Jewe: Bejeweled clone puzzle game

For

13016223 Artificial Intelligence

Napat Traikityanukul	59090011
Poonyawee Visetputtasart	59090023
Shouh Yann Mo	59090029

23 December 2018

International College,

King Mongkut's Institute of Technology Ladkrabang

Table of Contents

Introduction

JEWE

Overview

Jewe is a classic tile-matching puzzle game which is inspired by Bejeweled. The objective of Jewe is to swap a gem with an adjacent gem to form a horizontal or vertical chain of three or more gems of the same type and same color. Gems disappear when chains are formed, and gems fall from the top to fill in gaps.

Jewe is based on Gemgem, a Bejeweled clone written in Python with the Pygame library. Gemgem is an open source project from Invent with Python and source code is available to download for free.

Sometimes the player might get stuck or don't know where they should move the gems, a hint button is implemented by using various AI techniques.

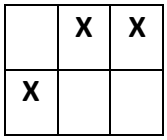
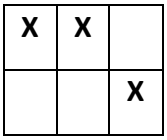
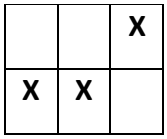
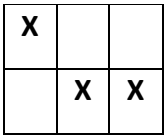
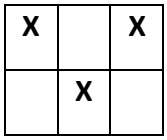
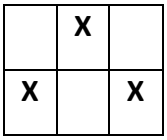
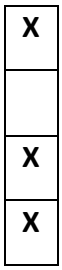
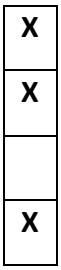
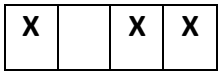
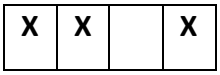
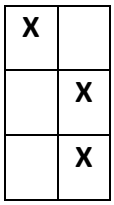
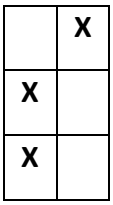
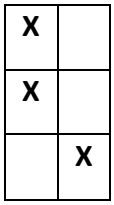
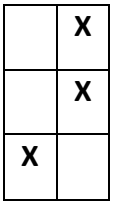
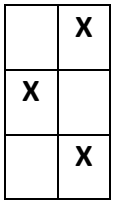
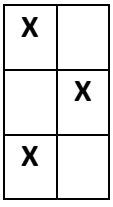
Figure 1 Sample Screenshot of Bejeweled

Purpose:

- The player can challenge themselves by trying to achieve as much score as possible.
- The player can play just for fun or relaxing.
- The player can improve their cognitive skill and their reflexes.

AI techniques/methods used in the project

There are 16 possible ways for the gems to be one move away from forming a triple.

1. 	2. 
3. 	4. 
5. 	6. 
7. 	8. 
9. 	10. 
11. 	12. 
13. 	14. 
15. 	16. 

In our project, we use **constraint logic programming over finite domains** (also known as CLP(FD).) along with pattern matching.

clp(fd) is a library included in the standard SWI-Prolog distribution. It solves problems that involve sets of variables, where relationships among the variables need satisfied.

clp(fd) is useful for solving a wide variety of find values for these variables problems.

Loop through each item in the same axis (x or y), if they are the same as the previous item them increment matches. When the next item becomes different, check if matches is or greater than 3, call a function that removes matching items, and continue.

Each type and color of gems can be represented as a number. We also implemented a “gem data structures” which are basically dictionaries with the integer index to denote which image this gem uses.

Image 1 What user see

5	1	5	6	2	4	5	1
3	2	6	7	5	6	4	6
1	5	1	2	7	4	5	3
5	3	7	4	1	1	7	1
6	2	5	2	3	4	3	4
7	4	3	4	1	5	6	1
3	2	6	1	7	3	4	7
4	1	3	6	4	7	5	1

Table 1 What programmer see

We try to match all of the number with 16 possible cases shown above. Because our game has finite set of domains and possibility, we can display a hint which is basically all of the possible moves which can be visualize by the following table.

