Department of Computer Science
Technical University of Cluj-Napoca

Artificial Intelligence

Laboratory activity

Name: Patica Andreea-Alexandra and Popa Adriana
Group:30433
Email:popaadriana2606@gmail.com

Teaching Assistant: Adrian Groza
Adrian.Groza@cs.utcluj.ro

Contents

Table 1: Lab scheduling

Activity	**Deadline**
Searching agents, Linux, Latex, Python, Pacman	W_1
Uninformed search	W_2
Informed Search	W_3
Adversarial search	W_4
Propositional logic	W_5
First order logic	W_6
Inference in first order logic	W_7
Knowledge representation in first order logic	W_8
Classical planning	W_9
Contingent, conformant and probabilistic planning	W_{10}
Multi-agent planing	W_{11}
Modelling planning domains	W_{12}
Planning with event calculus	W_{14}

0.0.0.0.1 Lab organisation.

1. Laboratory work is 25% from the final grade.
2. There are three deliverables in total: 1. Search, 2. Logic, 3. Planning.
3. Before each deadline, you have to send your work (latex documentation/code) at moodle.cs.utcluj.ro
4. We use Linux and Latex
5. Plagiarism: Don't be a cheater! Cheating affects your colleagues, scholarships and a lot more.

Chapter 1

A1: Search

Chapter 2

A2: Logics

Chapter 3

A3: Planning

3.1 Brief Introduction

On this assignment we focused on classical planning. Planners use a model of an application domain (such as blocks world) and a description of a specific problem (initial state and goals) to compute a plan.

For this assignment we chose to implement the planning for solving the AllOut game. In this game there is a grid of light-bulbs. Some are turned on and some are turned off. Turning on or off a light will trigger other lights around it to turn on or off, depending on the current state and a predefined rule. The goal is to turn off all the lights.

3.2 Implementation

For implementing this, it has been used a language known as PDDL(Planning Domain Definition Language). It is used to represent all actions into one action schema. PDLL describes the main points needed in a search problem:

- Initial State
- Actions
- Result
- Goal

PDDL is understandable in may situations like Sliding-tiles puzzle, Gripper, air transportation and many more.

3.2.1 Classical AllOut Game

This game uses a grid (preferably of 5x5) of lights which are on or off. By turning on/off a light, all the neighbour lights (up, down, left, right) change their state too.

Domain definition

The domain will consist of predicates and actions. There are seven predicates which should be defined:

1. is-on - defines the state in which the light is in (on or not on)

2. light on edge - states whether the light is on the edge or not. This would help for the situations where the light we wish to turn on/off is on one edge or in a corner. This action would affect other 3, respective 2 lights (not another 4 lights like the normal situation where the light we press is in the middle)
    - is-leftEdge
    - is-rightEdge
    - is-upEdge
    - is-downEdge
3. next-row - used to settle the up and down neighbours of a light
4. next-column - used to settle the left and right neighbours of a light

The actions will have a suggestive name, according to the situation the lights are in. There are three types of actions: center actions, edge actions and corner actions. They receive multiple parameters and manage the effect accordingly:

- **Turn on/off center light:** This would also change the state of the 4 neighbours: up, down, left and right. The state of the center light could be on or off (2 states). The states of the neighbour lights could vary so there are 2 combinations for each neighbour (2x2x2x2 = 16 combinations). We can now assume that there are 2x16=32 combinations for the center action, thus, 32 different actions. The action will receive 6 parameters: current row, current column, left column, right column, up column, down column. In the precondition part, the parameters are checked. The rows and columns must be the correct neighbours and the current position must not be on one of the edges. The effect is that the current light and its neighbours change their current state.

- **Turn on/off edge light:** This would also change the state of the 3 neighbours: left, down, right for the upper edge, left, up, right for the down edge, up, right, down for the left edge and up, right, down for the right edge. The state of the main light could be on or off (2 states). The states of the neighbour lights could vary so there are 2 combinations for each neighbour (2x2x2 = 8 combinations). We can now assume that there are 2x8=16 combinations for the edge action, thus, 16 different actions. The action will receive 5 parameters: current row, current column, the other 3 coordinates for the 3 neighbours. In the precondition part, the parameters are checked. The rows and columns must be the correct neighbours and the current position must be on one of the edges. The effect is that the current light and its neighbours change their current state.

- **Turn on/off corner light:** This would also change the state of the 2 neighbours: right-down for the left up corner, left-down for the right up corner, up-right for the left down corner and up-left for the right down corner. The state of the main light could be on or off (2 states). The states of the neighbour lights could vary so there are 2 combinations for each neighbour (2x2 = 4 combinations). We can now assume that there are 2x4=8 combinations for the corner action, thus, 8 different actions. The action will receive 4 parameters: current row, current column, the other 2 coordinates for the 2 neighbours. In the precondition part, the parameters are checked. The rows and columns must be the correct neighbours and the current position must be on one of the corners (i.e. must be on two consecutive edges). The effect is that the current light and its neighbours change their current state.

Problems:

Each problem consists of objects, init and goal. There are 2 **objects** that need to be defined:

- rows
- columns

since every light bulb is consider to have a position determined by a row and a column. Depending on the complexity each problem has a set of 3, respectively 5 rows and columns, distinguished using numbers. For example in a 3x3 table we will use row1, row2, row3, col1, col2, col3 to give the coordinates of the light bulbs.

The **init** represents the initial state of the problem and consists of declaring the state of each light bulb. The state is composed of the predicates defined in the domain:

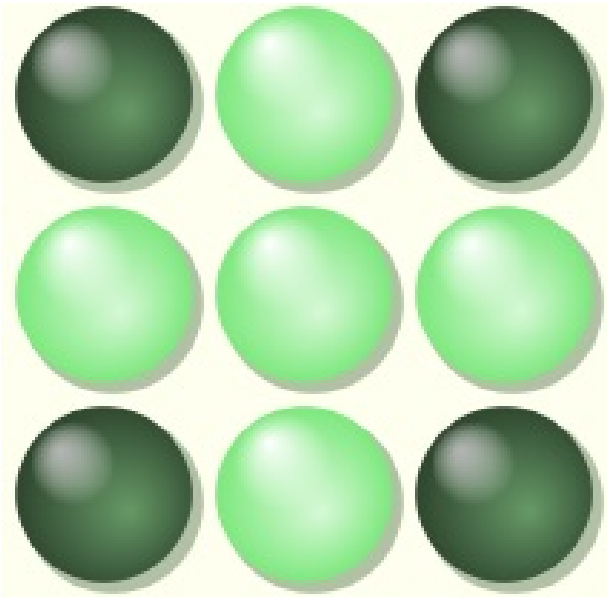
- is-on
- is-leftEdge
- is-rightEdge
- is-upEdge
- is-downEdge
- next-row
- next-column

followed by the row and the column of the bulb it applies to. The next row and column need to be specified in order to have the form off cross, and the margins of the table are declared for a correct defining of the table, so the program can solve it properly.

The **goal** is the same in each problem, all light bulbs need to be off, since this is the scope of the game. It is done by using the is-on predicate and the coordinates of each bulb.

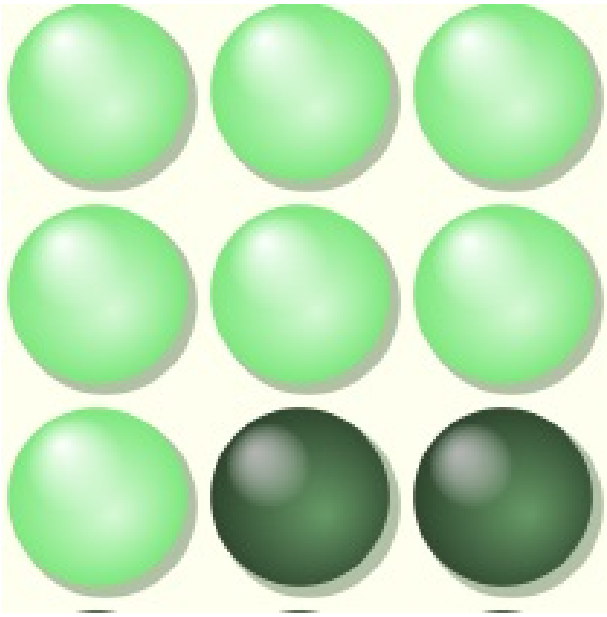
3x3 Table Problem 1

This problem is the simplest case of game which cam be solved in one move by turning off the center.



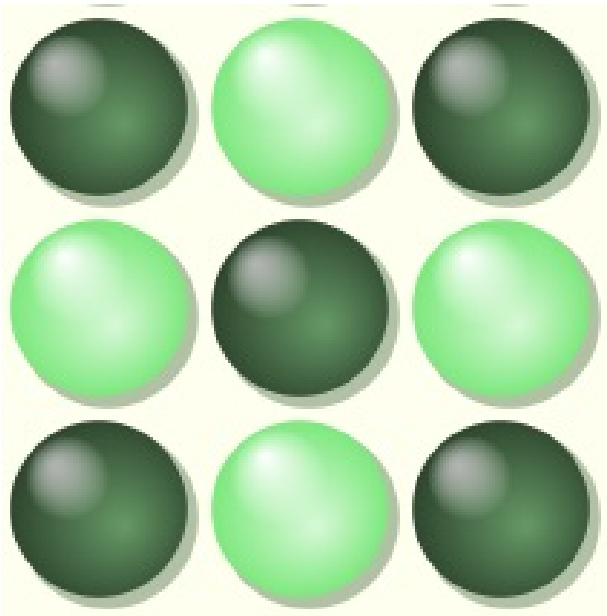
3x3 Table Problem 2

For this problem the complexity rises but the program is still able to solve it in 2 moves by turning off the left edge and the right up corner.



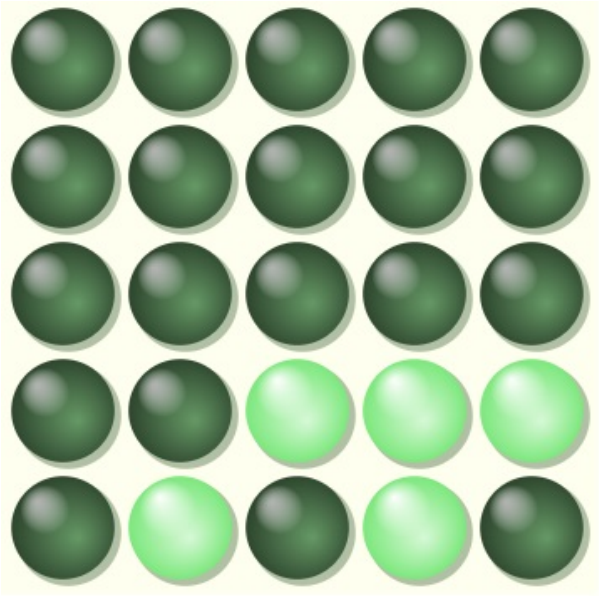
3x3 Table Problem 3

Again, the complexity rises but the program was able to solve it in 8 steps.



