PART 1

1. Reach Chapters 4 and 5 in Gelfond and Kahl's book.

PART 2

2. Review the "Intro to ASP and Practice Problems" document on the course page to practice with ASP.

PART 3

3. Solve Problems #1, #3, #5, #7, #9 from Chapter 5 in Gelfond and Kahl's book. All problems to be programmed and run on CLASP (download CLASP from https://potassco.org/; you will have to ground your program first using the GRINGO system, then find answer sets using the CLASP system).

We can use the application as:
gringo [options | files] | **clasp** [options | **number**#] OR
clingo [options | files | **number**#] (***Only produces first answer set? NO**#)

Intentionally didn't hide the grounded rules.

1. "Apollo and Helios are lions in a zoo. Normally lions are dangerous. Baby lions are not dangerous. Helios is a baby lion." Assume that the zoo has a complete list of baby lions that it maintains regularly. Your program should be able to deduce that Apollo is dangerous, whereas Helios is not. Make sure that

(a) if you add another baby lion to knowledge base, the program would derive that it is not dangerous, even though that knowledge is not explicit; and

(b) if you add an explicit fact that Apollo is not dangerous, there is no contradiction and the program answers intelligently.

Ans.

Output (p3q1simple.log):
clasp version 3.1.4
Reading from stdin
Solving...
Answer: 1
lion(apollo) lion(helios)
baby_lion(helios) dangerous(apollo)
-dangerous(helios)
SATISFIABLE

Models : 1
Calls : 1
Time : 0.000s (Solving: 0.00s 1st Model: 0.00s Unsat: 0.00s)
CPU Time : 0.000s

Image 01: Simple Run

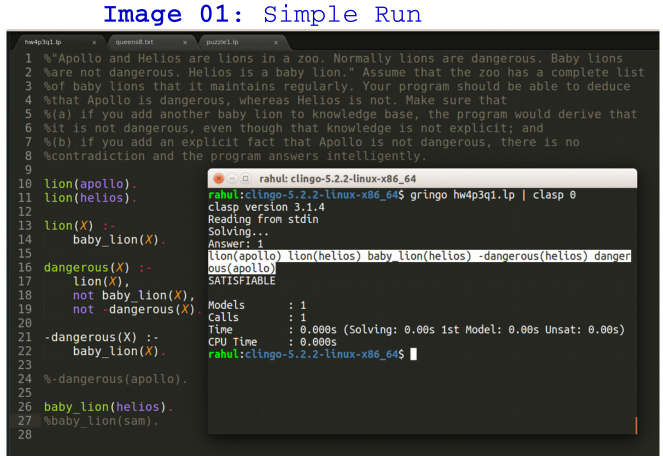
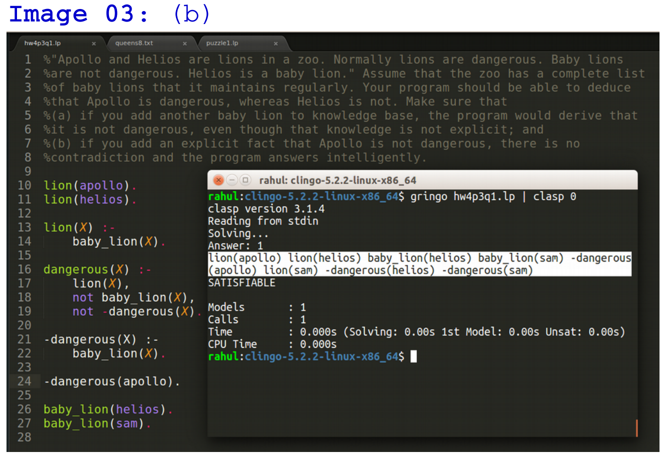


Image 02: (a)

Image 03: (b)



3. "American citizens normally live in United States. American diplomats may or may not live in the United States. John, Miriam, and Caleb are American citizens. John lives in Italy. Miriam is an American Diplomat."

a. Assume we do *not* have complete list of American diplomats. (Note that your program should not be able to conclude that Caleb lives in United States.)

b. Now assume we have a *complete* list of American diplomats. Add this information to the program. What does your new program say about Caleb's place of residence?

c. Rewrite the program from 3b by using the simplified form of the cancellation axiom.

Ans.