Image 2 The orange frame indicates possible moves

5	1	5	6	2	4	5	1
3	2	6	7	5	6	4	6
1	5	1	2	7	4	5	3
5	3	7	4	1	1	7	1
6	2	5	2	3	4	3	4
7	4	3	4	1	5	6	1
3	2	6	1	7	3	4	7
4	1	3	6	4	7	5	1

Table 2 Pattern matching techniques

How the program works

We try to solve the problem in many aspects.

Our first attempt is **constraint logic programming over finite domains** (also known as CLP(FD).)

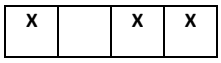
First, activate clp(fd) library.

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

Initialize an empty board.

```
:- initial_board([[0,0,0,0,0,0,0,0],
                  [0,0,0,0,0,0,0,0],
                  [0,0,0,0,0,0,0,0],
                  [0,0,0,0,0,0,0,0],
                  [0,0,0,0,0,0,0,0],
                  [0,0,0,0,0,0,0,0],
                  [0,0,0,0,0,0,0,0],
                  [0,0,0,0,0,0,0,0]]).
```

Then, we try to solve the easiest patterns (0,0), (1,0), (3,0)




```
solvepattern1_match(X, Y, Z, List) :-

      Num in 1..7, % Different type of gems (7 types)
      nth0(X, List, Num),
      nth0(Y, List, Num),
      nth0(Z, List, Num),
      format('(0,~w), (0,~w), (0,~w)', [X, Y, Z]), nl.
```

The `solvepattern1_match()` function works perfectly fine.

But our problem is

```
solvepattern1_eachrow(List) :-

      solvepattern1_match(0,1,3, List),
      solvepattern1_match(1,2,4, List),
      solvepattern1_match(2,3,5, List),
      solvepattern1_match(3,4,6, List),
      solvepattern1_match(4,5,7, List).

solvepattern1(Board) :-

      Board = [Row0,Row1,Row2,Row3,Row4,Row5,Row6,Row7],
      solvepattern1_eachrow(Row0),
      solvepattern1_eachrow(Row1),
      solvepattern1_eachrow(Row2),
      solvepattern1_eachrow(Row3),
      solvepattern1_eachrow(Row4),
      solvepattern1_eachrow(Row5),
      solvepattern1_eachrow(Row6),
      solvepattern1_eachrow(Row7),
      write('Pattern 1 Completed.'), nl.
```

The program keeps return an unnecessary output and errors. We did many attempts, but it failed. Gem data structures is considered as multiple nested lists, it is considered as a tedious job.

We did a second attempt by looking for the example from the internet. We found that in Stackoverflow, Q&A webboard, there is a topic called “Prolog – Iterate through matrix.” (https://stackoverflow.com/questions/34949724/prolog-iterate-through-matrix)

Since our project has to deal with a lot of nested list, it could be considered as 2D matrix. We try to implement a program by using that website as a guideline.

```
matrix(Matrix, I, J, Value) :-

    nth0(I, Matrix, Row),
    nth0(J, Row, Value).

equal(val,X,Y,Z):- val =:= X =:= Y =:= Z
```

We try to apply our knowledge study in the class and read a book “Programming in Prolog 5th edition.” We did manage to get the logic for pattern matching

```
match1(data,I,J,value):-

      matrix(data,I,J+1,temp1),
      matrix(data,I+1,J,temp2),
      matrix(data,I+2,J,temp3),
      result = equal(value,temp1,temp2,temp3),
      (result == true ->

            write('(',I,',',J+1,') '),
            write('(',I+1,',',J,') '),
            write('(',I+2,',',J,') '),ln;

       result == false ->

             write()
      ).
      %if result is yes, then print coordinates, else skip.
```

Unfortunately, in the second attempt, it failed.

In order to deliver a project on time, we decide to do a logic function in the Python instead. It might not meet a requirement that we have to use SWI-Prolog as a logic programming. But the program works perfectly fine.

Problem and Outcomes

Jewe requires Python and Prolog. Pyswip is an extension which act as a bridge, link between Python and Prolog.

We found that Pyswip cause an unexpected error and we really have no idea what it means. We try looking up, but no one faced this problem ever. We even ask for help in Stackoverflow (https://stackoverflow.com/questions/53887198/) but no one can help us.