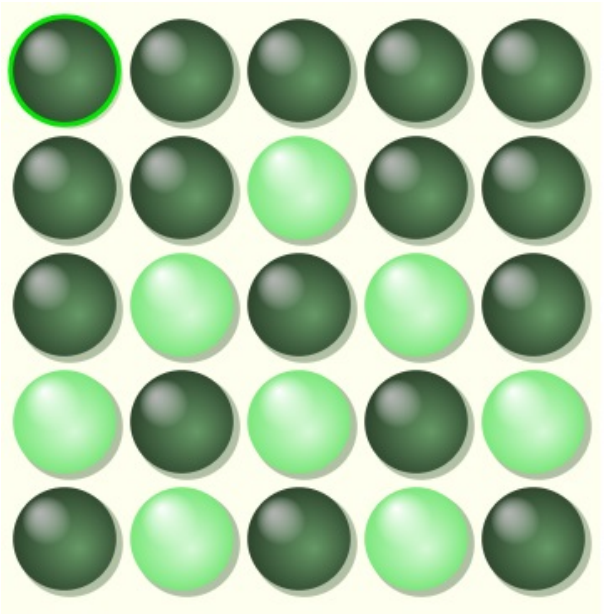
5x5 Table Problem 1

This problem illustrates the original problem of the game having a 5x5 table, although the complexity is not that high, being solved in 3 moves.



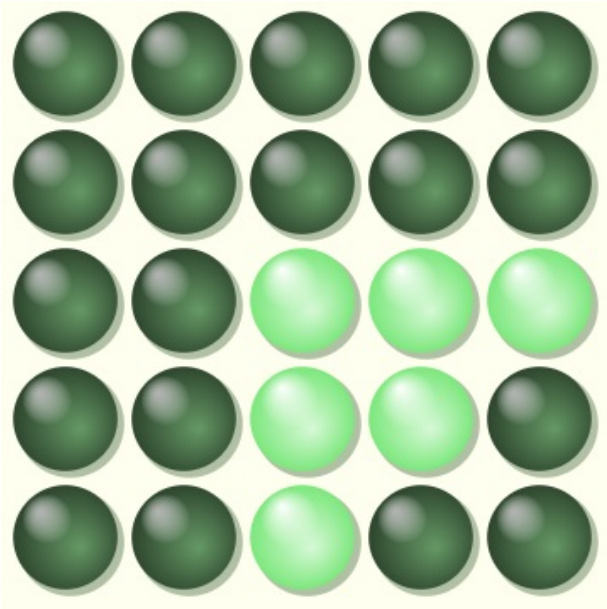
5x5 Table Problem 2

The increased difficulty of this problem gives an increased number of actions for solving, 15, but it's still the optimum number for solving this problem.



5x5 Table Problem 3

Again, the number of actions rises with the difficulty but we obtain the optimum solving for this using just 22 moves.



Bibliography

Laboratory work
https://www.mathsisfun.com/games/allout.html

Appendix A

Your original code

Don't be a cheater! Cheating affects your colleagues, scholarships and a lot more. This section should contain only code developed by you, without any line re-used from other sources. This section helps me to correctly evaluate your amount of work and results obtained.