Output (p3q3a.log): ...

```
Answer: 1
american_diplomat(miriam) live_in(john,italy) country(united_states) country(italy)
american_citizen(john) american_citizen(miriam) american_citizen(caleb)
ab_citizen(john) ab_citizen(caleb) live_in(miriam,united_states)
-live_in(miriam,italy) -live_in(john,united_states)
Answer: 2
american_diplomat(miriam) live_in(john,italy) country(united_states) country(italy)
american_citizen(john) american_citizen(miriam) american_citizen(caleb)
ab_citizen(john) ab_citizen(caleb) -live_in(miriam,united_states)
-live_in(john,united_states)
SATISFIABLE
...
```

Output (p3q3b.log): ...

```
Answer: 1
american_diplomat(miriam) live_in(john,italy) country(united_states) country(italy)
american_citizen(john) american_citizen(miriam) american_citizen(caleb)
-american_diplomat(john) -american_diplomat(caleb) -live_in(john,united_states)
live_in(caleb,united_states) -live_in(miriam,united_states) -live_in(caleb,italy)
Answer: 2
american_diplomat(miriam) live_in(john,italy) country(united_states) country(italy)
american_citizen(john) american_citizen(miriam) american_citizen(caleb)
-american_diplomat(john) -american_diplomat(caleb) -live_in(john,united_states)
live_in(caleb,united_states) live_in(miriam,united_states) -live_in(caleb,italy)
-live_in(miriam,italy)
SATISFIABLE
...
```

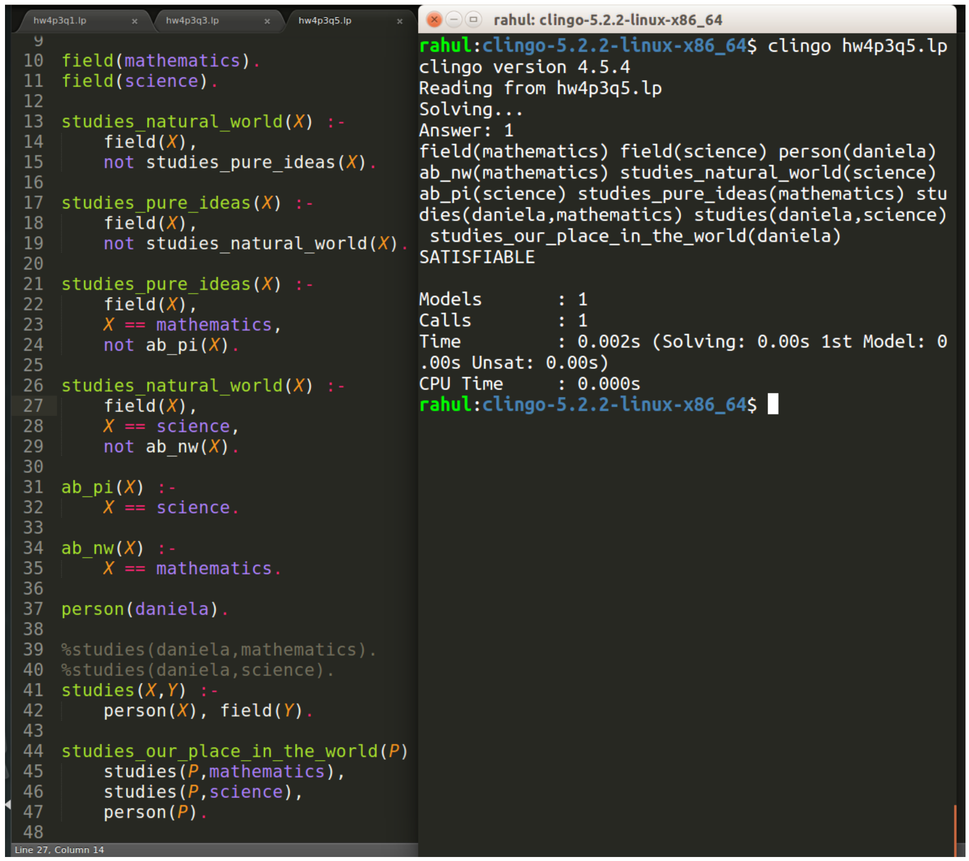
Output (p3q3c.log): ...

```
Answer: 1
american_diplomat(miriam) live_in(john,italy) country(united_states) country(italy)
american_citizen(john) american_citizen(miriam) american_citizen(caleb)
-american_diplomat(john) -american_diplomat(caleb) ab_citizen(miriam)
-live_in(john,united_states) live_in(caleb,united_states)
-live_in(miriam,united_states) -live_in(caleb,italy)
Answer: 2
american_diplomat(miriam) live_in(john,italy) country(united_states) country(italy)
american_citizen(john) american_citizen(miriam) american_citizen(caleb)
-american_diplomat(john) -american_diplomat(caleb) ab_citizen(miriam)
-live_in(john,united_states) live_in(caleb,united_states)
live_in(miriam,united_states) -live_in(caleb,italy) -live_in(miriam,italy)
SATISFIABLE
...
```

5. "A field that studies pure ideas does not study the natural world. A field that studies the natural world does not study the pure ideas. Mathematics normally studies pure ideas. Science normally studies the natural world. As a computer scientist, Daniela studies both mathematics and science. Both mathematics and science study our place in the world."

Make sure your program can deduce that Daniela studies our place in the world.

Ans.



