Computational Logic - CS 6374

Final Exam

1. Maximum points 150.

2 There are 6 problems in this exam. For programming problems, please include:
   a. A brief explanation of your solution strategy and your code.
   b. The Prolog/CLP(FD)/ASP code (along with comments briefly explaining each predicate).
   c. Screenshot pictures cut-pasted into the word document showing the execution run of 3 representative test-cases.
3. You can put all your answers in an MS Word Document. Answer to each problem must start from a new page.
4. Please put your name on the first page of the MS Word Document
5. Submit the entire MS Word document containing your solutions on eLearning as a single file.
6. No collaboration is allowed, all work has to be your own. You can look up your homework solution code, your notes/class lecture videos, your textbook, and resources provided on the course page. Please do not consult any other source. You may be asked to explain your solution during the grading of the exam. You are honor bound to not use any unfair means to program the solutions. Deadline to submit your exam solution: Midnight Wednesday, May 13th.

Pledge: I have neither given nor received any outside help during this exam. I have not collaborated with any one else or improperly used any online resources to find or solicit answers to the questions in this exam. I have not used any electronic devices to communicate with anyone about this exam.

SIGNED: _____

NAME: _____

DATE: _____

Location where the exam was taken: _____

DO NOT WRITE BELOW THIS SPACE

Question 1: 20 pts

Write a CLP(FD) program for solving the graph coloring problem. The graph is assumed to be undirected. Assume you have 4 colors (green, red, yellow, blue). In the graph coloring problem, we have to find an assignment of a color to each node so that two nodes connected by an edge do not have the same color. Your program will take a list of nodes each described as a single number and a list of edges denoted by a pair of numbers and will output a coloring assignment for the nodes. The four colors are hard wired into the program. An example query is shown below:

```
?- graphcolor([1,2,3,4,5],[(1,2),(1,3),(2,4),(2,5),(3,4),(3,5)], A).
A = [(1,r),(2,b),(3,y),(4,r),(5,g)]
```

Remember, you have to write the code in CLP(FD).

Question 2: 25 pts

Consider the following Prolog program.

```
sub([],X).
sub([X|L], L1) :- sff([X|L2], L1), sub(L, L2).

sff(L, L).
sff(L1, [X|L]) :- sff(L1, L).

comm(X,Y) :- sub([X,Y] ,[a,t,s]), sub([X,Y] ,[a,s,t]).
```

a. What does the predicate `sff` do?

b. Show the search tree for query: `?- sub([X,Y], [a,t,s]).`

c. Show the search tree for query `?- sub([X,Y], [a,s,t]).`

Note: If you can guess the values for `X` and `Y` based on your solution to part `b`, then you can simply write those values without drawing the search tree. Do justify your answer.

d. What will be the value(s) of `A` and `B` in the following query:

`?- comm(A,B).`

Question 3: 24 pts

Define the following terms:

1. Definite Clause Grammar

2. Coinductive Logic Programming

3. Gelfond Lifschitz Transform

Question 4: 36 pts

Given a directed graph, write a s(CASP) answer set program to find nodes on which no edges are incident. That is, such nodes can have edges coming out of them, but they cannot have any nodes coming into them. You must use negation as failure to model the concept that there is no incident edge. The graph is described as in Question 1. However, now the graph is directed, so an edge denoted by the pair (1, 2) means that there is a directed edge emanating from node `1` that is incident on node `2`. Example queries are shown below (for simplicity, the partial answer set s(CASP) will produce are not shown):

```
?- no_edge_in([1, 2, 3], [(1,2),(1,3)], N).

N=1;

no

?- no_edge_in([1,2,3,4,5,6],[(1,2),(1,3),(2,4),(2,5),(3,4),(3,5)], N).

N=1;

N=6;

no
```

Question 5: 15 pts (This question has 3 parts)

Question 5a: Define Abduction.

Question 5b: Consider the following ASP program:

```
p :- q, not r.
p :- q, not s, not t.
s :- not p.
r :- not p.
v :- not u.
```

Compute the dual rules for `p`, `s` and `r`.

Question 5c: Consider the program in 5b. Suppose the set of abducibles, `E`, is: `E = {q, t, u, v}`

Show how the query `?- not s.` will be executed on s(CASP) in abductive mode. What abducibles will be computed by this query? Note that these abducibles come from `E`.

Hint: For each abducible `a` $\in$ `E` , add the even loop:

```
a :- not not_a.
not_a :- not a.
```

then execute the query for the program extended with these even loops.

Question 6: 30 pts

This problem consists of modeling and solving the Lights Out puzzle (you can learn more details at https://www.logicgamesonline.com/lightsout/). The domain is a square grid of n x n cells, each one with a "light" that can be on (white cell) or off (black cell). The only possible action is touching (or clicking) on a given cell and the effect of that action is flipping the state of the light in that cell and *all* its adjacent ones. In other words, we also flip those cells that share a side (horizontally or vertically) with the clicked cell. The goal of the game is reaching a state in which all lights are off (lights out!).

The grid could be encoded as a list of lists. For example, the grid below (0 means light off in that cell, 1 means light on in that cell)

00001

11111

00010

01110

01001

is encoded as [[0,0,0,0,1],[1,1,1,1,1],[0,0,0,1,0],[0,1,1,1,0],[0,1,0,0,1]]

You can view the solution to the puzzle as a planning problem. Write a logic program to solve the lights out puzzle. Given an input configuration of lights, your program should produce a sequence of cells that should be touched to make the lights go out (you can describe a cell by its coordinates, i.e., (x,y) pair).

You should write your program in one of the following languages: Prolog, CLP(FD), or ASP. For ASP, you can use CLINGO or s(CASP), however, if you use CLINGO, you will have to use a different input format as CLINGO does not support lists.

Please describe the structure of your code, and do not forget to put comments describing each of your predicate definitions.