A.1 Domain

```
(define (domain allOut)

  (:predicates
   (is-on ?row ?col)
   (is-leftEdge ?row ?col)
   (is-rightEdge ?row ?col)
   (is-upEdge ?row ?col)
   (is-downEdge ?row ?col)
   (next-row ?row1 ?row2)
   (next-column ?col1 ?col2)
  )

  (:action turn-on-center-leftOff-rightOff-upOff-downOff
    :parameters (?row ?col ?leftCol ?rightCol ?upRow ?downRow)
    :precondition (and (not (is-on ?row ?col))
                       (is-on ?row ?leftCol)
                       (is-on ?row ?rightCol)
                       (is-on ?upRow ?col)
                       (is-on ?downRow ?col)
                       (next-row ?upRow ?row)
                       (next-row ?row ?downRow)
                       (next-column ?leftCol ?col)
                       (next-column ?col ?rightCol)
                       (not (is-leftEdge ?row ?col))
                       (not (is-rightEdge ?row ?col))
                       (not (is-upEdge ?row ?col))
                       (not (is-downEdge ?row ?col)))
    :effect (and (is-on ?row ?col)
                 (not (is-on ?row ?leftCol))
                 (not (is-on ?row ?rightCol))
                 (not (is-on ?upRow ?col))
```

```
            (not (is-on ?downRow ?col))))

(:action turn-on-center-leftOn-rightOff-upOff-downOff
:parameters (?row ?col ?leftCol ?rightCol ?upRow ?downRow)
:precondition (and (not (is-on ?row ?col))
                   (not (is-on ?row ?leftCol))
                   (is-on ?row ?rightCol)
                   (is-on ?upRow ?col)
                   (is-on ?downRow ?col)
                   (next-row ?upRow ?row)
                   (next-row ?row ?downRow)
                   (next-column ?leftCol ?col)
                   (next-column ?col ?rightCol)
                   (not (is-leftEdge ?row ?col))
                   (not (is-rightEdge ?row ?col))
                   (not (is-upEdge ?row ?col))
                   (not (is-downEdge ?row ?col)))
:effect (and (is-on ?row ?col)
             (is-on ?row ?leftCol)
             (not (is-on ?row ?rightCol))
             (not (is-on ?upRow ?col))
             (not (is-on ?downRow ?col))))

(:action turn-on-center-leftOff-rightOn-upOff-downOff
:parameters (?row ?col ?leftCol ?rightCol ?upRow ?downRow)
:precondition (and (not (is-on ?row ?col))
                   (is-on ?row ?leftCol)
                   (not (is-on ?row ?rightCol))
                   (is-on ?upRow ?col)
                   (is-on ?downRow ?col)
                   (next-row ?upRow ?row)
                   (next-row ?row ?downRow)
                   (next-column ?leftCol ?col)
                   (next-column ?col ?rightCol)
                   (not (is-leftEdge ?row ?col))
                   (not (is-rightEdge ?row ?col))
                   (not (is-upEdge ?row ?col))
                   (not (is-downEdge ?row ?col)))
:effect (and (is-on ?row ?col)
             (not (is-on ?row ?leftCol))
             (is-on ?row ?rightCol)
             (not (is-on ?upRow ?col))
             (not (is-on ?downRow ?col))))

(:action turn-on-center-leftOff-rightOff-upOn-downOff
:parameters (?row ?col ?leftCol ?rightCol ?upRow ?downRow)
:precondition (and (not (is-on ?row ?col))
                   (is-on ?row ?leftCol)
                   (is-on ?row ?rightCol)
                   (not (is-on ?upRow ?col))
```

```
                    (is-on ?downRow ?col)
                    (next-row ?upRow ?row)
                    (next-row ?row ?downRow)
                    (next-column ?leftCol ?col)
                    (next-column ?col ?rightCol)
                    (not (is-leftEdge ?row ?col))
                    (not (is-rightEdge ?row ?col))
                    (not (is-upEdge ?row ?col))
                    (not (is-downEdge ?row ?col)))
    :effect (and (is-on ?row ?col)
                (not (is-on ?row ?leftCol))
                (not (is-on ?row ?rightCol))
                (is-on ?upRow ?col)
                (not (is-on ?downRow ?col))))

    (:action turn-on-center-leftOff-rightOff-upOff-downOn
    :parameters (?row ?col ?leftCol ?rightCol ?upRow ?downRow)
    :precondition (and (not (is-on ?row ?col))
                    (is-on ?row ?leftCol)
                    (is-on ?row ?rightCol)
                    (is-on ?upRow ?col)
                    (not (is-on ?downRow ?col))
                    (next-row ?upRow ?row)
                    (next-row ?row ?downRow)
                    (next-column ?leftCol ?col)
                    (next-column ?col ?rightCol)
                    (not (is-leftEdge ?row ?col))
                    (not (is-rightEdge ?row ?col))
                    (not (is-upEdge ?row ?col))
                    (not (is-downEdge ?row ?col)))
    :effect (and (is-on ?row ?col)
                (not (is-on ?row ?leftCol))
                (not (is-on ?row ?rightCol))
                (not (is-on ?upRow ?col))
                (is-on ?downRow ?col)))

    (:action turn-on-center-leftOn-rightOn-upOff-downOff
    :parameters (?row ?col ?leftCol ?rightCol ?upRow ?downRow)
    :precondition (and (not (is-on ?row ?col))
                    (not (is-on ?row ?leftCol))
                    (not (is-on ?row ?rightCol))
                    (is-on ?upRow ?col)
                    (is-on ?downRow ?col)
                    (next-row ?upRow ?row)
                    (next-row ?row ?downRow)
                    (next-column ?leftCol ?col)
                    (next-column ?col ?rightCol)
                    (not (is-leftEdge ?row ?col))
                    (not (is-rightEdge ?row ?col))
                    (not (is-upEdge ?row ?col))
```

```
                (not (is-downEdge ?row ?col)))
:effect (and (is-on ?row ?col)
             (is-on ?row ?leftCol)
             (is-on ?row ?rightCol)
             (not (is-on ?upRow ?col))
             (not (is-on ?downRow ?col))))

(:action turn-on-center-leftOn-rightOff-upOn-downOff
:parameters (?row ?col ?leftCol ?rightCol ?upRow ?downRow)
:precondition (and (not (is-on ?row ?col))
                (not (is-on ?row ?leftCol))
                (is-on ?row ?rightCol)
                (not (is-on ?upRow ?col))
                (is-on ?downRow ?col)
                (next-row ?upRow ?row)
                (next-row ?row ?downRow)
                (next-column ?leftCol ?col)
                (next-column ?col ?rightCol)
                (not (is-leftEdge ?row ?col))
                (not (is-rightEdge ?row ?col))
                (not (is-upEdge ?row ?col))
                (not (is-downEdge ?row ?col)))
:effect (and (is-on ?row ?col)
             (is-on ?row ?leftCol)
             (not (is-on ?row ?rightCol))
             (is-on ?upRow ?col)
             (not (is-on ?downRow ?col))))

(:action turn-on-center-leftOn-rightOff-upOff-downOn
:parameters (?row ?col ?leftCol ?rightCol ?upRow ?downRow)
:precondition (and (not (is-on ?row ?col))
                (not (is-on ?row ?leftCol))
                (is-on ?row ?rightCol)
                (is-on ?upRow ?col)
                (not (is-on ?downRow ?col))
                (next-row ?upRow ?row)
                (next-row ?row ?downRow)
                (next-column ?leftCol ?col)
                (next-column ?col ?rightCol)
                (not (is-leftEdge ?row ?col))
                (not (is-rightEdge ?row ?col))
                (not (is-upEdge ?row ?col))
                (not (is-downEdge ?row ?col)))
:effect (and (is-on ?row ?col)
             (is-on ?row ?leftCol)
             (not (is-on ?row ?rightCol))
             (not (is-on ?upRow ?col))
             (is-on ?downRow ?col)))

(:action turn-on-center-leftOff-rightOn-upOn-downOff
```

```
:parameters (?row ?col ?leftCol ?rightCol ?upRow ?downRow)
:precondition (and (not (is-on ?row ?col))
                   (is-on ?row ?leftCol)
                   (not (is-on ?row ?rightCol))
                   (not (is-on ?upRow ?col))
                   (is-on ?downRow ?col)
                   (next-row ?upRow ?row)
                   (next-row ?row ?downRow)
                   (next-column ?leftCol ?col)
                   (next-column ?col ?rightCol)
                   (not (is-leftEdge ?row ?col))
                   (not (is-rightEdge ?row ?col))
                   (not (is-upEdge ?row ?col))
                   (not (is-downEdge ?row ?col)))
:effect (and (is-on ?row ?col)
             (not (is-on ?row ?leftCol))
             (is-on ?row ?rightCol)
             (is-on ?upRow ?col)
             (not (is-on ?downRow ?col))))

(:action turn-on-center-leftOff-rightOn-upOff-downOn
:parameters (?row ?col ?leftCol ?rightCol ?upRow ?downRow)
:precondition (and (not (is-on ?row ?col))
                   (is-on ?row ?leftCol)
                   (not (is-on ?row ?rightCol))
                   (is-on ?upRow ?col)
                   (not (is-on ?downRow ?col))
                   (next-row ?upRow ?row)
                   (next-row ?row ?downRow)
                   (next-column ?leftCol ?col)
                   (next-column ?col ?rightCol)
                   (not (is-leftEdge ?row ?col))
                   (not (is-rightEdge ?row ?col))
                   (not (is-upEdge ?row ?col))
                   (not (is-downEdge ?row ?col)))
:effect (and (is-on ?row ?col)
             (not (is-on ?row ?leftCol))
             (is-on ?row ?rightCol)
             (not (is-on ?upRow ?col))
             (is-on ?downRow ?col)))

(:action turn-on-center-leftOff-rightOff-upOn-downOn
:parameters (?row ?col ?leftCol ?rightCol ?upRow ?downRow)
:precondition (and (not (is-on ?row ?col))
                   (is-on ?row ?leftCol)
                   (is-on ?row ?rightCol)
                   (not (is-on ?upRow ?col))
                   (not (is-on ?downRow ?col))
                   (next-row ?upRow ?row)
                   (next-row ?row ?downRow)
```

```
                (next-column ?leftCol ?col)
                (next-column ?col ?rightCol)
                (not (is-leftEdge ?row ?col))
                (not (is-rightEdge ?row ?col))
                (not (is-upEdge ?row ?col))
                (not (is-downEdge ?row ?col)))
:effect (and (is-on ?row ?col)
             (not (is-on ?row ?leftCol))
             (not (is-on ?row ?rightCol))
             (is-on ?upRow ?col)
             (is-on ?downRow ?col)))

(:action turn-on-center-leftOn-rightOn-upOn-downOff
:parameters (?row ?col ?leftCol ?rightCol ?upRow ?downRow)
:precondition (and (not (is-on ?row ?col))
                (not (is-on ?row ?leftCol))
                (not (is-on ?row ?rightCol))
                (not (is-on ?upRow ?col))
                (is-on ?downRow ?col)
                (next-row ?upRow ?row)
                (next-row ?row ?downRow)
                (next-column ?leftCol ?col)
                (next-column ?col ?rightCol)
                (not (is-leftEdge ?row ?col))
                (not (is-rightEdge ?row ?col))
                (not (is-upEdge ?row ?col))
                (not (is-downEdge ?row ?col)))
:effect (and (is-on ?row ?col)
             (is-on ?row ?leftCol)
             (is-on ?row ?rightCol)
             (is-on ?upRow ?col)
             (not (is-on ?downRow ?col))))

(:action turn-on-center-leftOn-rightOn-upOff-downOn
:parameters (?row ?col ?leftCol ?rightCol ?upRow ?downRow)
:precondition (and (not (is-on ?row ?col))
                (not (is-on ?row ?leftCol))
                (not (is-on ?row ?rightCol))
                (is-on ?upRow ?col)
                (not (is-on ?downRow ?col))
                (next-row ?upRow ?row)
                (next-row ?row ?downRow)
                (next-column ?leftCol ?col)
                (next-column ?col ?rightCol)
                (not (is-leftEdge ?row ?col))
                (not (is-rightEdge ?row ?col))
                (not (is-upEdge ?row ?col))
                (not (is-downEdge ?row ?col)))
:effect (and (is-on ?row ?col)
             (is-on ?row ?leftCol)
```

```
            (is-on ?row ?rightCol)
            (not (is-on ?upRow ?col))
            (is-on ?downRow ?col)))

(:action turn-on-center-leftOn-rightOff-upOn-downOn
:parameters (?row ?col ?leftCol ?rightCol ?upRow ?downRow)
:precondition (and (not (is-on ?row ?col))
                (not (is-on ?row ?leftCol))
                (is-on ?row ?rightCol)
                (not (is-on ?upRow ?col))
                (not (is-on ?downRow ?col))
                (next-row ?upRow ?row)
