AKARI // ANSWER SET PROGRAMMING

PROGRAM FLOW:

1.1. Enter the given information about the puzzle. (Black cells and their numbers are given, those are your inputs.)

1.2. Generate the puzzle with every cell full of bulbs except black cells, then eliminate the light bulbs accordingly to the process of the puzzle. (Output is the enlightened grid.)

///numaraya göre bulb yerleştir 0{queen(I,J):num(I)}1 :- num(J).

2. Light bulbs should be distributed around black squares according to the number written inside that square.

3. When a light bulb inserted to the puzzle, that light bulb illuminated the row and column it placed until light hits to a puzzle corner or black cell.

4. Every white square should be illuminated .

5.İki lamba birbirini aydınlatamaz

ELEMENTS TO GENERATE

- NotNumeratedBlacksCells (I, J).
- NumberedBlackCells (I, J, K). K GİVEN NUMBER OF THE BLACK CELL.
- Lightbulbs (I, J): location of the lightbulbs should be generated in order to illuminate the puzzle
    - I and J combinations should be different than the generated black squares.
    - Firstly, light bulbs should be placed around black cells according to the number written inside them.
    - Afterwards, light bulbs should distribute to rows and columns which were not illuminated.
    - Lights coming from the bulbs can overlap.

CODE

1, 1	1,2 X 1	1,3 O	1,4 L OO	1,5 X	1,6
2,1	2,2 X	2,3 L	2,4	2,5	2,6
3,1 L	3,2 L	3,3 O	3,4 X 2	3,5	3,6 X
4,1 X	4,2 O	4,3 X 4	4,4 O	4,5 L	4,6 L
5,1 L	5,2 L	5,3 O	5,4 L	5,5 X 0	5,6
6,1OO	6,2 X	6,3 L	6,4 L	6,5 X	6,6

```
solution(3,3) solution(4,4) solution(4,2) solution(5,3) solution(6,1) solution(1,4)
solution(2,5) solution(1,6) solution(6,6) solution(1,1)
```

```
SATISFIABLE
```

1, 1 O	1,2 X 1	1,3 L	1,4 O	1,5 X	1,6 O
2,1 L	2,2 X	2,3 L	2,4	2,5 O	2,6
3,1 L	3,2 L	3,3 O	3,4 X 2	3,5	3,6 X
4,1 X	4,2 O	4,3 X 4	4,4 O	4,5 L	4,6 L
5,1 L	5,2 L	5,3 O	5,4 L	5,5 X 0	5,6
6,1 O	6,2 X	6,3 L	6,4 L	6,5 X	6,6 O

CODE #1

Const n = 4.

row(1..n).

column(1..n).

NumBlack(1,2,1). Black(1,5). Black(2,2). NumBlack(3,4,2). Black(3,6).

```
Black(4,1). NumBlack(4,3,4). NumBlack(5,5,0). Black(6,2). Black(6,5).

1{Bulb(I,J):column(I)}1 :- row(J). //Bu syntax'i sor.

:- Black(I,J), Bulb(K,P), I ==K, J ==P.

Bulb(I,J-1), Bulb(I,J+1), Bulb(I-1, J), Bulb(I+1,J) :- NumBlack(I, J, Z), Z==4  //Z =4'se bulbları kesin koy.

// Z=3

Bulb(I,J-1), Bulb(I,J+1), Bulb(I-1, J):- NumBlack(I, J, Z), Z==3

Bulb(I,J-1), Bulb(I,J+1), Bulb(I+1,J) :- NumBlack(I, J, Z), Z==3

Bulb(I,J-1), Bulb(I-1, J), Bulb(I+1,J) :- NumBlack(I, J, Z), Z==3

Bulb(I,J+1), Bulb(I-1, J), Bulb(I+1,J) :- NumBlack(I, J, Z), Z==3

//Z=2

Bulb(I,J-1), Bulb(I,J+1):- NumBlack(I, J, Z), Z==2

Bulb(I-1, J), Bulb(I+1,J) :- NumBlack(I, J, Z), Z==2

Bulb(I,J-1), Bulb(I+1,J) :- NumBlack(I, J, Z), Z==2

Bulb(I,J+1), Bulb(I-1, J) :- NumBlack(I, J, Z), Z==2

Bulb(I,J+1), Bulb(I+1,J) :- NumBlack(I, J, Z), Z==2

Bulb(I,J-1),Bulb(I-1, J) :- NumBlack(I, J, Z), Z==2

//Z=1

Bulb(I,J-1), :- NumBlack(I, J, Z), Z==4

Bulb(I,J+1), :- NumBlack(I, J, Z), Z==4

Bulb(I-1, J) :- NumBlack(I, J, Z), Z==4

Bulb(I+1,J) :- NumBlack(I, J, Z), Z==4

//Z=0

:- Bulb(I,J-1), Bulb(I,J+1), Bulb(I-1, J), Bulb(I+1,J), NumBlack(I, J, Z), Z==0.
:- NumBlack(I,J,Z), Bulb(K,P), I ==K, J ==P.
```