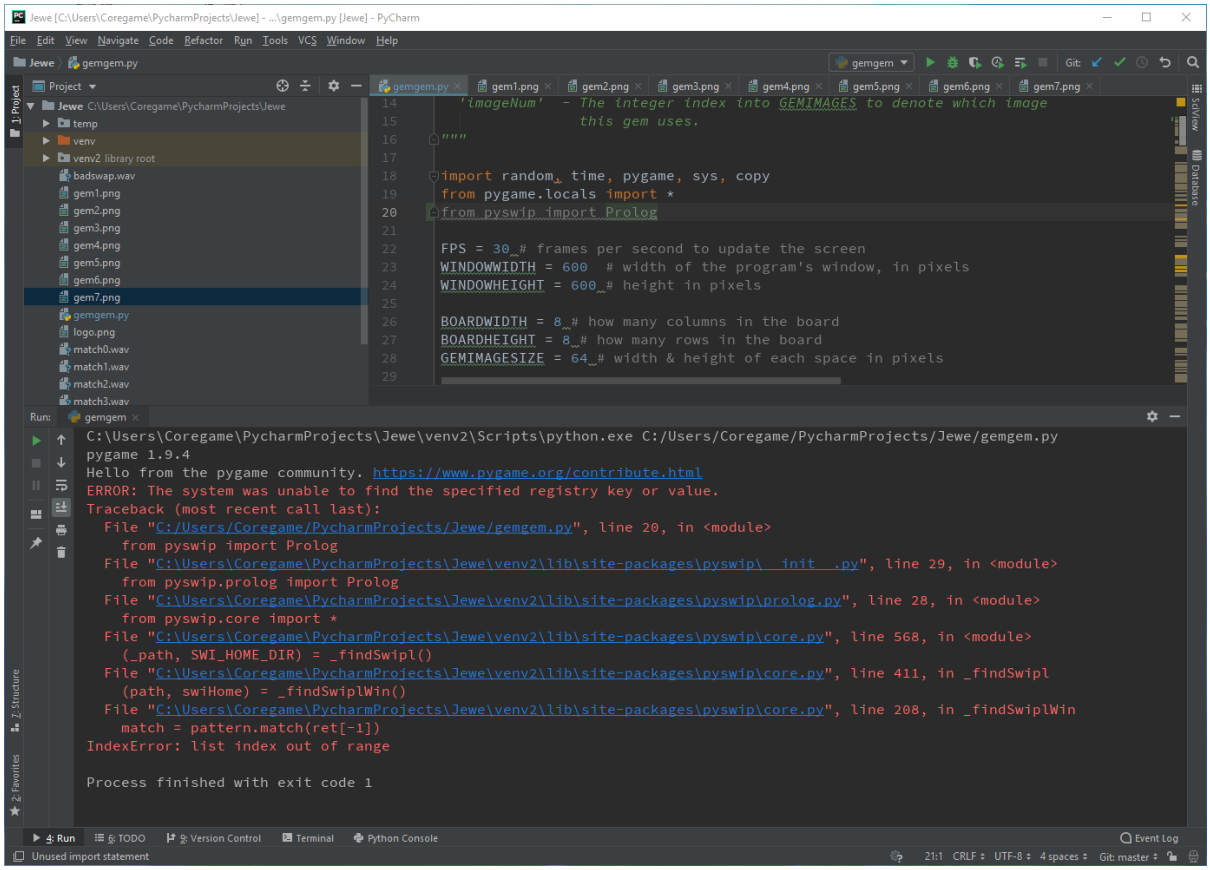


Figure 2 An unexpected error from pyswip

We believe that this is some bugs or glitch from pyswip because according to the Yuce Tekol, pyswip developer, pyswip itself is not yet perfect.

So, we try to use an alternative method instead. It might be an imperfect solution, but we manage to get the pleasant outcome.