                (next-row ?row ?downRow)
                (next-column ?leftCol ?col)
                (next-column ?col ?rightCol)
                (not (is-leftEdge ?row ?col))
                (not (is-rightEdge ?row ?col))
                (not (is-upEdge ?row ?col))
                (not (is-downEdge ?row ?col)))
:effect (and (is-on ?row ?col)
            (is-on ?row ?leftCol)
            (not (is-on ?row ?rightCol))
            (is-on ?upRow ?col)
            (is-on ?downRow ?col)))

(:action turn-on-center-leftOff-rightOn-upOn-downOn
:parameters (?row ?col ?leftCol ?rightCol ?upRow ?downRow)
:precondition (and (not (is-on ?row ?col))
                (is-on ?row ?leftCol)
                (not (is-on ?row ?rightCol))
                (not (is-on ?upRow ?col))
                (not (is-on ?downRow ?col))
                (next-row ?upRow ?row)
                (next-row ?row ?downRow)
                (next-column ?leftCol ?col)
                (next-column ?col ?rightCol)
                (not (is-leftEdge ?row ?col))
                (not (is-rightEdge ?row ?col))
                (not (is-upEdge ?row ?col))
                (not (is-downEdge ?row ?col)))
:effect (and (is-on ?row ?col)
            (not (is-on ?row ?leftCol))
            (is-on ?row ?rightCol)
            (is-on ?upRow ?col)
            (is-on ?downRow ?col)))

   (:action turn-on-center-leftOn-rightOn-upOn-downOn
:parameters (?row ?col ?leftCol ?rightCol ?upRow ?downRow)
:precondition (and (not (is-on ?row ?col))
                (not (is-on ?row ?leftCol))
```

```
                (not (is-on ?row ?rightCol))
                (not (is-on ?upRow ?col))
                (not (is-on ?downRow ?col))
                (next-row ?upRow ?row)
                (next-row ?row ?downRow)
                (next-column ?leftCol ?col)
                (next-column ?col ?rightCol)
                (not (is-leftEdge ?row ?col))
                (not (is-rightEdge ?row ?col))
                (not (is-upEdge ?row ?col))
                (not (is-downEdge ?row ?col)))
:effect (and (is-on ?row ?col)
             (is-on ?row ?leftCol)
             (is-on ?row ?rightCol)
             (is-on ?upRow ?col)
             (is-on ?downRow ?col)))

(:action turn-on-leftEdge-rightOff-upOff-downOff
:parameters (?row ?col ?rightCol ?upRow ?downRow)
:precondition (and (not (is-on ?row ?col))
                (is-on ?row ?rightCol)
                (is-on ?upRow ?col)
                (is-on ?downRow ?col)
                (is-leftEdge ?row ?col)
                (next-row ?upRow ?row)
                (next-row ?row ?downRow)
                (next-column ?col ?rightCol)
                (not (is-rightEdge ?row ?col))
                (not (is-upEdge ?row ?col))
                (not (is-downEdge ?row ?col)))
:effect (and (is-on ?row ?col)
             (not (is-on ?row ?rightCol))
             (not (is-on ?upRow ?col))
             (not (is-on ?downRow ?col))))

(:action turn-on-leftEdge-rightOn-upOff-downOff
:parameters (?row ?col ?rightCol ?upRow ?downRow)
:precondition (and (not (is-on ?row ?col))
                (not (is-on ?row ?rightCol))
                (is-on ?upRow ?col)
                (is-on ?downRow ?col)
                (is-leftEdge ?row ?col)
                (next-row ?upRow ?row)
                (next-row ?row ?downRow)
                (next-column ?col ?rightCol)
                (not (is-rightEdge ?row ?col))
                (not (is-upEdge ?row ?col))
                (not (is-downEdge ?row ?col)))
:effect (and (is-on ?row ?col)
             (is-on ?row ?rightCol)
```

```
            (not (is-on ?upRow ?col))
            (not (is-on ?downRow ?col))))

(:action turn-on-leftEdge-rightOff-upOn-downOff
:parameters (?row ?col ?rightCol ?upRow ?downRow)
:precondition (and (not (is-on ?row ?col))
                (is-on ?row ?rightCol)
                (not (is-on ?upRow ?col))
                (is-on ?downRow ?col)
                (is-leftEdge ?row ?col)
                (next-row ?upRow ?row)
                (next-row ?row ?downRow)
                (next-column ?col ?rightCol)
                (not (is-rightEdge ?row ?col))
                (not (is-upEdge ?row ?col))
                (not (is-downEdge ?row ?col)))
:effect (and (is-on ?row ?col)
            (not (is-on ?row ?rightCol))
            (is-on ?upRow ?col)
            (not (is-on ?downRow ?col))))

(:action turn-on-leftEdge-rightOff-upOff-downOn
:parameters (?row ?col ?rightCol ?upRow ?downRow)
:precondition (and (not (is-on ?row ?col))
                (is-on ?row ?rightCol)
                (is-on ?upRow ?col)
                (not (is-on ?downRow ?col))
                (is-leftEdge ?row ?col)
                (next-row ?upRow ?row)
                (next-row ?row ?downRow)
                (next-column ?col ?rightCol)
                (not (is-rightEdge ?row ?col))
                (not (is-upEdge ?row ?col))
                (not (is-downEdge ?row ?col)))
:effect (and (is-on ?row ?col)
            (not (is-on ?row ?rightCol))
            (not (is-on ?upRow ?col))
            (is-on ?downRow ?col)))

(:action turn-on-leftEdge-rightOn-upOn-downOff
:parameters (?row ?col ?rightCol ?upRow ?downRow)
:precondition (and (not (is-on ?row ?col))
                (not (is-on ?row ?rightCol))
                (not (is-on ?upRow ?col))
                (is-on ?downRow ?col)
                (is-leftEdge ?row ?col)
                (next-row ?upRow ?row)
                (next-row ?row ?downRow)
                (next-column ?col ?rightCol)
                (not (is-rightEdge ?row ?col))
```

```
                (not (is-upEdge ?row ?col))
                (not (is-downEdge ?row ?col)))
:effect (and (is-on ?row ?col)
           (is-on ?row ?rightCol)
           (is-on ?upRow ?col)
           (not (is-on ?downRow ?col))))

(:action turn-on-leftEdge-rightOn-upOff-downOn
:parameters (?row ?col ?rightCol ?upRow ?downRow)
:precondition (and (not (is-on ?row ?col))
                (not (is-on ?row ?rightCol))
                (is-on ?upRow ?col)
                (not (is-on ?downRow ?col))
                (is-leftEdge ?row ?col)
                (next-row ?upRow ?row)
                (next-row ?row ?downRow)
                (next-column ?col ?rightCol)
                (not (is-rightEdge ?row ?col))
                (not (is-upEdge ?row ?col))
                (not (is-downEdge ?row ?col)))
:effect (and (is-on ?row ?col)
           (is-on ?row ?rightCol)
           (not (is-on ?upRow ?col))
           (is-on ?downRow ?col)))

(:action turn-on-leftEdge-rightOff-upOn-downOn
:parameters (?row ?col ?rightCol ?upRow ?downRow)
:precondition (and (not (is-on ?row ?col))
                (is-on ?row ?rightCol)
                (not (is-on ?upRow ?col))
                (not (is-on ?downRow ?col))
                (is-leftEdge ?row ?col)
                (next-row ?upRow ?row)
                (next-row ?row ?downRow)
                (next-column ?col ?rightCol)
                (not (is-rightEdge ?row ?col))
                (not (is-upEdge ?row ?col))
                (not (is-downEdge ?row ?col)))
:effect (and (is-on ?row ?col)
           (not (is-on ?row ?rightCol))
           (is-on ?upRow ?col)
           (is-on ?downRow ?col)))

(:action turn-on-leftEdge-rightOn-upOn-downOn
:parameters (?row ?col ?rightCol ?upRow ?downRow)
:precondition (and (not (is-on ?row ?col))
                (not (is-on ?row ?rightCol))
                (not (is-on ?upRow ?col))
                (not (is-on ?downRow ?col))
                (is-leftEdge ?row ?col)
```

```
                (next-row ?upRow ?row)
                (next-row ?row ?downRow)
                (next-column ?col ?rightCol)
                (not (is-rightEdge ?row ?col))
                (not (is-upEdge ?row ?col))
                (not (is-downEdge ?row ?col)))
:effect (and (is-on ?row ?col)
             (is-on ?row ?rightCol)
             (is-on ?upRow ?col)
             (is-on ?downRow ?col)))

(:action turn-on-rightEdge-leftOff-upOff-downOff
:parameters (?row ?col ?leftCol ?upRow ?downRow)
:precondition (and (not (is-on ?row ?col))
                (is-on ?row ?leftCol)
                (is-on ?upRow ?col)
                (is-on ?downRow ?col)
                (next-row ?upRow ?row)
                (next-row ?row ?downRow)
                (next-column ?leftCol ?col)
                (not (is-leftEdge ?row ?col))
                (is-rightEdge ?row ?col)
                (not (is-upEdge ?row ?col))
                (not (is-downEdge ?row ?col)))
:effect (and (is-on ?row ?col)
             (not (is-on ?row ?leftCol))
             (not (is-on ?upRow ?col))
             (not (is-on ?downRow ?col))))

(:action turn-on-rightEdge-leftOn-upOff-downOff
:parameters (?row ?col ?leftCol ?upRow ?downRow)
:precondition (and (not (is-on ?row ?col))
                (not (is-on ?row ?leftCol))
                (is-on ?upRow ?col)
                (is-on ?downRow ?col)
                (next-row ?upRow ?row)
                (next-row ?row ?downRow)
                (next-column ?leftCol ?col)
                (not (is-leftEdge ?row ?col))
                (is-rightEdge ?row ?col)
                (not (is-upEdge ?row ?col))
                (not (is-downEdge ?row ?col)))
:effect (and (is-on ?row ?col)
             (is-on ?row ?leftCol)
             (not (is-on ?upRow ?col))
             (not (is-on ?downRow ?col))))


(:action turn-on-rightEdge-leftOff-upOn-downOff
:parameters (?row ?col ?leftCol ?upRow ?downRow)
```

```
:precondition (and (not (is-on ?row ?col))
                   (is-on ?row ?leftCol)
                   (not (is-on ?upRow ?col))
                   (is-on ?downRow ?col)
                   (next-row ?upRow ?row)
                   (next-row ?row ?downRow)
                   (next-column ?leftCol ?col)
                   (not (is-leftEdge ?row ?col))
                   (is-rightEdge ?row ?col)
                   (not (is-upEdge ?row ?col))
                   (not (is-downEdge ?row ?col)))
:effect (and (is-on ?row ?col)
             (not (is-on ?row ?leftCol))
             (is-on ?upRow ?col)
             (not (is-on ?downRow ?col))))

(:action turn-on-rightEdge-leftOff-upOff-downOn
:parameters (?row ?col ?leftCol ?upRow ?downRow)
:precondition (and (not (is-on ?row ?col))
                   (is-on ?row ?leftCol)
                   (is-on ?upRow ?col)
                   (not (is-on ?downRow ?col))
                   (next-row ?upRow ?row)
                   (next-row ?row ?downRow)
                   (next-column ?leftCol ?col)
                   (not (is-leftEdge ?row ?col))
                   (is-rightEdge ?row ?col)
                   (not (is-upEdge ?row ?col))
                   (not (is-downEdge ?row ?col)))
:effect (and (is-on ?row ?col)
             (not (is-on ?row ?leftCol))
             (not (is-on ?upRow ?col))
             (is-on ?downRow ?col) ))

(:action turn-on-rightEdge-leftOn-upOn-downOff
:parameters (?row ?col ?leftCol ?upRow ?downRow)
:precondition (and (not (is-on ?row ?col))
                   (not (is-on ?row ?leftCol))
                   (not (is-on ?upRow ?col))
                   (is-on ?downRow ?col)
                   (next-row ?upRow ?row)
                   (next-row ?row ?downRow)
                   (next-column ?leftCol ?col)
                   (not (is-leftEdge ?row ?col))
                   (is-rightEdge ?row ?col)
                   (not (is-upEdge ?row ?col))
                   (not (is-downEdge ?row ?col)))
:effect (and (is-on ?row ?col)
             (is-on ?row ?leftCol)
             (is-on ?upRow ?col)
```

```
            (not (is-on ?downRow ?col)) ))

(:action turn-on-rightEdge-leftOn-upOff-downOn
:parameters (?row ?col ?leftCol ?upRow ?downRow)
:precondition (and (not (is-on ?row ?col))
                   (not (is-on ?row ?leftCol))
                   (is-on ?upRow ?col)
                   (not (is-on ?downRow ?col))
                   (next-row ?upRow ?row)
                   (next-row ?row ?downRow)
                   (next-column ?leftCol ?col)
                   (not (is-leftEdge ?row ?col))
                   (is-rightEdge ?row ?col)
                   (not (is-upEdge ?row ?col))
                   (not (is-downEdge ?row ?col)))
:effect (and (is-on ?row ?col)
             (is-on ?row ?leftCol)
             (not (is-on ?upRow ?col))
             (is-on ?downRow ?col) ))