NEW CODE #2

```
#const n = 6.
row(1..n).
column(1..n).
index(0..4).
numBlack(1,2,1). black(1,5). black(2,2). numBlack(3,4,2). black(3,6).
black(4,1). numBlack(4,3,4). numBlack(5,5,0). black(6,2). black(6,5).
black(I,J) :- numBlack(I,J,Z).
white(I,J) :- not black(I,J), row(I), column(J).
neighbor(I, J, I1, J1) :- |I-I1| + |J-J1| == 1, row(I), row(I1), column(J), column(J1).
Z{bulb(I1,J1) : neighbor(I, J, I1, J1), white(I1, J1)}Z :- numBlack(I,J,Z), index(Z).
%:- black(I,J), bulb(K,P), I == K, J == P.
#show bulb/2.
```

```
#CODE 3
#const n = 6.
row(1..n).
column(1..n).
index(0..4).
numBlack(1,2,1). black(1,5). black(2,2). numBlack(3,4,2). black(3,6).
black(4,1). numBlack(4,3,4). numBlack(5,5,0). black(6,2). black(6,5).
black(I,J) :- numBlack(I,J,Z).
white(I,J) :- not black(I,J), row(I), column(J).
neighbor(I, J, I1, J1) :- |I-I1| + |J-J1| == 1, row(I), row(I1), column(J), column(J1).
Z{bulb(I1,J1) : neighbor(I, J, I1, J1), white(I1, J1)}Z :- numBlack(I,J,Z), index(Z).

light(X,Y) :- bulb(X, Y1),  white(X,Y), Y1 < Y, {black(X,K) :  Y1 < K , K < Y}0.  %sağ
% BlackCell görene kadar aydınlat. Black bir olana kadar aydınlatır 1 olursa aydınlatma durur
light(X,Y) :- bulb(X, Y1), white(X,Y), Y1 > Y, {black(X,K) :  Y1 > K, K > Y}0. % sol
light(X,Y) :- bulb(X1, Y),  white(X,Y), X1 < X, {black(M,Y) : X1 < M, M < X}0. % yukarı
light(X,Y) :- bulb(X1, Y),  white(X,Y), X1 > X, {black(M,Y) : X1 > M, M > X}0. % aşağı
light(X,Y) :- bulb (X,Y).
{newbulb(X,Y)} :- not light(X,Y), not black(X,Y) ,row(X), column(Y).
```

```
not newbulb(I1,J1), row(I1), column(J1) :- Z{numBlack(I,J,Z):neighbor(I, J, I1, J1), newbulb(I1, J1)}4, row(I), row(I1), column(J), column(J1), index(Z).

%Fikrimiz  beyaz kalan her tarafa bulb yerleştirmek ve constraint olarak da numblacklerin etrafına bulb yerleştirmemek. Çünkü daha önce numblacklerin çevresine bulb yerleştirdik.

%:- black(I,J), bulb(K,P), I == K, J == P.

#show bulb/2.

#show light/2.
```