Output (p3q5.log):

```
clasp version 3.1.4
Reading from stdin
Solving...
Answer: 1
field(mathematics) field(science) person(daniela) ab_nw(mathematics)
studies_natural_world(science) ab_pi(science) studies_pure_ideas(mathematics)
studies(daniela,mathematics) studies(daniela,science)
studies_our_place_in_the_world(daniela)
SATISFIABLE

Models       : 1
Calls        : 1
Time         : 0.000s (Solving: 0.00s 1st Model: 0.00s Unsat: 0.00s)
CPU Time     : 0.000s
```

7. You are given three complete lists of facts of the form
course(math), course(graphs), ...
student(john), student(mary), ...
took(john,math), took(mary,graphs), ...

Students can graduate only if they have taken all the courses in the first list. Write a program that, given the above information, determines which students can graduate. Make sure that, given the following sample knowledge base,
student(john).
student(mary).
course(graphs).
course(math).
took(john,math).
took(john,graphs).
took(mary,graphs).

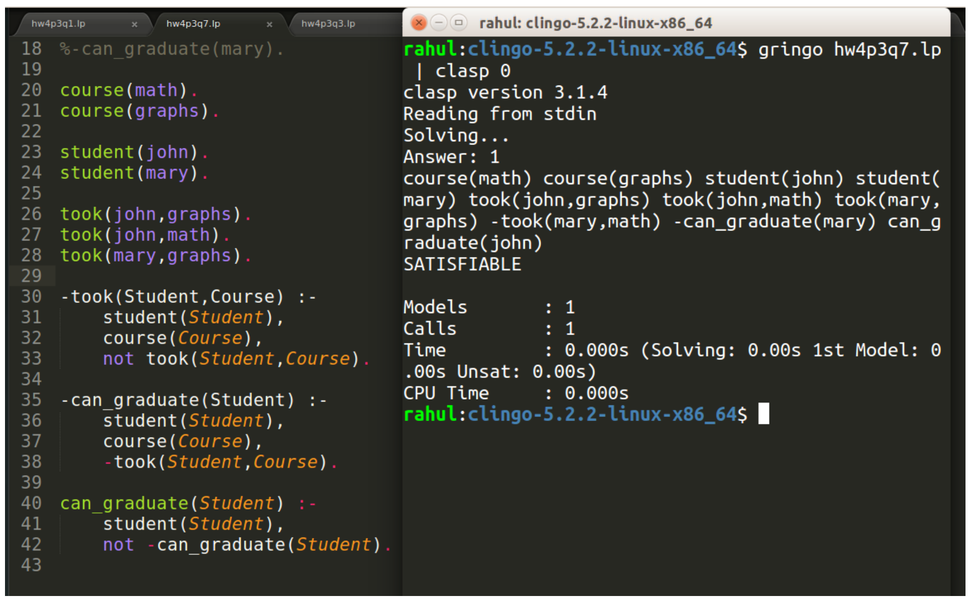
your program is able to

conclude can_graduate(john).
~can_graduate(mary).

Ans.

Output (p3q7.log):

```
clasp version 3.1.4
Reading from stdin
Solving...
Answer: 1
course(math)
course(graphs)
student(john)
student(mary)
took(john,graphs)
took(john,math)
took(mary,graphs)
-took(mary,math)
-can_graduate(mary)
can_graduate(john)
SATISFIABLE

Models       : 1
Calls        : 1
Time         : 0.000s (Solving: 0.00s 1st Model: 0.00s Unsat: 0.00s)
CPU Time     : 0.000s
```