(:action turn-on-rightEdge-leftOff-upOn-downOn
:parameters (?row ?col ?leftCol ?upRow ?downRow)
:precondition (and (not (is-on ?row ?col))
                   (is-on ?row ?leftCol)
                   (not (is-on ?upRow ?col))
                   (not (is-on ?downRow ?col))
                   (next-row ?upRow ?row)
                   (next-row ?row ?downRow)
                   (next-column ?leftCol ?col)
                   (not (is-leftEdge ?row ?col))
                   (is-rightEdge ?row ?col)
                   (not (is-upEdge ?row ?col))
                   (not (is-downEdge ?row ?col)))
:effect (and (is-on ?row ?col)
             (not (is-on ?row ?leftCol))
             (is-on ?upRow ?col)
             (is-on ?downRow ?col)))

(:action turn-on-rightEdge-leftOn-upOn-downOn
:parameters (?row ?col ?leftCol ?upRow ?downRow)
:precondition (and (not (is-on ?row ?col))
                   (not (is-on ?row ?leftCol))
                   (not (is-on ?upRow ?col))
                   (not (is-on ?downRow ?col))
                   (next-row ?upRow ?row)
                   (next-row ?row ?downRow)
                   (next-column ?leftCol ?col)
                   (not (is-leftEdge ?row ?col))
                   (is-rightEdge ?row ?col)
                   (not (is-upEdge ?row ?col))
```

```
                (not (is-downEdge ?row ?col)))
:effect (and (is-on ?row ?col)
             (is-on ?row ?leftCol)
             (is-on ?upRow ?col)
             (is-on ?downRow ?col)))

(:action turn-on-upEdge-leftOff-rightOff-downOff
:parameters (?row ?col ?leftCol ?rightCol ?downRow)
:precondition (and (not (is-on ?row ?col))
                (is-on ?row ?leftCol)
                (is-on ?row ?rightCol)
                (is-on ?downRow ?col)
                (next-row ?row ?downRow)
                (next-column ?leftCol ?col)
                (next-column ?col ?rightCol)
                (not (is-leftEdge ?row ?col))
                (not (is-rightEdge ?row ?col))
                (is-upEdge ?row ?col)
                (not (is-downEdge ?row ?col)))
:effect (and (is-on ?row ?col)
             (not (is-on ?row ?leftCol))
             (not (is-on ?row ?rightCol))
             (not (is-on ?downRow ?col)) ))

(:action turn-on-upEdge-leftOn-rightOff-downOff
:parameters (?row ?col ?leftCol ?rightCol ?downRow)
:precondition (and (not (is-on ?row ?col))
                (not (is-on ?row ?leftCol))
                (is-on ?row ?rightCol)
                (is-on ?downRow ?col)
                (next-row ?row ?downRow)
                (next-column ?leftCol ?col)
                (next-column ?col ?rightCol)
                (not (is-leftEdge ?row ?col))
                (not (is-rightEdge ?row ?col))
                (is-upEdge ?row ?col)
                (not (is-downEdge ?row ?col)))
:effect (and (is-on ?row ?col)
             (is-on ?row ?leftCol)
             (not (is-on ?row ?rightCol))
             (not (is-on ?downRow ?col)) ))

(:action turn-on-upEdge-leftOff-rightOn-downOff
:parameters (?row ?col ?leftCol ?rightCol ?downRow)
:precondition (and (not (is-on ?row ?col))
                (is-on ?row ?leftCol)
                (not (is-on ?row ?rightCol))
                (is-on ?downRow ?col)
                (next-row ?row ?downRow)
                (next-column ?leftCol ?col)
```

```
                (next-column ?col ?rightCol)
                (not (is-leftEdge ?row ?col))
                (not (is-rightEdge ?row ?col))
                (is-upEdge ?row ?col)
                (not (is-downEdge ?row ?col)))
:effect (and (is-on ?row ?col)
           (not (is-on ?row ?leftCol))
           (is-on ?row ?rightCol)
           (not (is-on ?downRow ?col)) ))

(:action turn-on-upEdge-leftOff-rightOff-downOn
:parameters (?row ?col ?leftCol ?rightCol ?downRow)
:precondition (and (not (is-on ?row ?col))
                (is-on ?row ?leftCol)
                (is-on ?row ?rightCol)
                (not (is-on ?downRow ?col))
                (next-row ?row ?downRow)
                (next-column ?leftCol ?col)
                (next-column ?col ?rightCol)
                (not (is-leftEdge ?row ?col))
                (not (is-rightEdge ?row ?col))
                (is-upEdge ?row ?col)
                (not (is-downEdge ?row ?col)))
:effect (and (is-on ?row ?col)
           (not (is-on ?row ?leftCol))
           (not (is-on ?row ?rightCol))
           (is-on ?downRow ?col) ))


(:action turn-on-upEdge-leftOn-rightOn-downOff
:parameters (?row ?col ?leftCol ?rightCol ?downRow)
:precondition (and (not (is-on ?row ?col))
                (not (is-on ?row ?leftCol))
                (not (is-on ?row ?rightCol))
                (is-on ?downRow ?col)
                (next-row ?row ?downRow)
                (next-column ?leftCol ?col)
                (next-column ?col ?rightCol)
                (not (is-leftEdge ?row ?col))
                (not (is-rightEdge ?row ?col))
                (is-upEdge ?row ?col)
                (not (is-downEdge ?row ?col)))
:effect (and (is-on ?row ?col)
           (is-on ?row ?leftCol)
           (is-on ?row ?rightCol)
           (not (is-on ?downRow ?col)) ))

(:action turn-on-upEdge-leftOn-rightOff-downOn
:parameters (?row ?col ?leftCol ?rightCol ?downRow)
:precondition (and (not (is-on ?row ?col))
```

```
                (not (is-on ?row ?leftCol))
                (is-on ?row ?rightCol)
                (not (is-on ?downRow ?col))
                (next-row ?row ?downRow)
                (next-column ?leftCol ?col)
                (next-column ?col ?rightCol)
                (not (is-leftEdge ?row ?col))
                (not (is-rightEdge ?row ?col))
                (is-upEdge ?row ?col)
                (not (is-downEdge ?row ?col)))
:effect (and (is-on ?row ?col)
            (is-on ?row ?leftCol)
            (not (is-on ?row ?rightCol))
            (is-on ?downRow ?col) ))

(:action turn-on-upEdge-leftOff-rightOn-downOn
:parameters (?row ?col ?leftCol ?rightCol ?downRow)
:precondition (and (not (is-on ?row ?col))
                (is-on ?row ?leftCol)
                (not (is-on ?row ?rightCol))
                (not (is-on ?downRow ?col))
                (next-row ?row ?downRow)
                (next-column ?leftCol ?col)
                (next-column ?col ?rightCol)
                (not (is-leftEdge ?row ?col))
                (not (is-rightEdge ?row ?col))
                (is-upEdge ?row ?col)
                (not (is-downEdge ?row ?col)))
:effect (and (is-on ?row ?col)
            (not (is-on ?row ?leftCol))
            (is-on ?row ?rightCol)
            (is-on ?downRow ?col) ))

(:action turn-on-upEdge-leftOn-rightOn-downOn
:parameters (?row ?col ?leftCol ?rightCol ?downRow)
:precondition (and (not (is-on ?row ?col))
                (not (is-on ?row ?leftCol))
                (not (is-on ?row ?rightCol))
                (not (is-on ?downRow ?col))
                (next-row ?row ?downRow)
                (next-column ?leftCol ?col)
                (next-column ?col ?rightCol)
                (not (is-leftEdge ?row ?col))
                (not (is-rightEdge ?row ?col))
                (is-upEdge ?row ?col)
                (not (is-downEdge ?row ?col)))
:effect (and (is-on ?row ?col)
            (is-on ?row ?leftCol)
            (is-on ?row ?rightCol)
            (is-on ?downRow ?col) ))
```

```
(:action turn-on-downEdge-leftOff-rightOff-upOff
:parameters (?row ?col ?leftCol ?rightCol ?upRow)
:precondition (and (not (is-on ?row ?col))
                   (is-on ?row ?leftCol)
                   (is-on ?row ?rightCol)
                   (is-on ?upRow ?col)
                   (next-row ?upRow ?row)
                   (next-column ?leftCol ?col)
                   (next-column ?col ?rightCol)
                   (not (is-leftEdge ?row ?col))
                   (not (is-rightEdge ?row ?col))
                   (not (is-upEdge ?row ?col))
                   (is-downEdge ?row ?col))
:effect (and (is-on ?row ?col)
             (not (is-on ?row ?leftCol))
             (not (is-on ?row ?rightCol))
             (not (is-on ?upRow ?col)) ))

(:action turn-on-downEdge-leftOn-rightOff-upOff
:parameters (?row ?col ?leftCol ?rightCol ?upRow)
:precondition (and (not (is-on ?row ?col))
                   (not (is-on ?row ?leftCol))
                   (is-on ?row ?rightCol)
                   (is-on ?upRow ?col)
                   (next-row ?upRow ?row)
                   (next-column ?leftCol ?col)
                   (next-column ?col ?rightCol)
                   (not (is-leftEdge ?row ?col))
                   (not (is-rightEdge ?row ?col))
                   (not (is-upEdge ?row ?col))
                   (is-downEdge ?row ?col))
:effect (and (is-on ?row ?col)
             (is-on ?row ?leftCol)
             (not (is-on ?row ?rightCol))
             (not (is-on ?upRow ?col))))

(:action turn-on-downEdge-leftOff-rightOn-upOff
:parameters (?row ?col ?leftCol ?rightCol ?upRow)
:precondition (and (not (is-on ?row ?col))
                   (is-on ?row ?leftCol)
                   (not (is-on ?row ?rightCol))
                   (is-on ?upRow ?col)
                   (next-row ?upRow ?row)
                   (next-column ?leftCol ?col)
                   (next-column ?col ?rightCol)
                   (not (is-leftEdge ?row ?col))
                   (not (is-rightEdge ?row ?col))
                   (not (is-upEdge ?row ?col))
                   (is-downEdge ?row ?col))
```

```
:effect (and (is-on ?row ?col)
             (not (is-on ?row ?leftCol))
             (is-on ?row ?rightCol)
             (not (is-on ?upRow ?col))))

(:action turn-on-downEdge-leftOff-rightOff-upOn
:parameters (?row ?col ?leftCol ?rightCol ?upRow)
:precondition (and (not (is-on ?row ?col))
                   (is-on ?row ?leftCol)
                   (is-on ?row ?rightCol)
                   (not (is-on ?upRow ?col))
                   (next-row ?upRow ?row)
                   (next-column ?leftCol ?col)
                   (next-column ?col ?rightCol)
                   (not (is-leftEdge ?row ?col))
                   (not (is-rightEdge ?row ?col))
                   (not (is-upEdge ?row ?col))
                   (is-downEdge ?row ?col))
:effect (and (is-on ?row ?col)
             (not (is-on ?row ?leftCol))
             (not (is-on ?row ?rightCol))
             (is-on ?upRow ?col)))

(:action turn-on-downEdge-leftOn-rightOn-upOff
:parameters (?row ?col ?leftCol ?rightCol ?upRow)
:precondition (and (not (is-on ?row ?col))
                   (not (is-on ?row ?leftCol))
                   (not (is-on ?row ?rightCol))
                   (is-on ?upRow ?col)
                   (next-row ?upRow ?row)
                   (next-column ?leftCol ?col)
                   (next-column ?col ?rightCol)
                   (not (is-leftEdge ?row ?col))
                   (not (is-rightEdge ?row ?col))
                   (not (is-upEdge ?row ?col))
                   (is-downEdge ?row ?col))
:effect (and (is-on ?row ?col)
             (is-on ?row ?leftCol)
             (is-on ?row ?rightCol)
             (not (is-on ?upRow ?col))))

(:action turn-on-downEdge-leftOn-rightOff-upOn
:parameters (?row ?col ?leftCol ?rightCol ?upRow)
:precondition (and (not (is-on ?row ?col))
                   (not (is-on ?row ?leftCol))
                   (is-on ?row ?rightCol)
                   (not (is-on ?upRow ?col))
                   (next-row ?upRow ?row)
                   (next-column ?leftCol ?col)
                   (next-column ?col ?rightCol)
```

```
                (not (is-leftEdge ?row ?col))
                (not (is-rightEdge ?row ?col))
                (not (is-upEdge ?row ?col))
                (is-downEdge ?row ?col))
:effect (and (is-on ?row ?col)
           (is-on ?row ?leftCol)
           (not (is-on ?row ?rightCol))
           (is-on ?upRow ?col)))

(:action turn-on-downEdge-leftOff-rightOn-upOn
:parameters (?row ?col ?leftCol ?rightCol ?upRow)
:precondition (and (not (is-on ?row ?col))
                (is-on ?row ?leftCol)
                (not (is-on ?row ?rightCol))
                (not (is-on ?upRow ?col))
                (next-row ?upRow ?row)
                (next-column ?leftCol ?col)
                (next-column ?col ?rightCol)
                (not (is-leftEdge ?row ?col))
                (not (is-rightEdge ?row ?col))