Screenshots

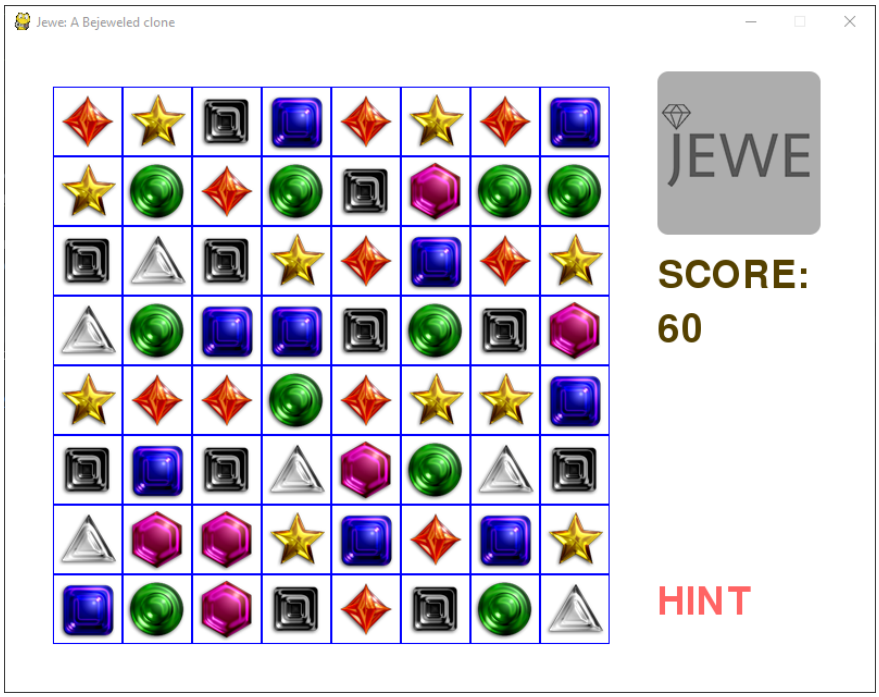


Figure 3 Without Hint

Figure 4 With Hint indicator

Conclusion

In conclusion, Jewe is a fun and astonishing project written in Python with the Pygame and the hint function is implemented in Prolog. We have tried to keep the source code simple so it is easy to follow, and it might be a potential for those who want to learn programming from it.

At this rate, this project might be considered as a playable but not yet perfect. We strongly believe that Jewe have a lot of potential and more additional features with new game mechanic can be added in the future.

Sourcecodes

First Attempt

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

initial_board([[0,0,0,0,0,0,0,0],
               [0,0,0,0,0,0,0,0],
               [0,0,0,0,0,0,0,0],
               [0,0,0,0,0,0,0,0],
               [0,0,0,0,0,0,0,0],
               [0,0,0,0,0,0,0,0],
                    [0,0,0,0,0,0,0,0],
               [0,0,0,0,0,0,0,0]]).

sample_board([[5,5,3,5,2,4,5,1],
               [3,2,6,7,5,6,4,6],
               [1,5,1,2,7,4,5,3],
               [5,3,7,4,1,1,7,1],
                    [6,2,5,2,3,4,3,4],
                    [7,4,3,4,1,5,6,1],
                    [3,2,6,1,7,3,4,7],
                    [4,1,3,6,4,7,5,1],
                    [2,2,5,2,1,3,2,4]]).

%%% Solve different type of patterns
%%
%%       Pattern1
%%       (0,0), (1,0), (3,0)
%%
%%       XX_X
%%
solvepattern1_match(X, Y, Z, List) :-
         Num in 1..7, % Different type of gems (7 types)
         nth0(X, List, Num),
         nth0(Y, List, Num),
         nth0(Z, List, Num),
         format('(0,~w), (0,~w), (0,~w)', [X, Y, Z]), nl.

solvepattern1_eachrow(List) :-
         solvepattern1_match(0,1,3, List),
         solvepattern1_match(1,2,4, List),
         solvepattern1_match(2,3,5, List),
         solvepattern1_match(3,4,6, List),
         solvepattern1_match(4,5,7, List).

solvepattern1(Board) :-
         Board = [Row0,Row1,Row2,Row3,Row4,Row5,Row6,Row7],
         solvepattern1_eachrow(Row0),
         solvepattern1_eachrow(Row1),
         solvepattern1_eachrow(Row2),
         solvepattern1_eachrow(Row3),
         solvepattern1_eachrow(Row4),
         solvepattern1_eachrow(Row5),
         solvepattern1_eachrow(Row6),
         solvepattern1_eachrow(Row7),
         write('Pattern 1 Completed.'), nl.

