Saishnu Ramesh Kumar (300758706)

CSCI 117 – Lab 10

Part 1:

1) Eight Queens Example:

For this eight-queen puzzle, the main objective is to place 8 queens on an 8 by 8 chessboard but, none of the queens are under attack from each other. Starting off with the example where N = 4, this would use a 4 by 4 board. The first query, "n_queens (4, Qs), label (Qs)", is where label 1 returns the values. The first argument would be a positive integer, and the second argument would be the length list of the first argument. The three loops from safe_queens([Q|Qs], Q0, D0) affect the constraints of the column values of the queens Q2 to Q7, where 4 queens and Q0 = 1, Q1 = 5, and Q2 = 9. Since in this case it's placed in a 4 by 4 board, each queen is assigned to each of the columns, and it is impossible to place two queens in the same column together. We would need to figure out which of the rows are suitable to place each queen. For Q0 = 1, we can place the first queen on the third row. When running the loops three times, they display the possible positions the queens are placed in the space for Q2 to Q7. The 1st loop is (Q2 = 4, Q3 = 3, Q4 = 2, Q5 = 1), the 2nd loop is (Q6 = 3, Q7 = 4, Q8 = 1, Q9 = 2), and the third loop is (Q10 = 1, Q11 = 2, Q12 = 3, Q13 = 4). In total there are 12 constraints, and they are determined by how many variables and set of values are chosen when the queens are decided to be placed in different rows. The role of the third argument to the safe_queens would be another list of constraints for the loop.

2) Sudoku Example:

```
1) sudoku(Rows) :-
2)         length(Rows, 9), maplist(same_length(Rows), Rows), %1
3)         append(Rows, Vs), Vs ins 1..9, %2
4)         maplist(all_distinct, Rows), %3
5)         transpose(Rows, Columns), %4
6)         maplist(all_distinct, Columns), %5
7)         Rows = [As,Bs,Cs,Ds,Es,Fs,Gs,Hs,Is], %6
8)         blocks(As, Bs, Cs), %7
9)         blocks(Ds, Es, Fs),
10)          blocks(Gs, Hs, Is).
11)
12)  blocks([], [], []).
13)  blocks([N1,N2,N3|Ns1], [N4,N5,N6|Ns2], [N7,N8,N9|Ns3]) :-
14)          all_distinct([N1,N2,N3,N4,N5,N6,N7,N8,N9]),
15)          blocks(Ns1, Ns2, Ns3).
16)
17)  problem(1, [[_,_,_,_,_,_,_,_,_], %8
18)              [_,_,_,_,_,3,_,8,5],
19)              [_,_,1,_,2,_,_,_,_],
20)              [_,_,_,5,_,7,_,_,_],
21)              [_,_,4,_,_,_,1,_,_],
22)              [_,9,_,_,_,_,_,_,_],
23)              [5,_,_,_,_,_,_,7,3],
24)              [_,_,2,_,1,_,_,_,_],
```

```
25)                [_,_,_,_,4,_,_,_,9]]).
```

1) In the first line, the length is defined to basically create a 9 by 9 square for the sudoku puzzle. It also takes in 9 rows and maps them.

2) The appends would insert the variables with the range of possible values into the list. In VS, it can be seen that cases 1...9 are included.

3) Using maplist, we can call all_distinct to make sure that all the numbers inputted inside the list are from the library.

4) We then transpose the rows and columns of the puzzle.

5) Once again in maplist, we call all_distinct to ensure that all the numbers only occur once in each column.

6) For rows, it will convert Rows in the n*n block to check if they are in their own self-distinct blocks.

7) The blocks are where the variables are contained, and each domain has only one concrete value. Moreover, the predicate block is true if it receives three empty lists as its input. If the blocks/3 is non-empty, Prolog will use the pipe to split up the rows and store the rest for later.

8) This is where the values are then inserted into the 9 by 9 table for the sudoku puzzle following all the restrictions set by the program.

3) Knights and Knaves Examples:

```
% Example 6: B says: "I am a knight, but A isn't."
%            A says: "At least one of us is a knave."


example_knights(6, [A,B,C]):-
        sat(B=:=(~B + A)).
        sat(B=:= ~A),
        sat(B=:=(C=:=A)).
```

Output: A = B, B = 1

% Example 7: B says: “Either one of us is a knight.”

```
example_knights(7, [A,B]) :-
        sat(B=:=card([2,3],[~A,~B])).
```

Output: A = 1, B = 0

% Example 8: C says: “We are all knaves.”

A says: “Only one of us is a knight.”

```
example_knights(8, Ks) :-
            Ks = [A,B,C],
        sat(C=:=(~A * ~B * ~C)),
            sat(A=:=card([1],Ks)).
```

Output: Ks = [1,0,0]

Part 2:

```
:- use_module(library(clpfd)).


% Base case: Reverse of empty is empty

reverse([],[]).

reverse([H|T],RevList):-

%concatenation

reverse(T,RevT),append(RevT,[H],RevList).


Helper([],[],C,|T3|):-T3 #= C.

Helper([H1|T1],[H2|T2],Carry2,[H3|T3]):-

        H3 #= (H1+H2+Carry2)div 10,

        Helper(T1,T2,Carry,T3).


crypt1([H1|L1],[H2|L2],[H3|L3],L4) :-

    % Give constraints on variable values in L4

L4 ins 0..9,

    % Heads cannot have value 0

H1 #/= 0, H2 #/= 0

    % Variable values are distinct in L4

all_different(L4),

    % Reverse the 3 input words

reverse([H1|L1], Out1), reverse([H2|L2], Out2), reverse(H3|L3], Out3),

    % Call to helper function that does the sum with reversed words one iteration
at a time

Helper(Out1,Out2,0,Out3).
```

Query:

```
crypt1([S,E,N,D],[M,O,R,E],[M,O,N,E,Y],[D,N,E,S,R,O,M,Y]), labeling([ff],[D,N,E,S,
R,O,M,Y]).


Output:
```

```
D = 7

E = 5

M = 1

N = 6

O = 0

R = 8

S = 9

Y = 2
```