                (not (is-upEdge ?row ?col))
                (is-downEdge ?row ?col))
:effect (and (is-on ?row ?col)
           (not (is-on ?row ?leftCol))
           (is-on ?row ?rightCol)
           (is-on ?upRow ?col)))

(:action turn-on-downEdge-leftOn-rightOn-upOn
:parameters (?row ?col ?leftCol ?rightCol ?upRow)
:precondition (and (not (is-on ?row ?col))
                (not (is-on ?row ?leftCol))
                (not (is-on ?row ?rightCol))
                (not (is-on ?upRow ?col))
                (next-row ?upRow ?row)
                (next-column ?leftCol ?col)
                (next-column ?col ?rightCol)
                (not (is-leftEdge ?row ?col))
                (not (is-rightEdge ?row ?col))
                (not (is-upEdge ?row ?col))
                (is-downEdge ?row ?col))
:effect (and (is-on ?row ?col)
           (is-on ?row ?leftCol)
           (is-on ?row ?rightCol)
           (is-on ?upRow ?col)))

(:action turn-on-leftUpCorner-rightOff-downOff
:parameters (?row ?col ?rightCol ?downRow)
:precondition (and (not (is-on ?row ?col))
                (is-on ?row ?rightCol)
                (is-on ?downRow ?col)
```

```
                (is-leftEdge ?row ?col)
                (next-row ?row ?downRow)
                (next-column ?col ?rightCol)
                (not (is-rightEdge ?row ?col))
                (is-upEdge ?row ?col)
                (not (is-downEdge ?row ?col)))
:effect (and (is-on ?row ?col)
           (not (is-on ?row ?rightCol))
           (not (is-on ?downRow ?col))))

(:action turn-on-leftUpCorner-rightOn-downOff
:parameters (?row ?col ?rightCol ?downRow)
:precondition (and (not (is-on ?row ?col))
                (not (is-on ?row ?rightCol))
                (is-on ?downRow ?col)
                (next-row ?row ?downRow)
                (next-column ?col ?rightCol)
                (is-leftEdge ?row ?col)
                (not (is-rightEdge ?row ?col))
                (is-upEdge ?row ?col)
                (not (is-downEdge ?row ?col)))
:effect (and (is-on ?row ?col)
           (is-on ?row ?rightCol)
           (not (is-on ?downRow ?col))))

(:action turn-on-leftUpCorner-rightOff-downOn
:parameters (?row ?col ?rightCol ?downRow)
:precondition (and (not (is-on ?row ?col))
                (is-on ?row ?rightCol)
                (not (is-on ?downRow ?col))
                (next-row ?row ?downRow)
                (next-column ?col ?rightCol)
                (is-leftEdge ?row ?col)
                (not (is-rightEdge ?row ?col))
                (is-upEdge ?row ?col)
                (not (is-downEdge ?row ?col)))
:effect (and (is-on ?row ?col)
           (not (is-on ?row ?rightCol))
           (is-on ?downRow ?col)))

(:action turn-on-leftUpCorner-rightOn-downOn
:parameters (?row ?col ?rightCol ?downRow)
:precondition (and (not (is-on ?row ?col))
                (not (is-on ?row ?rightCol))
                (not (is-on ?downRow ?col))
                (next-row ?row ?downRow)
                (next-column ?col ?rightCol)
                (is-leftEdge ?row ?col)
                (not (is-rightEdge ?row ?col))
                (is-upEdge ?row ?col)
```

```
                (not (is-downEdge ?row ?col)))
:effect (and (is-on ?row ?col)
            (is-on ?row ?rightCol)
            (is-on ?downRow ?col)))

(:action turn-on-leftdownCorner-rightOff-upOff
:parameters (?row ?col ?rightCol ?upRow)
:precondition (and (not (is-on ?row ?col))
                (is-on ?row ?rightCol)
                (is-on ?upRow ?col)
                (next-row ?upRow ?row)
                (next-column ?col ?rightCol)
                (is-leftEdge ?row ?col)
                (not (is-rightEdge ?row ?col))
                (not (is-upEdge ?row ?col))
                (is-downEdge ?row ?col))
:effect (and (is-on ?row ?col)
            (not (is-on ?row ?rightCol))
            (not (is-on ?upRow ?col)) ))


(:action turn-on-leftdownCorner-rightOn-upOff
:parameters (?row ?col ?rightCol ?upRow)
:precondition (and (not (is-on ?row ?col))
                (not (is-on ?row ?rightCol))
                (is-on ?upRow ?col)
                (next-row ?upRow ?row)
                (next-column ?col ?rightCol)
                (is-leftEdge ?row ?col)
                (not (is-rightEdge ?row ?col))
                (not (is-upEdge ?row ?col))
                (is-downEdge ?row ?col))
:effect (and (is-on ?row ?col)
            (is-on ?row ?rightCol)
            (not (is-on ?upRow ?col)) ))

(:action turn-on-leftdownCorner-rightOff-upOn
:parameters (?row ?col ?rightCol ?upRow)
:precondition (and (not (is-on ?row ?col))
                (is-on ?row ?rightCol)
                (not (is-on ?upRow ?col))
                (next-row ?upRow ?row)
                (next-column ?col ?rightCol)
                (is-leftEdge ?row ?col)
                (not (is-rightEdge ?row ?col))
                (not (is-upEdge ?row ?col))
                (is-downEdge ?row ?col))
:effect (and (is-on ?row ?col)
            (not (is-on ?row ?rightCol))
            (is-on ?upRow ?col) ))
```

```
(:action turn-on-leftdownCorner-rightOn-upOn
:parameters (?row ?col ?rightCol ?upRow)
:precondition (and (not (is-on ?row ?col))
                   (not (is-on ?row ?rightCol))
                   (not (is-on ?upRow ?col))
                   (next-row ?upRow ?row)
                   (next-column ?col ?rightCol)
                   (is-leftEdge ?row ?col)
                   (not (is-rightEdge ?row ?col))
                   (not (is-upEdge ?row ?col))
                   (is-downEdge ?row ?col))
:effect (and (is-on ?row ?col)
             (is-on ?row ?rightCol)
             (is-on ?upRow ?col) ))

(:action turn-on-rightUpCorner-leftOff-downOff
:parameters (?row ?col ?leftCol ?downRow)
:precondition (and (not (is-on ?row ?col))
                   (is-on ?row ?leftCol)
                   (is-on ?downRow ?col)
                   (next-row ?row ?downRow)
                   (next-column ?leftCol ?col)
                   (not (is-leftEdge ?row ?col))
                   (is-rightEdge ?row ?col)
                   (is-upEdge ?row ?col)
                   (not (is-downEdge ?row ?col)))
:effect (and (is-on ?row ?col)
             (not (is-on ?row ?leftCol))
             (not (is-on ?downRow ?col))))

(:action turn-on-rightUpCorner-leftOn-downOff
:parameters (?row ?col ?leftCol ?downRow)
:precondition (and (not (is-on ?row ?col))
                   (not (is-on ?row ?leftCol))
                   (is-on ?downRow ?col)
                   (next-row ?row ?downRow)
                   (next-column ?leftCol ?col)
                   (not (is-leftEdge ?row ?col))
                   (is-rightEdge ?row ?col)
                   (is-upEdge ?row ?col)
                   (not (is-downEdge ?row ?col)))
:effect (and (is-on ?row ?col)
             (is-on ?row ?leftCol)
             (not (is-on ?downRow ?col)) ))

(:action turn-on-rightUpCorner-leftOff-downOn
:parameters (?row ?col ?leftCol ?downRow)
:precondition (and (not (is-on ?row ?col))
                   (is-on ?row ?leftCol)
```

```
                (not (is-on ?downRow ?col))
                (next-row ?row ?downRow)
                (next-column ?leftCol ?col)
                (not (is-leftEdge ?row ?col))
                (is-rightEdge ?row ?col)
                (is-upEdge ?row ?col)
                (not (is-downEdge ?row ?col)))
:effect (and (is-on ?row ?col)
           (not (is-on ?row ?leftCol))
           (is-on ?downRow ?col) ))

(:action turn-on-rightUpCorner-leftOn-downOn
:parameters (?row ?col ?leftCol ?downRow)
:precondition (and (not (is-on ?row ?col))
                (not (is-on ?row ?leftCol))
                (not (is-on ?downRow ?col))
                (next-row ?row ?downRow)
                (next-column ?leftCol ?col)
                (not (is-leftEdge ?row ?col))
                (is-rightEdge ?row ?col)
                (is-upEdge ?row ?col)
                (not (is-downEdge ?row ?col)))
:effect (and (is-on ?row ?col)
           (is-on ?row ?leftCol)
           (is-on ?downRow ?col) ))

(:action turn-on-rightDownCorner-leftOff-upOff
:parameters (?row ?col ?leftCol ?upRow)
:precondition (and (not (is-on ?row ?col))
                (is-on ?row ?leftCol)
                (is-on ?upRow ?col)
                (next-row ?upRow ?row)
                (next-column ?leftCol ?col)
                (not (is-leftEdge ?row ?col))
                (is-rightEdge ?row ?col)
                (not (is-upEdge ?row ?col))
                (is-downEdge ?row ?col))
:effect (and (is-on ?row ?col)
           (not (is-on ?row ?leftCol))
           (not (is-on ?upRow ?col)) ))

(:action turn-on-rightDownCorner-leftOn-upOff
:parameters (?row ?col ?leftCol ?upRow)
:precondition (and (not (is-on ?row ?col))
                (not (is-on ?row ?leftCol))
                (is-on ?upRow ?col)
                (next-row ?upRow ?row)
                (next-column ?leftCol ?col)
                (not (is-leftEdge ?row ?col))
                (is-rightEdge ?row ?col)
```

```
                    (not (is-upEdge ?row ?col))
                    (is-downEdge ?row ?col))
  :effect (and (is-on ?row ?col)
               (is-on ?row ?leftCol)
               (not (is-on ?upRow ?col)) ))

  (:action turn-on-rightDownCorner-leftOff-upOn
  :parameters (?row ?col ?leftCol ?upRow)
  :precondition (and (not (is-on ?row ?col))
                    (is-on ?row ?leftCol)
                    (not (is-on ?upRow ?col))
                    (next-row ?upRow ?row)
                    (next-column ?leftCol ?col)
                    (not (is-leftEdge ?row ?col))
                    (is-rightEdge ?row ?col)
                    (not (is-upEdge ?row ?col))
                    (is-downEdge ?row ?col))
  :effect (and (is-on ?row ?col)
               (not (is-on ?row ?leftCol))
               (is-on ?upRow ?col) ))

  (:action turn-on-rightDownCorner-leftOn-upOn
  :parameters (?row ?col ?leftCol ?upRow)
  :precondition (and (not (is-on ?row ?col))
                    (not (is-on ?row ?leftCol))
                    (not (is-on ?upRow ?col))
                    (next-row ?upRow ?row)
                    (next-column ?leftCol ?col)
                    (not (is-leftEdge ?row ?col))
                    (is-rightEdge ?row ?col)
                    (not (is-upEdge ?row ?col))
                    (is-downEdge ?row ?col))
  :effect (and (is-on ?row ?col)
               (is-on ?row ?leftCol)
               (is-on ?upRow ?col) ))



(:action turn-off-center-leftOff-rightOff-upOff-downOff
  :parameters (?row ?col ?leftCol ?rightCol ?upRow ?downRow)
  :precondition (and (is-on ?row ?col)
                    (is-on ?row ?leftCol)
                    (is-on ?row ?rightCol)
                    (is-on ?upRow ?col)
                    (is-on ?downRow ?col)
                    (next-row ?upRow ?row)
                    (next-row ?row ?downRow)
                    (next-column ?leftCol ?col)
                    (next-column ?col ?rightCol)
                    (not (is-leftEdge ?row ?col))
```

```
                (not (is-rightEdge ?row ?col))
                (not (is-upEdge ?row ?col))
                (not (is-downEdge ?row ?col)))
:effect (and (not (is-on ?row ?col))
             (not (is-on ?row ?leftCol))
             (not (is-on ?row ?rightCol))
             (not (is-on ?upRow ?col))
             (not (is-on ?downRow ?col))))

(:action turn-off-center-leftOn-rightOff-upOff-downOff
:parameters (?row ?col ?leftCol ?rightCol ?upRow ?downRow)
:precondition (and (is-on ?row ?col)
                (not (is-on ?row ?leftCol))
                (is-on ?row ?rightCol)
                (is-on ?upRow ?col)
                (is-on ?downRow ?col)
                (next-row ?upRow ?row)
                (next-row ?row ?downRow)
                (next-column ?leftCol ?col)
                (next-column ?col ?rightCol)
                (not (is-leftEdge ?row ?col))
                (not (is-rightEdge ?row ?col))
                (not (is-upEdge ?row ?col))
                (not (is-downEdge ?row ?col)))
:effect (and (not (is-on ?row ?col))
             (is-on ?row ?leftCol)
             (not (is-on ?row ?rightCol))
             (not (is-on ?upRow ?col))
             (not (is-on ?downRow ?col))))

(:action turn-off-center-leftOff-rightOn-upOff-downOff
:parameters (?row ?col ?leftCol ?rightCol ?upRow ?downRow)