%%       Pattern2
%%       (0,0), (2,0), (3,0)
%%
%%       X_XX
%%
solvepattern2_match(X, Y, Z, List) :-
         Num in 1..7, % Different type of gems (7 types)
         nth0(X, List, Num),
         nth0(Y, List, Num),
```

```
64          nth0(Z, List, Num),
65          format('(0,~w), (0,~w), (0,~w)', [X, Y, Z]), nl.
66
67 solvepattern2_eachrow(List) :-
68          solvepattern2_match(0,2,3, List),
69          solvepattern2_match(1,3,4, List),
70          solvepattern2_match(2,4,5, List),
71          solvepattern2_match(3,5,6, List),
72          solvepattern2_match(4,6,7, List).
73
74 solvepattern2(Board) :-
75          Board = [Row0,Row1,Row2,Row3,Row4,Row5,Row6,Row7],
76          solvepattern2_eachrow(Row0),
77          solvepattern2_eachrow(Row1),
78          solvepattern2_eachrow(Row2),
79          solvepattern2_eachrow(Row3),
80          solvepattern2_eachrow(Row4),
81          solvepattern2_eachrow(Row5),
82          solvepattern2_eachrow(Row6),
83          solvepattern2_eachrow(Row7),
84          write('Pattern 2 Completed.'), nl.
```

Second Attempt

```
 1 use warnings FATAL => 'all';
 2 use strict;
 3
 4 initial_board([[0,0,0,0,0,0,0,0],
 5                 [0,0,0,0,0,0,0,0],
 6                 [0,0,0,0,0,0,0,0],
 7                 [0,0,0,0,0,0,0,0],
 8                 [0,0,0,0,0,0,0,0],
 9                 [0,0,0,0,0,0,0,0],
10                     [0,0,0,0,0,0,0,0],
11                 [0,0,0,0,0,0,0,0]]).
12
13
14 sample_board([[5,1,5,6,2,4,5,1],
15                 [3,2,6,7,5,6,4,6],
16                 [1,5,1,2,7,4,5,3],
17                 [5,3,7,4,1,1,7,1],
18                     [6,2,5,2,3,4,3,4],
19                     [7,4,3,4,1,5,6,1],
20                     [3,2,6,1,7,3,4,7],
21                     [4,1,3,6,4,7,5,1]]).
22
23 pattern1 = ((0,1), (1,0), (2,0))
24 pattern2 = ((0,1), (1,1), (2,0))
25 pattern3 = ((0,0), (1,1), (2,0))
26 pattern4 = ((0,1), (1,0), (2,1))
27 pattern5 = ((0,0), (1,0), (2,1))
28 pattern6 = ((0,0), (1,1), (2,1))
29 pattern7 = ((0,0), (0,2), (0,3))
30 pattern8 = ((0,0), (0,1), (0,3))
31 pattern9 = ((0,0), (1,0), (3,0))
32 pattern10 = ((0,0), (2,0), (3,0))
33 pattern11 = ((0,0), (0,1), (1,2))
34 pattern12 = ((0,0), (1,1), (0,2))
35 pattern13 = ((1,0), (0,1), (0,2))
36 pattern14 = ((1,0), (1,1), (0,2))
37 pattern15 = ((1,0), (0,1), (1,2))
38 pattern16 = ((0,0), (1,1), (1,2))
39
40 %%%%%%%% get value of that index from data
41 matrix(Matrix, I, J, Value) :-
42     nth0(I, Matrix, Row),
```