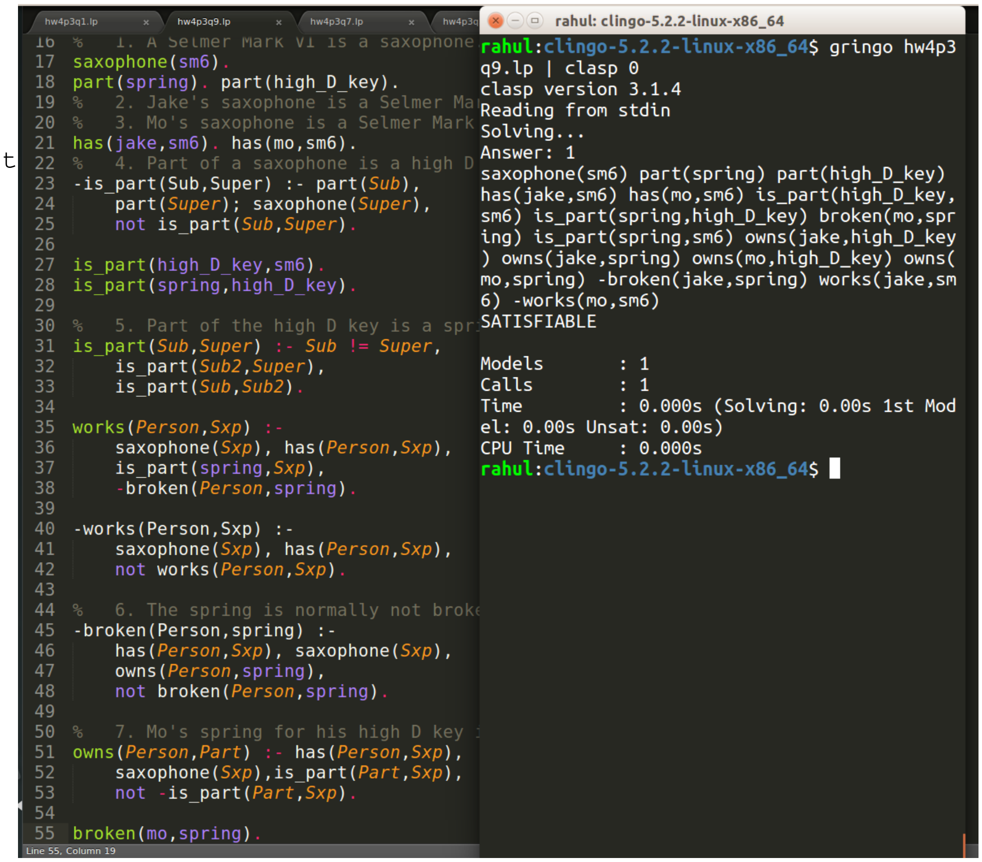


9. Using the notions of hierarchy and defaults as detailed in Section 5.4, write an ASP program to represent the following information. Be as general as you can.

1. A Selmer Mark VI is a saxophone.
2. Jake's saxophone is a Selmer Mark VI.
3. Mo's saxophone is a Selmer Mark VI.
4. Part of a saxophone is a high D key.
5. Part of the high D key is a spring that makes it work.
6. The spring is normally not broken.
7. Mo's spring for his high D key is broken.

Make sure that your program correctly entails that Jake's saxophone works while Mo's is broken. For simplicity, assume that no one has more than one saxophone, and hence, saxophones can be identified by the name of their owner.

Ans.



Output (p3q9.log):

clasp version 3.1.4
Reading from stdin
Solving...
Answer: 1
saxophone(sm6) part(spring) part(high_D_key) has(jake,sm6) has(mo,sm6) is_part(high_D_key,sm6) is_part(spring,high_D_key) broken(mo,spring) is_part(spring,sm6) owns(jake,high_D_key) owns(jake,spring) owns(mo,high_D_key) owns(mo,spring) -broken(jake,spring) works(jake,sm6) -works(mo,sm6)
SATISFIABLE

```
Models       : 1
Calls        : 1
Time         : 0.000s (Solving: 0.00s 1st Model: 0.00s Unsat: 0.00s)
CPU Time     : 0.000s
```