:precondition (and (is-on ?row ?col)
                (is-on ?row ?leftCol)
                (not (is-on ?row ?rightCol))
                (is-on ?upRow ?col)
                (is-on ?downRow ?col)
                (next-row ?upRow ?row)
                (next-row ?row ?downRow)
                (next-column ?leftCol ?col)
                (next-column ?col ?rightCol)
                (not (is-leftEdge ?row ?col))
                (not (is-rightEdge ?row ?col))
                (not (is-upEdge ?row ?col))
                (not (is-downEdge ?row ?col)))
:effect (and (not (is-on ?row ?col))
             (not (is-on ?row ?leftCol))
             (is-on ?row ?rightCol)
             (not (is-on ?upRow ?col))
             (not (is-on ?downRow ?col))))
```

```
(:action turn-off-center-leftOff-rightOff-upOn-downOff
:parameters (?row ?col ?leftCol ?rightCol ?upRow ?downRow)
:precondition (and (is-on ?row ?col)
                   (is-on ?row ?leftCol)
                   (is-on ?row ?rightCol)
                   (not (is-on ?upRow ?col))
                   (is-on ?downRow ?col)
                   (next-row ?upRow ?row)
                   (next-row ?row ?downRow)
                   (next-column ?leftCol ?col)
                   (next-column ?col ?rightCol)
                   (not (is-leftEdge ?row ?col))
                   (not (is-rightEdge ?row ?col))
                   (not (is-upEdge ?row ?col))
                   (not (is-downEdge ?row ?col)))
:effect (and (not (is-on ?row ?col))
             (not (is-on ?row ?leftCol))
             (not (is-on ?row ?rightCol))
             (is-on ?upRow ?col)
             (not (is-on ?downRow ?col)) ))

(:action turn-off-center-leftOff-rightOff-upOff-downOn
:parameters (?row ?col ?leftCol ?rightCol ?upRow ?downRow)
:precondition (and (is-on ?row ?col)
                   (is-on ?row ?leftCol)
                   (is-on ?row ?rightCol)
                   (is-on ?upRow ?col)
                   (not (is-on ?downRow ?col))
                   (next-row ?upRow ?row)
                   (next-row ?row ?downRow)
                   (next-column ?leftCol ?col)
                   (next-column ?col ?rightCol)
                   (not (is-leftEdge ?row ?col))
                   (not (is-rightEdge ?row ?col))
                   (not (is-upEdge ?row ?col))
                   (not (is-downEdge ?row ?col)))
:effect (and (not (is-on ?row ?col))
             (not (is-on ?row ?leftCol))
             (not (is-on ?row ?rightCol))
             (not (is-on ?upRow ?col))
             (is-on ?downRow ?col)))

(:action turn-off-center-leftOn-rightOn-upOff-downOff
:parameters (?row ?col ?leftCol ?rightCol ?upRow ?downRow)
:precondition (and (is-on ?row ?col)
                   (not (is-on ?row ?leftCol))
                   (not (is-on ?row ?rightCol))
                   (is-on ?upRow ?col)
                   (is-on ?downRow ?col)
```

```
                    (next-row ?upRow ?row)
                    (next-row ?row ?downRow)
                    (next-column ?leftCol ?col)
                    (next-column ?col ?rightCol)
                    (not (is-leftEdge ?row ?col))
                    (not (is-rightEdge ?row ?col))
                    (not (is-upEdge ?row ?col))
                    (not (is-downEdge ?row ?col)))
    :effect (and (not (is-on ?row ?col))
                (is-on ?row ?leftCol)
                (is-on ?row ?rightCol)
                (not (is-on ?upRow ?col))
                (not (is-on ?downRow ?col))))

    (:action turn-off-center-leftOn-rightOff-upOn-downOff
    :parameters (?row ?col ?leftCol ?rightCol ?upRow ?downRow)
    :precondition (and (is-on ?row ?col)
                    (not (is-on ?row ?leftCol))
                    (is-on ?row ?rightCol)
                    (not (is-on ?upRow ?col))
                    (is-on ?downRow ?col)
                    (next-row ?upRow ?row)
                    (next-row ?row ?downRow)
                    (next-column ?leftCol ?col)
                    (next-column ?col ?rightCol)
                    (not (is-leftEdge ?row ?col))
                    (not (is-rightEdge ?row ?col))
                    (not (is-upEdge ?row ?col))
                    (not (is-downEdge ?row ?col)))
    :effect (and (not (is-on ?row ?col))
                (is-on ?row ?leftCol)
                (not (is-on ?row ?rightCol))
                (is-on ?upRow ?col)
                (not (is-on ?downRow ?col))))

    (:action turn-off-center-leftOn-rightOff-upOff-downOn
    :parameters (?row ?col ?leftCol ?rightCol ?upRow ?downRow)
    :precondition (and (is-on ?row ?col)
                    (not (is-on ?row ?leftCol))
                    (is-on ?row ?rightCol)
                    (is-on ?upRow ?col)
                    (not (is-on ?downRow ?col))
                    (next-row ?upRow ?row)
                    (next-row ?row ?downRow)
                    (next-column ?leftCol ?col)
                    (next-column ?col ?rightCol)
                    (not (is-leftEdge ?row ?col))
                    (not (is-rightEdge ?row ?col))
                    (not (is-upEdge ?row ?col))
                    (not (is-downEdge ?row ?col)))
```

```
:effect (and (not (is-on ?row ?col))
             (is-on ?row ?leftCol)
             (not (is-on ?row ?rightCol))
             (not (is-on ?upRow ?col))
             (is-on ?downRow ?col)))

(:action turn-off-center-leftOff-rightOn-upOn-downOff
:parameters (?row ?col ?leftCol ?rightCol ?upRow ?downRow)
:precondition (and (is-on ?row ?col)
                   (is-on ?row ?leftCol)
                   (not (is-on ?row ?rightCol))
                   (not (is-on ?upRow ?col))
                   (is-on ?downRow ?col)
                   (next-row ?upRow ?row)
                   (next-row ?row ?downRow)
                   (next-column ?leftCol ?col)
                   (next-column ?col ?rightCol)
                   (not (is-leftEdge ?row ?col))
                   (not (is-rightEdge ?row ?col))
                   (not (is-upEdge ?row ?col))
                   (not (is-downEdge ?row ?col)))
:effect (and (not (is-on ?row ?col))
             (not (is-on ?row ?leftCol))
             (is-on ?row ?rightCol)
             (is-on ?upRow ?col)
             (not (is-on ?downRow ?col))))

(:action turn-off-center-leftOff-rightOn-upOff-downOn
:parameters (?row ?col ?leftCol ?rightCol ?upRow ?downRow)
:precondition (and (is-on ?row ?col)
                   (is-on ?row ?leftCol)
                   (not (is-on ?row ?rightCol))
                   (is-on ?upRow ?col)
                   (not (is-on ?downRow ?col))
                   (next-row ?upRow ?row)
                   (next-row ?row ?downRow)
                   (next-column ?leftCol ?col)
                   (next-column ?col ?rightCol)
                   (not (is-leftEdge ?row ?col))
                   (not (is-rightEdge ?row ?col))
                   (not (is-upEdge ?row ?col))
                   (not (is-downEdge ?row ?col)))
:effect (and (not (is-on ?row ?col))
             (not (is-on ?row ?leftCol))
             (is-on ?row ?rightCol)
             (not (is-on ?upRow ?col))
             (is-on ?downRow ?col)))

(:action turn-off-center-leftOff-rightOff-upOn-downOn
:parameters (?row ?col ?leftCol ?rightCol ?upRow ?downRow)
```

```
:precondition (and (is-on ?row ?col)
                   (is-on ?row ?leftCol)
                   (is-on ?row ?rightCol)
                   (not (is-on ?upRow ?col))
                   (not (is-on ?downRow ?col))
                   (next-row ?upRow ?row)
                   (next-row ?row ?downRow)
                   (next-column ?leftCol ?col)
                   (next-column ?col ?rightCol)
                   (not (is-leftEdge ?row ?col))
                   (not (is-rightEdge ?row ?col))
                   (not (is-upEdge ?row ?col))
                   (not (is-downEdge ?row ?col)))
:effect (and (not (is-on ?row ?col))
             (not (is-on ?row ?leftCol))
             (not (is-on ?row ?rightCol))
             (is-on ?upRow ?col)
             (is-on ?downRow ?col)))

(:action turn-off-center-leftOn-rightOn-upOn-downOff
:parameters (?row ?col ?leftCol ?rightCol ?upRow ?downRow)
:precondition (and (is-on ?row ?col)
                   (not (is-on ?row ?leftCol))
                   (not (is-on ?row ?rightCol))
                   (not (is-on ?upRow ?col))
                   (is-on ?downRow ?col)
                   (next-row ?upRow ?row)
                   (next-row ?row ?downRow)
                   (next-column ?leftCol ?col)
                   (next-column ?col ?rightCol)
                   (not (is-leftEdge ?row ?col))
                   (not (is-rightEdge ?row ?col))
                   (not (is-upEdge ?row ?col))
                   (not (is-downEdge ?row ?col)))
:effect (and (not (is-on ?row ?col))
             (is-on ?row ?leftCol)
             (is-on ?row ?rightCol)
             (is-on ?upRow ?col)
             (not (is-on ?downRow ?col))))

(:action turn-off-center-leftOn-rightOn-upOff-downOn
:parameters (?row ?col ?leftCol ?rightCol ?upRow ?downRow)
:precondition (and (is-on ?row ?col)
                   (not (is-on ?row ?leftCol))
                   (not (is-on ?row ?rightCol))
                   (is-on ?upRow ?col)
                   (not (is-on ?downRow ?col))
                   (next-row ?upRow ?row)
                   (next-row ?row ?downRow)
                   (next-column ?leftCol ?col)
```

```
                (next-column ?col ?rightCol)
                (not (is-leftEdge ?row ?col))
                (not (is-rightEdge ?row ?col))
                (not (is-upEdge ?row ?col))
                (not (is-downEdge ?row ?col)))
:effect (and (not (is-on ?row ?col))
             (is-on ?row ?leftCol)
             (is-on ?row ?rightCol)
             (not (is-on ?upRow ?col))
             (is-on ?downRow ?col)))

(:action turn-off-center-leftOn-rightOff-upOn-downOn
:parameters (?row ?col ?leftCol ?rightCol ?upRow ?downRow)
:precondition (and (is-on ?row ?col)
                (not (is-on ?row ?leftCol))
                (is-on ?row ?rightCol)
                (not (is-on ?upRow ?col))
                (not (is-on ?downRow ?col))
                (next-row ?upRow ?row)
                (next-row ?row ?downRow)
                (next-column ?leftCol ?col)
                (next-column ?col ?rightCol)
                (not (is-leftEdge ?row ?col))
                (not (is-rightEdge ?row ?col))
                (not (is-upEdge ?row ?col))
                (not (is-downEdge ?row ?col)))
:effect (and (not (is-on ?row ?col))
             (is-on ?row ?leftCol)
             (not (is-on ?row ?rightCol))
             (is-on ?upRow ?col)
             (is-on ?downRow ?col)))

(:action turn-off-center-leftOff-rightOn-upOn-downOn
:parameters (?row ?col ?leftCol ?rightCol ?upRow ?downRow)
:precondition (and (is-on ?row ?col)
                (is-on ?row ?leftCol)
                (not (is-on ?row ?rightCol))
                (not (is-on ?upRow ?col))
                (not (is-on ?downRow ?col))
                (next-row ?upRow ?row)
                (next-row ?row ?downRow)
                (next-column ?leftCol ?col)
                (next-column ?col ?rightCol)
                (not (is-leftEdge ?row ?col))
                (not (is-rightEdge ?row ?col))
                (not (is-upEdge ?row ?col))
                (not (is-downEdge ?row ?col)))
:effect (and (not (is-on ?row ?col))
             (not (is-on ?row ?leftCol))
             (is-on ?row ?rightCol)
```

```
            (is-on ?upRow ?col)
            (is-on ?downRow ?col)))

   (:action turn-off-center-leftOn-rightOn-upOn-downOn
:parameters (?row ?col ?leftCol ?rightCol ?upRow ?downRow)
:precondition (and (is-on ?row ?col)
                   (not (is-on ?row ?leftCol))
                   (not (is-on ?row ?rightCol))
                   (not (is-on ?upRow ?col))
                   (not (is-on ?downRow ?col))
                   (next-row ?upRow ?row)
                   (next-row ?row ?downRow)
                   (next-column ?leftCol ?col)
                   (next-column ?col ?rightCol)
                   (not (is-leftEdge ?row ?col))
                   (not (is-rightEdge ?row ?col))
                   (not (is-upEdge ?row ?col))
                   (not (is-downEdge ?row ?col)))
:effect (and (not (is-on ?row ?col))
            (is-on ?row ?leftCol)