```
43      nth0(J, Row, Value).
44
45  equal(val,X,Y,Z):- val =:= X =:= Y =:= Z
46
47  %%https://stackoverflow.com/questions/34949724/prolog-iterate-through-matrix
48  match(data,I,J,match):-
49          nth0(I,data,Row),
50          nth0(J,Row,value),%getValue of XY
51          match1(data,I,J,value),%%send value to check pattern
52          match2(data,I,J,value),
53          match3(data,I,J,value),
54          match4(data,I,J,value),
55          match5(data,I,J,value),
56          match6(data,I,J,value),
57          match7(data,I,J,value),
58          match8(data,I,J,value).
59
60  match1(data,I,J,value):-
61          matrix(data,I,J+1,temp1),%%apply pattern here
62          matrix(data,I+1,J,temp2),
63          matrix(data,I+2,J,temp3),
64          result = equal(value,temp1,temp2,temp3),
65          (result == true ->
66                  write('(',I,',',J+1,') '),
67                  write('(',I+1,',',J,') '),
68                  write('(',I+2,',',J,') '),ln;
69           result == false ->
70                   write()
71          ).
72          %if result is yes, then print coordinates, else skip.
73
74  match2(data,I,J,value):-
75          matrix(data,I,J+1,temp1),%%apply pattern here
76          matrix(data,I+1,J+1,temp2),
77          matrix(data,I+2,J,temp3),
78          result = equal(value,temp1,temp2,temp3),
79          (result == true ->
80                  write('(',I,',',J+1,') '),%change the rest to match the coord
81  from match2 onwards
82                  write('(',I+1,',',J+1,') '),
83                  write('(',I+2,',',J,') '),ln;
84           result == false ->
85                   write()
86          ).
87
88  match3(data,I,J,value):-
89          matrix(data,I,J,temp1),%%apply pattern here
90          matrix(data,I+1,J+1,temp2),
91          matrix(data,I+2,J,temp3),
92          result = equal(value,temp1,temp2,temp3),
93          (result == true ->
94                  write('(',I,',',J,') '),
95                  write('(',I+1,',',J+1,') '),
96                  write('(',I+2,',',J,') '),ln;
97           result == false ->
98                   write()
99          ).
100
101 match4(data,I,J,value):-
102         matrix(data,I,J+1,temp1),%%apply pattern here
103         matrix(data,I+1,J,temp2),
104         matrix(data,I+2,J+1,temp3),
105         result = equal(value,temp1,temp2,temp3),
106         (result == true ->
107                 write('(',I,',',J+1,') '),
108                 write('(',I+1,',',J,') '),
109                 write('(',I+2,',',J+1,') '),ln;
110          result == false ->
```

```
                    write()
        ).

match5(data,I,J,value):-
        matrix(data,I,J,temp1),%%apply pattern here
        matrix(data,I+1,J,temp2),
        matrix(data,I+2,J+1,temp3),
        result = equal(value,temp1,temp2,temp3),
        (result == true ->
                write('(',I,',',J,') '),
                write('(',I+1,',',J,') '),
                write('(',I+2,',',J+1,') '),ln;
         result == false ->
                write()
        ).

match6(data,I,J,value):-
        matrix(data,I,J,temp1),%%apply pattern here
        matrix(data,I+1,J+1,temp2),
        matrix(data,I+2,J+1,temp3),
        result = equal(value,temp1,temp2,temp3),
        (result == true ->
                write('(',I,',',J,') '),
                write('(',I+1,',',J+1,') '),
                write('(',I+2,',',J+1,') '),ln;
         result == false ->
                write()
        ).

match7(data,I,J,value):-
        matrix(data,I,J,temp1),%%apply pattern here
        matrix(data,I,J+2,temp2),
        matrix(data,I,J+3,temp3),
        result = equal(value,temp1,temp2,temp3),
        (result == true ->
                write('(',I,',',J,') '),
                write('(',I,',',J+2,') '),
                write('(',I+3,',',J,') '),ln;
         result == false ->
                write()
        ).