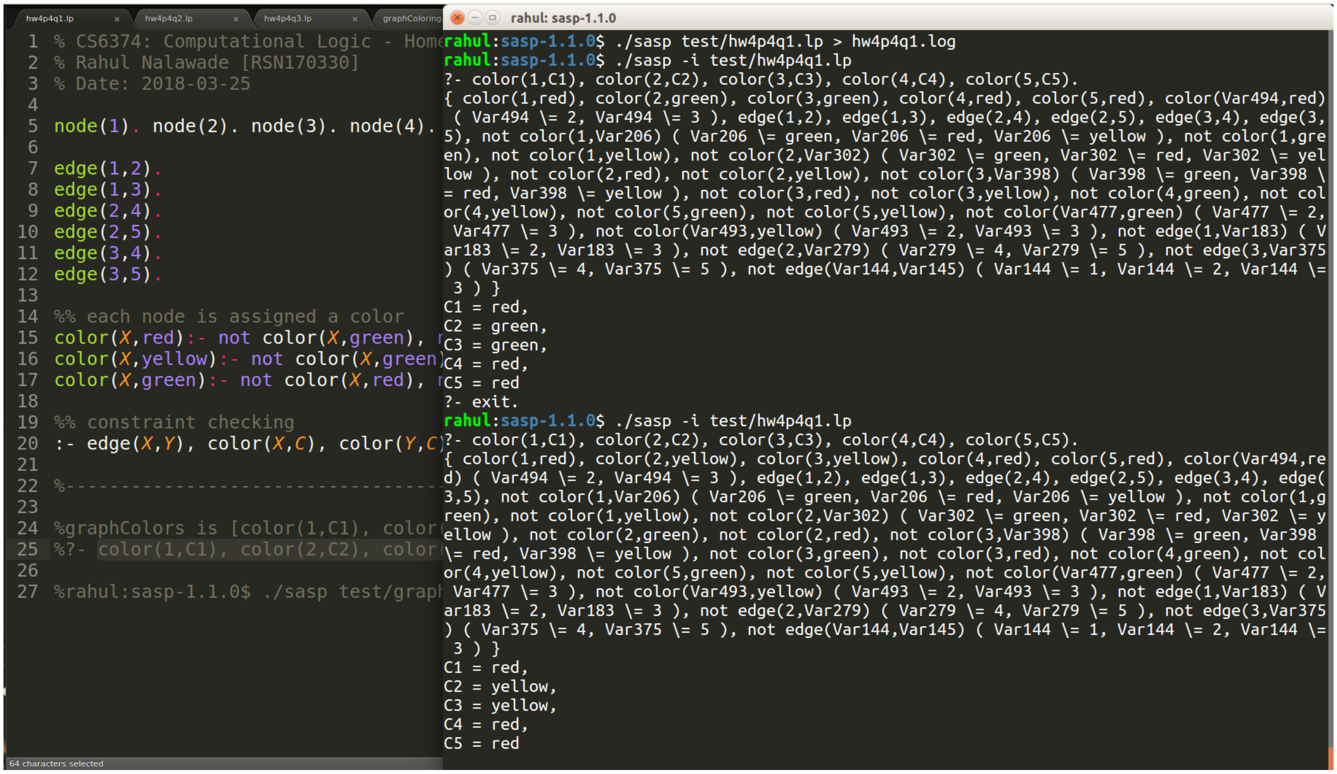
PART 4

4. Solve the following problems using the s(ASP) system; recode them in ASP, not Prolog:

a. Program the graph coloring problem in ASP (use the code discussed in class)

Ans.

Source code: hw4p4q1.lp **Output file: hw4p4q1.log**



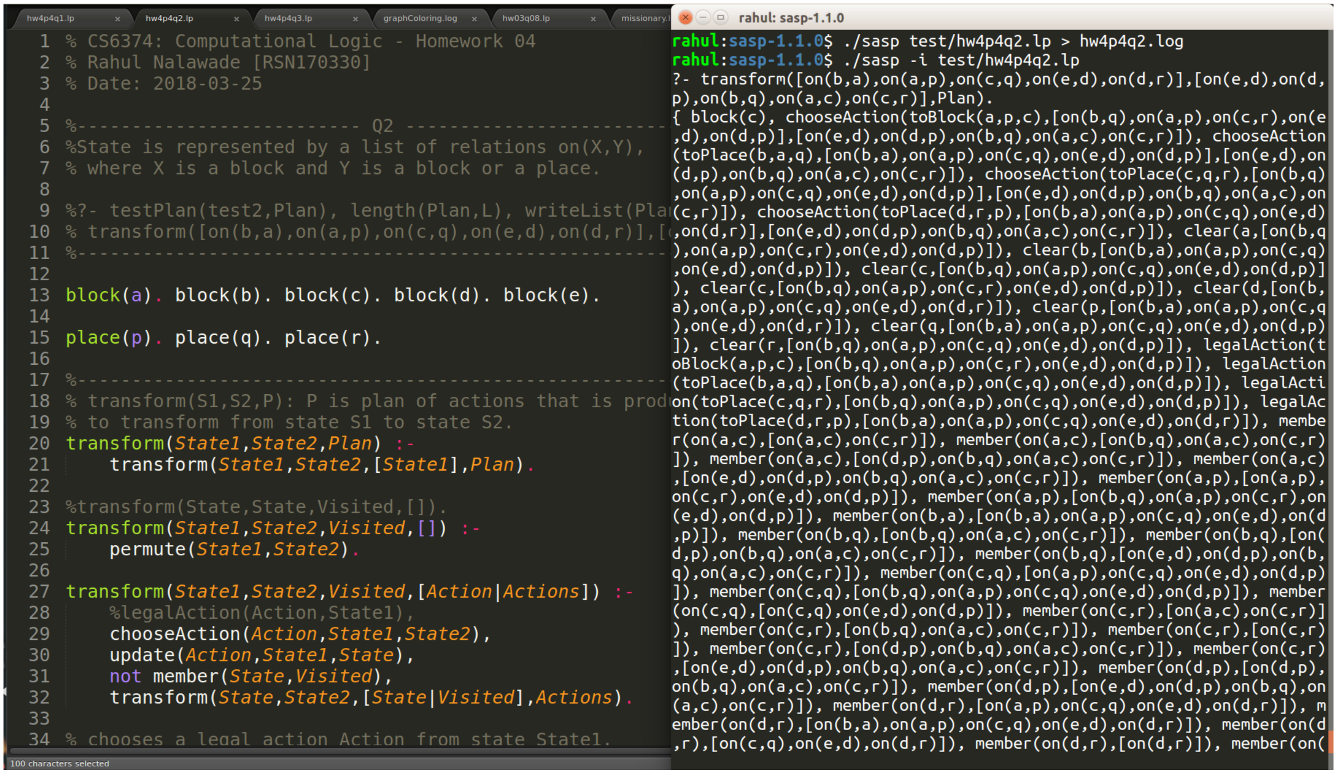
```
% CS6374: Computational Logic - Home
% Rahul Nalawade [RSN170330]
% Date: 2018-03-25

node(1). node(2). node(3). node(4).

edge(1,2).
edge(1,3).
edge(2,4).
edge(2,5).
edge(3,4).
edge(3,5).

%% each node is assigned a color
color(X,red):- not color(X,green), r
color(X,yellow):- not color(X,green)
color(X,green):- not color(X,red), r

%% constraint checking
:- edge(X,Y), color(X,C), color(Y,C)

%-----------------------------------

%graphColors is [color(1,C1), color(
%?- color(1,C1), color(2,C2), color(

%rahul:sasp-1.1.0$ ./sasp test/graph
```

