CSC520 Fall 2019 Assignment 3

Due October 15^{th} at 11:59pm

This assignment includes **conceptual and code** questions. It must be completed individually. You may not collaborate with other students, exchange partial answers, or use others' code. Questions about your work must be emailed to the instructor or TAs, or discussed during office hours.

You must submit your code and documentation as a single self-contained zip file called Assign3.zip. This file must contain all code necessary to run along with a README file that specifies how to build and execute the code. Your grade will be based upon the execution of your code and on its compilation, clarity, and documentation. Your answers to the logic questions should be uploaded as a single pdf file called Assign3.pdf. This will be graded on the completeness, correctness, coherence, and readability of your answers. As always, *show your work.*

The reference platform for the code execution is the CSC520_VCL image which is available via the NCSU VCL platform. Verify that your code will run on that platform *before submission* as it will be used for grading.

Question 1 (20 pts)

Consider the following English sentences.

1. Black tea is a type of tea and cheese is not.
2. Black tea can blend with Green Tea
3. If an element is a not a tea, then no tea can blend with it.
4. Some tea are unwilted and unoxidized.
5. Some tea can blend with all tea except the unoxidized ones.

a. (5 pts) Convert the sentences into first-order predicate logic. Use the following lexicon:

   tea (X) - X is a type of tea.
   unoxidize (X) - X has NOT been oxidized.
   wilt (X) - X has been wilted.
   blend (X, Y) - X can blend with Y.

b. (10 pts) Follow the steps on text book to convert the logic statements into CNF.

c. (5 pts) Using FOPL resolution prove the conclusion that Black tea cannot blend with cheese. Number your clauses, and indicate explicitly step-by-step what resolves together, and under what substitution.

Question 2 (15 pts)

Use the lexicon:
s – A customer can shop at Costco
o – A Costco customer
i – A customer has a membership identification with him/her
d – Customers will get membership identifications
g – A customer has gold membership
si – A customer has silver membership
pre – A customer can enjoy premium discount
l – A customer can collect loyalty points
c – Carol is a customer

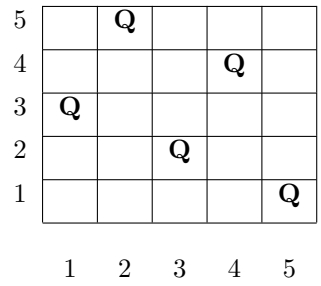
1. All Costco customers can collect loyalty points.
2. All Costco customers can get membership identifications.
3. Costco customers can shop at Costco stores only when they bring membership identifications with them.
4. Carol cannot shop at Costco stores.
5. Costco customers has either gold membership or Silver membership.
6. Carol is a Costco customer.
7. Carol has gold membership but doesn't get premium discount and cannot shop at Costco stores since she forgot to take her Costco identification.
8. Customers with gold memberships can have premium discount only if they have their identification with them while customers with silver memberships cannot.

a. (10 pts) Use propositional logic to determine if the specification is consistent. If the sentences are not consistent, use resolution to derive a contradiction. If they are consistent, use the truth-table method to show at least one model.

b. (5 pts) What if anything changes if Carol can shop at Costco stores and does the consistency change? Be specific in showing what's different.

Question 3 (65 pts)

This is a code question in which you are tasked with implementing a tool to solve arbitrary versions of the n-queens problem. The N-queens problem is a classical constraint problem where the goal is to determine where queens can be placed on an $n * n$ chessboard such that no two queens threaten the other. A sample image for this problem is shown below:



For this problem you will develop a python or java package that is called as follows:

```
NQueens.py ALG N CFile RFile
```
or
```
java -jar NQueens.jar ALG N CFile RFile
```

where: ALG is one of FOR or MAC representing Backtracking search with forward checking or Maintaining Arc Consistency respectively. N represents the number of rows and columns in the chessboard as well as the number of queens to be assigned. CFile is an output filename for your constraint problem. And RFile is an output file for your results.

When called your code must generate and maintain an internal representation for each problem as an instance of an object called QueenGraph and use that to calculate as many unique solutions for the problem as it can find up to up to 2*N using the specified algorithm. At the start of the execution your code must write a *human-readable* representation of the variables, domains, and constraints out to CFile. And as each solution is found it must write it in human-readable form out to RFile. When the code completes it should report the number of solutions found, the real time taken, and the number of backtracking steps. As always your code should be clear, readable, and well-written.