match8(data,I,J,value):-
        matrix(data,I,J,temp1),%%apply pattern here
        matrix(data,I,J+1,temp2),
        matrix(data,I,J+3,temp3),
        result = equal(value,temp1,temp2,temp3),
        (result == true ->
                write('(',I,',',J,') '),
                write('(',I,',',J+1,') '),
                write('(',I,',',J+3,') '),ln;
         result == false ->
                write()
        ).

match9(data,I,J,value):-
        matrix(data,I,J,temp1),%%apply pattern here
        matrix(data,I+1,J,temp2),
        matrix(data,I+3,J,temp3),
        result = equal(value,temp1,temp2,temp3),
        (result == true ->
                write('(',I,',',J,') '),
                write('(',I+1,',',J,') '),
                write('(',I+3,',',J,') '),ln;
         result == false ->
                write()
        ).

```

```
179 match10(data,I,J,value):-
180         matrix(data,I,J,temp1),%%apply pattern here
181         matrix(data,I+2,J,temp2),
182         matrix(data,I+3,J,temp3),
183         result = equal(value,temp1,temp2,temp3),
184         (result == true ->
185                 write('(',I,',',J,') '),
186                 write('(',I+2,',',J,') '),
187                 write('(',I+3,',',J,') '),ln;
188          result == false ->
189                  write()
190         ).
191
192 match11(data,I,J,value):-
193         matrix(data,I,J,temp1),%%apply pattern here
194         matrix(data,I,J+1,temp2),
195         matrix(data,I+1,J+2,temp3),
196         result = equal(value,temp1,temp2,temp3),
197         (result == true ->
198                 write('(',I,',',J,') '),
199                 write('(',I,',',J+1,') '),
200                 write('(',I+1,',',J+2,') '),ln;
201          result == false ->
202                  write()
203         ).
204
205 match12(data,I,J,value):-
206         matrix(data,I,J,temp1),%%apply pattern here
207         matrix(data,I+1,J+1,temp2),
208         matrix(data,I,J+2,temp3),
209         result = equal(value,temp1,temp2,temp3),
210         (result == true ->
211                 write('(',I,',',J,') '),
212                 write('(',I+1,',',J+1,') '),
213                 write('(',I,',',J+2,') '),ln;
214          result == false ->
215                  write()
216         ).
217
218 match13(data,I,J,value):-
219         matrix(data,I+1,J,temp1),%%apply pattern here
220         matrix(data,I,J+1,temp2),
221         matrix(data,I,J+2,temp3),
222         result = equal(value,temp1,temp2,temp3),
223         (result == true ->
224                 write('(',I+1,',',J,') '),
225                 write('(',I,',',J+1,') '),
226                 write('(',I,',',J+2,') '),ln;
227          result == false ->
228                  write()
229         ).
230
231 match14(data,I,J,value):-
232         matrix(data,I+1,J,temp1),%%apply pattern here
233         matrix(data,I+1,J+1,temp2),
234         matrix(data,I,J+2,temp3),
235         result = equal(value,temp1,temp2,temp3),
236         (result == true ->
237                 write('(',I+1,',',J,') '),
238                 write('(',I+1,',',J+1,') '),
239                 write('(',I,',',J+2,') '),ln;
240          result == false ->
241                  write()
242         ).
243
244 match15(data,I,J,value):-
245         matrix(data,I+1,J,temp1),%%apply pattern here
246         matrix(data,I,J+1,temp2),
```

```
247         matrix(data,I+1,J+2,temp3),
248         result = equal(value,temp1,temp2,temp3),
249         (result == true ->
250                 write('(',I+1,',',J,') '),
251                 write('(',I+1,',',J+1,') '),
252                 write('(',I,',',J+2,') '),ln;
253          result == false ->
254                  write()
255         ).
256
257 match16(data,I,J,value):-
258         matrix(data,I,J,temp1),%%apply pattern here
259         matrix(data,I+1,J+1,temp2),
260         matrix(data,I+1,J+2,temp3),
261         result = equal(value,temp1,temp2,temp3),
262         (result == true ->
263                 write('(',I,',',J,') '),
264                 write('(',I+1,',',J+1,') '),
265                 write('(',I+1,',',J+2,') '),ln;
266          result == false ->
267                  write()
            ).
```