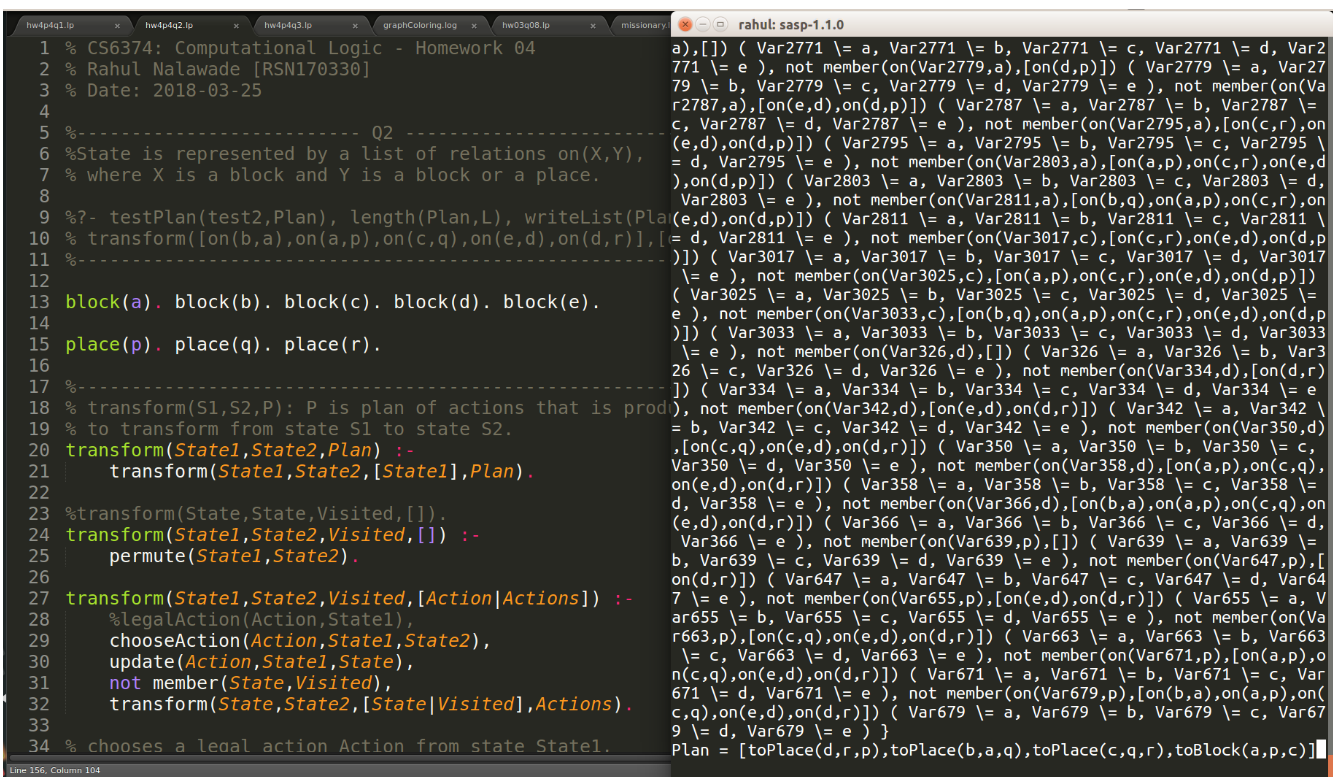
```
rahul:sasp-1.1.0$ ./sasp test/hw4p4q1.lp > hw4p4q1.log
rahul:sasp-1.1.0$ ./sasp -i test/hw4p4q1.lp
?- color(1,C1), color(2,C2), color(3,C3), color(4,C4), color(5,C5).
{ color(1,red), color(2,green), color(3,green), color(4,red), color(5,red), color(Var494,red)
 ( Var494 \= 2, Var494 \= 3 ), edge(1,2), edge(1,3), edge(2,4), edge(2,5), edge(3,4), edge(3,
5), not color(1,Var206) ( Var206 \= green, Var206 \= red, Var206 \= yellow ), not color(1,gre
en), not color(1,yellow), not color(2,Var302) ( Var302 \= green, Var302 \= red, Var302 \= yel
low ), not color(2,red), not color(2,yellow), not color(3,Var398) ( Var398 \= green, Var398 \
= red, Var398 \= yellow ), not color(3,red), not color(3,yellow), not color(4,green), not col
or(4,yellow), not color(5,green), not color(5,yellow), not color(Var477,green) ( Var477 \= 2,
 Var477 \= 3 ), not color(Var493,yellow) ( Var493 \= 2, Var493 \= 3 ), not edge(1,Var183) ( V
ar183 \= 2, Var183 \= 3 ), not edge(2,Var279) ( Var279 \= 4, Var279 \= 5 ), not edge(3,Var375
) ( Var375 \= 4, Var375 \= 5 ), not edge(Var144,Var145) ( Var144 \= 1, Var144 \= 2, Var144 \=
 3 ) }
C1 = red,
C2 = green,
C3 = green,
C4 = red,
C5 = red
?- exit.
rahul:sasp-1.1.0$ ./sasp -i test/hw4p4q1.lp
?- color(1,C1), color(2,C2), color(3,C3), color(4,C4), color(5,C5).
{ color(1,red), color(2,yellow), color(3,yellow), color(4,red), color(5,red), color(Var494,re
d) ( Var494 \= 2, Var494 \= 3 ), edge(1,2), edge(1,3), edge(2,4), edge(2,5), edge(3,4), edge(
3,5), not color(1,Var206) ( Var206 \= green, Var206 \= red, Var206 \= yellow ), not color(1,g
reen), not color(1,yellow), not color(2,Var302) ( Var302 \= green, Var302 \= red, Var302 \= y
ellow ), not color(2,green), not color(2,red), not color(3,Var398) ( Var398 \= green, Var398
\= red, Var398 \= yellow ), not color(3,green), not color(3,red), not color(4,green), not col
or(4,yellow), not color(5,green), not color(5,yellow), not color(Var477,green) ( Var477 \= 2,
 Var477 \= 3 ), not color(Var493,yellow) ( Var493 \= 2, Var493 \= 3 ), not edge(1,Var183) ( V
ar183 \= 2, Var183 \= 3 ), not edge(2,Var279) ( Var279 \= 4, Var279 \= 5 ), not edge(3,Var375
) ( Var375 \= 4, Var375 \= 5 ), not edge(Var144,Var145) ( Var144 \= 1, Var144 \= 2, Var144 \=
 3 ) }
C1 = red,
C2 = yellow,
C3 = yellow,
C4 = red,
C5 = red
```

