave?" A gives an answer but you don't hear what she said. B says: "A said she was a knave." C says: "don't believe B; he is lying." What are B and C? Can you tell something about A? **Explain your reasoning in detail.**

Q4: Suppose we have a knowledge base KB containing the following 4 sentences (we are using **~** as the negation symbol):

KB1: (A \/ ~C) => B
KB2: ~C \/ D
KB3: D => A
KB4: C

Using the above knowledge base and inference rules for propositional logic, derive (i.e. give a proof of) the sentence A /\ B. For each step indicate the rule applied and the sentences used to derive the new sentence.

PART 2: First Order Logic

We would like to find out more information about a very shady organization called the Society of Americans for Nepotism. We already know the following facts about this organization:

1. donny is a member.
2. connie is a member.
3. connie is related to eric.
4. Anyone related to a member is also a member.
5. Being related is symmetric (i.e., if X is related to Y, then Y is related to X)
6. alice is not a member.

Q5. Represent these facts as sentences in first-order predicate calculus (you can use **~** as the negation symbol).

Q6. Translate your sentences into CNF and then into clause form (show your work).

Q7. Use resolution-refutation to prove that **eric** is a member of the organization. At each step show the current sub-goal, the clause from the knowledge-based being applied, and the resolvent. Also show any substitutions being applied (See examples from the lecture slides for how to format your answer).

Q8. Use resolution-refutation to prove that there is someone to whom **donny** is not related**.**

Q9. How would the answer to Q8 change is the statement #6 in the above knowledgebase was changed to: "There is someone who is not a member" (first convert this sentence into clause form, then show specifically what changes in the refutation steps of Q8).

Q10. Consider a simple social network that allows member to **exchange** goods and services with other members under some conditions. A member can exchange with other members who are either in his/her circle or who are his/her friends. To be in someone's circle, you have to be a friend of someone linked to them in the network. This information, together with some facts about a few members are encoded in the following Horn knowledgebase:

1. **friend(ann,bob)**
2. **friend(kim,don)**
3. **linked(kim,bob)**
4. **linked(don,bob)**
5. **friend(*x*,*y*) => exchange(*x*,*y*)**
6. **in-cricle(*x*,*y*) => exchange(*x*,*y*)**
7. **linked(*z*,*y*) /\ friend(*x*,*z*) => in-cricle(*x*,*y*)**

where ***x***, ***y***, and ***z*** are variables and ann, bob, kim, and don are constants representing individual members. Using Generalized Modus Ponens, draw the backward chaining (AND/OR) proof tree for the query: **exchange(*x*,bob)**. Clearly indicate all of the answers for ***x*** (i.e. all the people who can exchange with **bob**).

PART 3: Exploring Prolog

In this part you will explore using the logic programming language Prolog. We will use the online interpreter for Prolog available at: https://www.swi-prolog.org/ (Of course, you can download and install a full version of SWI Prolog if you are interested in logic programming. You can also find good documentation on this Web site.) There is also a good online tutorial that covers the basics of Prolog: Learning Prolog Now.

Practice: Begin a new program on the online interpreter. Copy and paste this code into the program (left hand pane). Note that in Prolog, identifiers starting with upper case letters are assumed to be variables and those starting with lower case letters are constants (or are other atomic constructs such as numbers, etc.); function and predicate symbols also start with lower-case letters. All statements and queries, must end with a ".".

```
woman(mia).
loves(vincent,mia). loves(marsellus,mia).
loves(pumpkin,honey_bunny).
loves(honey_bunny,pumpkin).
jealous(X,Y):-  loves(X,Z),  loves(Y,Z).
```

Copy and paste these queries one at a time into the lower right hand pane (after the "?-" prompt). Record your results.

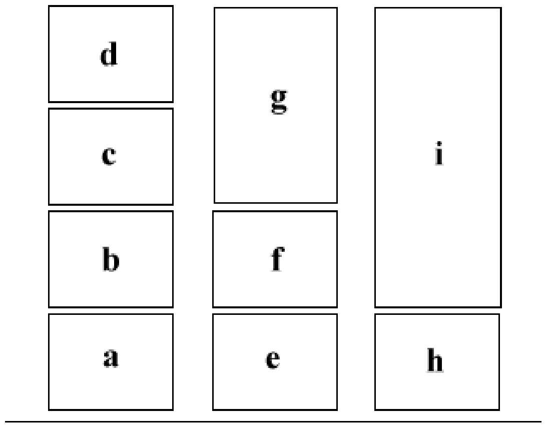
```
:-  woman(mia).

:-  woman(X).
:-  loves(vincent,X).
:-  loves(X,vincent).
:- loves(X,mia). % Note that in this case you will need to click the "NEXT"
                 % button to force backtracking and get more answers for X.
:-  jealous(marsellus,X). % Again watch for the "NEXT" button.
:-  jealous(honey_bunny,X).
```

In the above example, we had 5 facts and 1 rule. The facts are rather easy to figure out. TAKE NOTE: The rule "jealous(X,Y):- loves(X,Z), loves(Y,Z)." can be interpreted as "For all X and Y, X is jealous of Y if, for some Z, X loves Z and Y loves Z."

Q11. Convert the knowledgebase of **Q10** into a Prolog program and enter it as a new program in the SWI Prolog Web interface. As before, issue a query to see who can exchange with bob. Confirm your answers from **Q10** and record all the answers. Next Issue a query to determine who are all the people who can exchange with each other?

Q12. The following picture depicts three stacks of blocks on a table.



a. Represent in Prolog, the facts describing the relationships between the blocks. Use two predicates, **on** that says a block is directly on another block and **left** that says a block is immediately to the left of another (but not higher). For example, **f** and **g** are to the left of i; **e** is to the left of **h**, but not to the left of **i**.

   Now ask the following queries in the Prolog interpreter (**Provide your code, query and results**):

   - what blocks are on block **b**?
   - what blocks is block **a** on?
   - what blocks are on other blocks?
   - what blocks are on blocks that are immediately to the left of other blocks?

b. Define a new predicate **above** which is true when a block is anywhere in the same stack above another block. Also define a predicate **stackleft** which is true when a block is anywhere in a stack that is immediately to the left of the stack of another block. Now pose the following queries (**Provide your code, query and results)**:

   - what blocks are above other blocks?
   - what blocks are in a stack immediately to the left of stack for **f**?

Q13 [Extra Credit]: Lists in Prolog. Write the following Prolog predicates to add or delete an element to/from a list:

```
add2end(X, L, NewL)   % NewL is the result of adding a new
                      % element X to the end of the existing list L.

delend(L, X, NewL)    % X is the last element in existing list L
                      % and NewL is the remaining part of L without X.

add2front(X, L, NewL)  % NewL is the result of adding a new
                       % element X to the front of the existing list L.

delfront(L, X, NewL)  % X is the first (leftmost) element in existing
                      % list L and NewL is the remaining part of L
                      % without X.
```

Below are some examples:

```
?- add2end(5,[1,2,3,4],NL).
```

Answer: NL = [1, 2, 3, 4, 5]

```
?- delend([1,2,3,4], X, L).
```

Answer: **L** = [1, 2, 3],
X = 4

Demonstrate your program by testing each of the predicates using several queries. **Provide your code, query and results.**

Submit a single PDF document that contains **ONLY THE ANSWERS**. You do not need to repeat the questions. However, be sure to mark your answers clearly.