CODE #4

```
#const n = 6.

row(1..n).

column(1..n).

index(0..4).

numBlack(1,2,1). black(1,5). black(2,2). numBlack(3,4,2). black(3,6).

black(4,1). numBlack(4,3,4). numBlack(5,5,0). black(6,2). black(6,5).

black(I,J) :- numBlack(I,J,Z).

white(I,J) :- not black(I,J), row(I), column(J).

neighbor(I, J, I1, J1) :- |I-I1| + |J-J1| == 1, row(I), row(I1), column(J), column(J1).

Z{bulb(I1,J1) : neighbor(I, J, I1, J1), white(I1, J1)}Z :- numBlack(I,J,Z), index(Z).


% BlackCell görene kadar aydınlat. Black bir olana kadar aydınlatır 1 olursa aydınlatma durur

light(X,Y) :- bulb(X, Y1),  white(X,Y), Y1 < Y, {black(X,K) :  Y1 < K , K < Y}0.  %sağ

light(X,Y) :- bulb(X, Y1), white(X,Y), Y1 > Y, {black(X,K) :  Y1 > K, K > Y}0. % sol
light(X,Y) :- bulb(X1, Y),  white(X,Y), X1 < X, {black(M,Y) : X1 < M, M < X}0. % yukarı
light(X,Y) :- bulb(X1, Y),  white(X,Y), X1 > X, {black(M,Y) : X1 > M, M > X}0. % aşağı
light(X,Y) :- bulb (X,Y).
{bulb(X,Y)} :- white(X,Y) ,row(X), column(Y).


:- white(X,Y), not light(X,Y).


:- bulb(X,Y), bulb(X,Y1), Y != Y1, Y<Y1,{black(X,Y2):  Y < Y2, Y2 < Y1}0.%sağ same row aralarında black yoksa
:- bulb(X,Y), bulb(X,Y1), Y != Y1, Y>Y1,{black(X,Y2):  Y > Y2, Y2 > Y1}0.%sol
:- bulb(X,Y), bulb(X1,Y), X!=X1, X<X1,{black(X2,Y): X < X2, X2 < X1}0.
:- bulb(X,Y), bulb(X1,Y), X!=X1, X>X1,{black(X2,Y): X > X2, X2 > X1}0.


#show bulb/2.
```

CODE#5

```
#const n = 6.
row(1..n).
column(1..n).
index(0..4).

numBlack(1,2,1). black(1,5). black(2,2). numBlack(3,4,2). black(3,6).
black(4,1). numBlack(4,3,4). numBlack(5,5,0). black(6,2). black(6,5).

black(I,J) :- numBlack(I,J,Z).
white(I,J) :- not black(I,J), row(I), column(J).

neighbor(I, J, I1, J1) :- |I-I1| + |J-J1| == 1, row(I), row(I1), column(J), column(J1).
Z{bulb(I1,J1) : neighbor(I, J, I1, J1), white(I1, J1)}Z :- numBlack(I,J,Z), index(Z).

light(X,Y) :- bulb(X, Y1),  white(X,Y), Y1 < Y, {black(X,K) :  Y1 < K , K < Y}0.  %right
light(X,Y) :- bulb(X, Y1), white(X,Y), Y1 > Y, {black(X,K) :  Y1 > K, K > Y}0. % left
light(X,Y) :- bulb(X1, Y),  white(X,Y), X1 < X, {black(M,Y) : X1 < M, M < X}0. % up
light(X,Y) :- bulb(X1, Y),  white(X,Y), X1 > X, {black(M,Y) : X1 > M, M > X}0. % down
light(X,Y) :- bulb (X,Y).

%For every white cell, generate bulb such that it is not neighbor of any numBlack square
{bulb(X,Y): numBlack(X1, Y1, Z), not neighbor(X1, Y1, X, Y)} :- white(X,Y) ,row(X), column(Y).
```

```
:- white(X,Y), not light(X,Y).


:- bulb(X,Y), bulb(X,Y1), Y != Y1, Y<Y1,{black(X,Y2):  Y < Y2, Y2 < Y1}0.%right

:- bulb(X,Y), bulb(X,Y1), Y != Y1, Y>Y1,{black(X,Y2):  Y > Y2, Y2 > Y1}0.%left

:- bulb(X,Y), bulb(X1,Y), X!=X1, X<X1,{black(X2,Y): X < X2, X2 < X1}0.%up

:- bulb(X,Y), bulb(X1,Y), X!=X1, X>X1,{black(X2,Y): X > X2, X2 > X1}0.%down


#show bulb/2.
```