b. Program the block's world problem from HW 7 using ASP

Ans.

Source code: hw4p4q2.lp **Output file: hw4p4q2.log**





c. Program the Missionary-Cannibal problem

Ans.

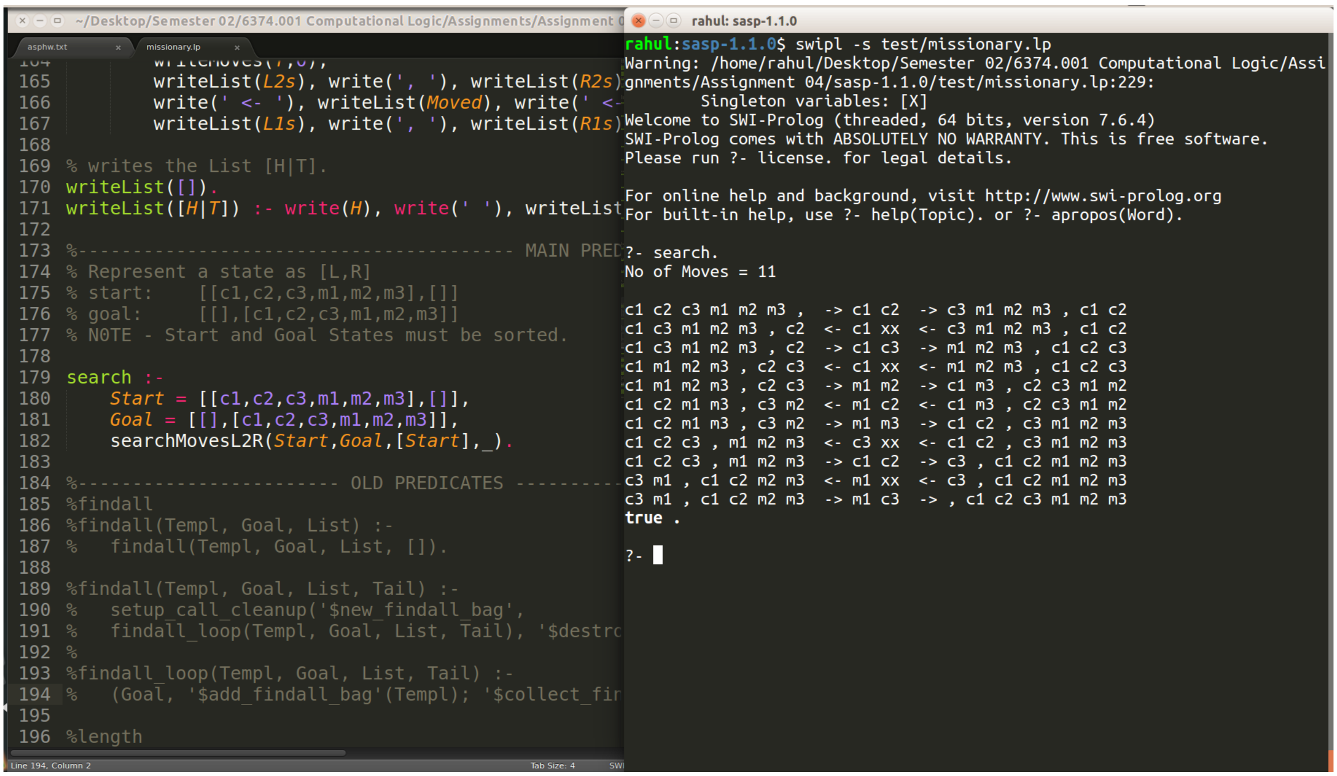
Source code: hw4p4q3.lp **Output file: hw4p4q3.log**

First, I tried using my previous code (**missionary.lp**) in here, but due to built-ins like **findall**, I couldn't convert it into s(ASP). So, I referred a code for missionary cannibal (link mentioned in the above file) for solving this problem.

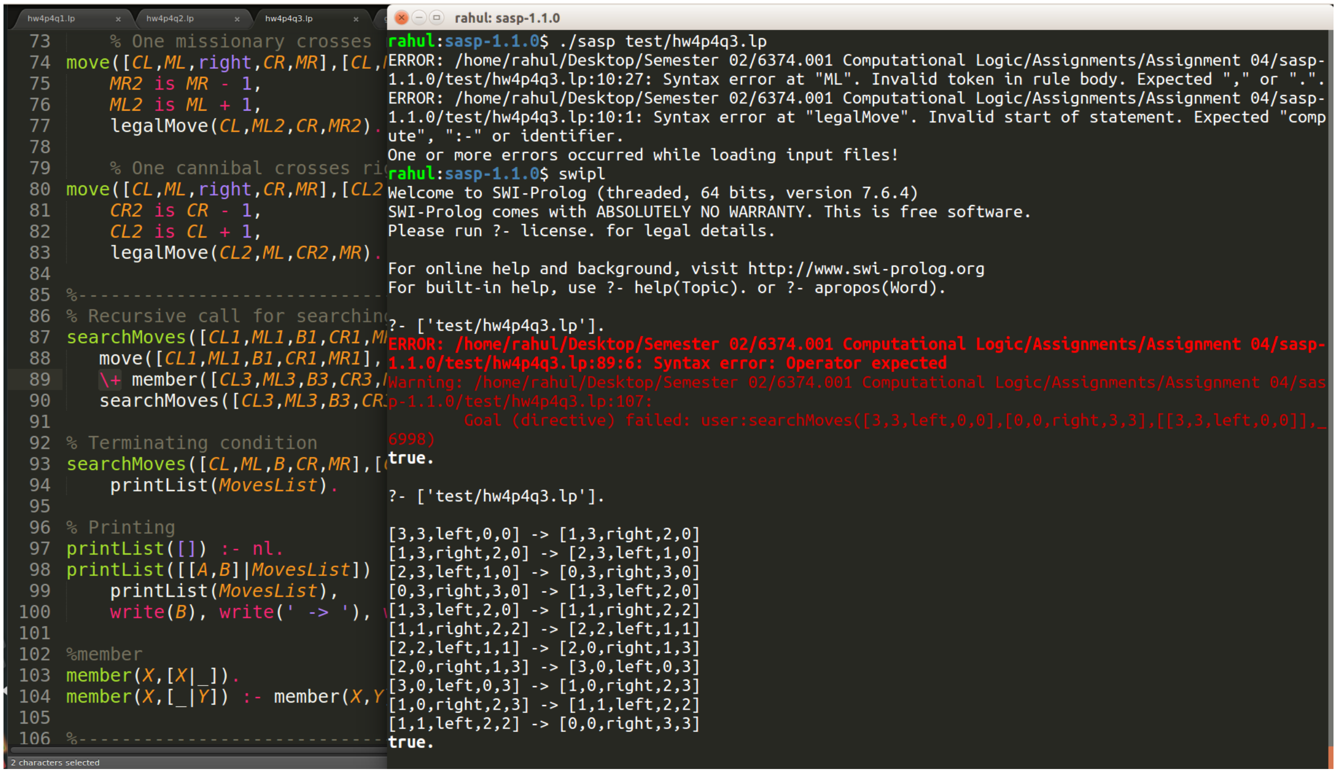
But again, I got only two errors here and I tried rectifying these simple errors, but couldn't find anything to do. I don't know why inequality is not working here. I've had referred the s(ASP) manual but couldn't find there any also.

Here is the attempt:

missionary.lp



hw4p4q3.lp



Note: Please refer all source codes and output (.log) files which are attached with this file.