            (is-on ?row ?rightCol)
            (is-on ?upRow ?col)
            (is-on ?downRow ?col)))

(:action turn-off-leftEdge-rightOff-upOff-downOff
:parameters (?row ?col ?rightCol ?upRow ?downRow)
:precondition (and (is-on ?row ?col)
                   (is-on ?row ?rightCol)
                   (is-on ?upRow ?col)
                   (is-on ?downRow ?col)
                   (next-row ?upRow ?row)
                   (next-row ?row ?downRow)
                   (next-column ?col ?rightCol)
                   (is-leftEdge ?row ?col)
                   (not (is-rightEdge ?row ?col))
                   (not (is-upEdge ?row ?col))
                   (not (is-downEdge ?row ?col)))
:effect (and (not (is-on ?row ?col))
            (not (is-on ?row ?rightCol))
            (not (is-on ?upRow ?col))
            (not (is-on ?downRow ?col))))

(:action turn-off-leftEdge-rightOn-upOff-downOff
:parameters (?row ?col ?rightCol ?upRow ?downRow)
:precondition (and (is-on ?row ?col)
                   (not (is-on ?row ?rightCol))
                   (is-on ?upRow ?col)
                   (is-on ?downRow ?col)
                   (next-row ?upRow ?row)
                   (next-row ?row ?downRow)
```

```
                (next-column ?col ?rightCol)
                (is-leftEdge ?row ?col)
                (not (is-rightEdge ?row ?col))
                (not (is-upEdge ?row ?col))
                (not (is-downEdge ?row ?col)))
:effect (and (not (is-on ?row ?col))
           (is-on ?row ?rightCol)
           (not (is-on ?upRow ?col))
           (not (is-on ?downRow ?col))))

(:action turn-off-leftEdge-rightOff-upOn-downOff
:parameters (?row ?col ?rightCol ?upRow ?downRow)
:precondition (and (is-on ?row ?col)
                (is-on ?row ?rightCol)
                (not (is-on ?upRow ?col))
                (is-on ?downRow ?col)
                (next-row ?upRow ?row)
                (next-row ?row ?downRow)
                (next-column ?col ?rightCol)
                (is-leftEdge ?row ?col)
                (not (is-rightEdge ?row ?col))
                (not (is-upEdge ?row ?col))
                (not (is-downEdge ?row ?col)))
:effect (and (not (is-on ?row ?col))
           (not (is-on ?row ?rightCol))
           (is-on ?upRow ?col)
           (not (is-on ?downRow ?col))))

(:action turn-off-leftEdge-rightOff-upOff-downOn
:parameters (?row ?col ?rightCol ?upRow ?downRow)
:precondition (and (is-on ?row ?col)
                (is-on ?row ?rightCol)
                (is-on ?upRow ?col)
                (not (is-on ?downRow ?col))
                (next-row ?upRow ?row)
                (next-row ?row ?downRow)
                (next-column ?col ?rightCol)
                (is-leftEdge ?row ?col)
                (not (is-rightEdge ?row ?col))
                (not (is-upEdge ?row ?col))
                (not (is-downEdge ?row ?col)))
:effect (and (not (is-on ?row ?col))
           (not (is-on ?row ?rightCol))
           (not (is-on ?upRow ?col))
           (is-on ?downRow ?col)))

(:action turn-off-leftEdge-rightOn-upOn-downOff
:parameters (?row ?col ?rightCol ?upRow ?downRow)
:precondition (and (is-on ?row ?col)
                (not (is-on ?row ?rightCol))
```

```
                (not (is-on ?upRow ?col))
                (is-on ?downRow ?col)
                (next-row ?upRow ?row)
                (next-row ?row ?downRow)
                (next-column ?col ?rightCol)
                (is-leftEdge ?row ?col)
                (not (is-rightEdge ?row ?col))
                (not (is-upEdge ?row ?col))
                (not (is-downEdge ?row ?col)))
:effect (and (not (is-on ?row ?col))
             (is-on ?row ?rightCol)
             (is-on ?upRow ?col)
             (not (is-on ?downRow ?col))))

(:action turn-off-leftEdge-rightOn-upOff-downOn
:parameters (?row ?col ?rightCol ?upRow ?downRow)
:precondition (and (is-on ?row ?col)
                (not (is-on ?row ?rightCol))
                (is-on ?upRow ?col)
                (not (is-on ?downRow ?col))
                (next-row ?upRow ?row)
                (next-row ?row ?downRow)
                (next-column ?col ?rightCol)
                (is-leftEdge ?row ?col)
                (not (is-rightEdge ?row ?col))
                (not (is-upEdge ?row ?col))
                (not (is-downEdge ?row ?col)))
:effect (and (not (is-on ?row ?col))
             (is-on ?row ?rightCol)
             (not (is-on ?upRow ?col))
             (is-on ?downRow ?col)))

(:action turn-off-leftEdge-rightOff-upOn-downOn
:parameters (?row ?col ?rightCol ?upRow ?downRow)
:precondition (and (is-on ?row ?col)
                (is-on ?row ?rightCol)
                (not (is-on ?upRow ?col))
                (not (is-on ?downRow ?col))
                (next-row ?upRow ?row)
                (next-row ?row ?downRow)
                (next-column ?col ?rightCol)
                (is-leftEdge ?row ?col)
                (not (is-rightEdge ?row ?col))
                (not (is-upEdge ?row ?col))
                (not (is-downEdge ?row ?col)))
:effect (and (not (is-on ?row ?col))
             (not (is-on ?row ?rightCol))
             (is-on ?upRow ?col)
             (is-on ?downRow ?col)))
```

```
(:action turn-off-leftEdge-rightOn-upOn-downOn
:parameters (?row ?col ?rightCol ?upRow ?downRow)
:precondition (and (is-on ?row ?col)
                   (not (is-on ?row ?rightCol))
                   (not (is-on ?upRow ?col))
                   (not (is-on ?downRow ?col))
                   (next-row ?upRow ?row)
                   (next-row ?row ?downRow)
                   (next-column ?col ?rightCol)
                   (is-leftEdge ?row ?col)
                   (not (is-rightEdge ?row ?col))
                   (not (is-upEdge ?row ?col))
                   (not (is-downEdge ?row ?col)))
:effect (and (not (is-on ?row ?col))
             (is-on ?row ?rightCol)
             (is-on ?upRow ?col)
             (is-on ?downRow ?col)))

(:action turn-off-rightEdge-leftOff-upOff-downOff
:parameters (?row ?col ?leftCol ?upRow ?downRow)
:precondition (and (is-on ?row ?col)
                   (is-on ?row ?leftCol)
                   (is-on ?upRow ?col)
                   (is-on ?downRow ?col)
                   (next-row ?upRow ?row)
                   (next-row ?row ?downRow)
                   (next-column ?leftCol ?col)
                   (not (is-leftEdge ?row ?col))
                   (is-rightEdge ?row ?col)
                   (not (is-upEdge ?row ?col))
                   (not (is-downEdge ?row ?col)))
:effect (and (not (is-on ?row ?col))
             (not (is-on ?row ?leftCol))
             (not (is-on ?upRow ?col))
             (not (is-on ?downRow ?col)) ))

(:action turn-off-rightEdge-leftOn-upOff-downOff
:parameters (?row ?col ?leftCol ?upRow ?downRow)
:precondition (and (is-on ?row ?col)
                   (not (is-on ?row ?leftCol))
                   (is-on ?upRow ?col)
                   (is-on ?downRow ?col)
                   (next-row ?upRow ?row)
                   (next-row ?row ?downRow)
                   (next-column ?leftCol ?col)
                   (not (is-leftEdge ?row ?col))
                   (is-rightEdge ?row ?col)
                   (not (is-upEdge ?row ?col))
                   (not (is-downEdge ?row ?col)))
:effect (and (not (is-on ?row ?col))
```

```
            (is-on ?row ?leftCol)
            (not (is-on ?upRow ?col))
            (not (is-on ?downRow ?col))))

(:action turn-off-rightEdge-leftOff-upOn-downOff
:parameters (?row ?col ?leftCol ?upRow ?downRow)
:precondition (and (is-on ?row ?col)
                (is-on ?row ?leftCol)
                (not (is-on ?upRow ?col))
                (is-on ?downRow ?col)
                (next-row ?upRow ?row)
                (next-row ?row ?downRow)
                (next-column ?leftCol ?col)
                (not (is-leftEdge ?row ?col))
                (is-rightEdge ?row ?col)
                (not (is-upEdge ?row ?col))
                (not (is-downEdge ?row ?col)))
:effect (and (not (is-on ?row ?col))
            (not (is-on ?row ?leftCol))
            (is-on ?upRow ?col)
            (not (is-on ?downRow ?col)) ))

(:action turn-off-rightEdge-leftOff-upOff-downOn
:parameters (?row ?col ?leftCol ?upRow ?downRow)
:precondition (and (is-on ?row ?col)
                (is-on ?row ?leftCol)
                (is-on ?upRow ?col)
                (not (is-on ?downRow ?col))
                (next-row ?upRow ?row)
                (next-row ?row ?downRow)
                (next-column ?leftCol ?col)
                (not (is-leftEdge ?row ?col))
                (is-rightEdge ?row ?col)
                (not (is-upEdge ?row ?col))
                (not (is-downEdge ?row ?col)))
:effect (and (not (is-on ?row ?col))
            (not (is-on ?row ?leftCol))
            (not (is-on ?upRow ?col))
            (is-on ?downRow ?col) ))

(:action turn-off-rightEdge-leftOn-upOn-downOff
:parameters (?row ?col ?leftCol ?upRow ?downRow)
:precondition (and (is-on ?row ?col)
                (not (is-on ?row ?leftCol))
                (not (is-on ?upRow ?col))
                (is-on ?downRow ?col)
                (next-row ?upRow ?row)
                (next-row ?row ?downRow)
                (next-column ?leftCol ?col)
                (not (is-leftEdge ?row ?col))
```

```
                (is-rightEdge ?row ?col)
                (not (is-upEdge ?row ?col))
                (not (is-downEdge ?row ?col)))
:effect (and (not (is-on ?row ?col))
           (is-on ?row ?leftCol)
           (is-on ?upRow ?col)
           (not (is-on ?downRow ?col)) ))

(:action turn-off-rightEdge-leftOn-upOff-downOn
:parameters (?row ?col ?leftCol ?upRow ?downRow)
:precondition (and (is-on ?row ?col)
                (not (is-on ?row ?leftCol))
                (is-on ?upRow ?col)
                (not (is-on ?downRow ?col))
                (next-row ?upRow ?row)
                (next-row ?row ?downRow)
                (next-column ?leftCol ?col)
                (not (is-leftEdge ?row ?col))
                (is-rightEdge ?row ?col)
                (not (is-upEdge ?row ?col))
                (not (is-downEdge ?row ?col)))
:effect (and (not (is-on ?row ?col))
           (is-on ?row ?leftCol)
           (not (is-on ?upRow ?col))
           (is-on ?downRow ?col) ))

(:action turn-off-rightEdge-leftOff-upOn-downOn
:parameters (?row ?col ?leftCol ?upRow ?downRow)
:precondition (and (is-on ?row ?col)
                (is-on ?row ?leftCol)
                (not (is-on ?upRow ?col))
                (not (is-on ?downRow ?col))
                (next-row ?upRow ?row)
                (next-row ?row ?downRow)
                (next-column ?leftCol ?col)
                (not (is-leftEdge ?row ?col))
                (is-rightEdge ?row ?col)
                (not (is-upEdge ?row ?col))
                (not (is-downEdge ?row ?col)))
:effect (and (not (is-on ?row ?col))
           (not (is-on ?row ?leftCol))
           (is-on ?upRow ?col)
           (is-on ?downRow ?col)))

(:action turn-off-rightEdge-leftOn-upOn-downOn
:parameters (?row ?col ?leftCol ?upRow ?downRow)
:precondition (and (is-on ?row ?col)
                (not (is-on ?row ?leftCol))
                (not (is-on ?upRow ?col))
                (not (is-on ?downRow ?col))
```

```
                (next-row ?upRow ?row)
                (next-row ?row ?downRow)
                (next-column ?leftCol ?col)
                (not (is-leftEdge ?row ?col))
                (is-rightEdge ?row ?col)
                (not (is-upEdge ?row ?col))
                (not (is-downEdge ?row ?col)))
:effect (and (not (is-on ?row ?col))
            (is-on ?row ?leftCol)
            (is-on ?upRow ?col)
            (is-on ?downRow ?col)))

(:action turn-off-upEdge-leftOff-rightOff-downOff
:parameters (?row ?col ?leftCol ?rightCol ?downRow)
:precondition (and (is-on ?row ?col)
                (is-on ?row ?leftCol)
                (is-on ?row ?rightCol)
                (is-on ?downRow ?col)
                (next-row ?row ?downRow)
                (next-column ?leftCol ?col)
                (next-column ?col ?rightCol)
                (not (is-leftEdge ?row ?col))
                (not (is-rightEdge ?row ?col))
                (is-upEdge ?row ?col)
                (not (is-downEdge ?row ?col)))
:effect (and (not (is-on ?row ?col))
            (not (is-on ?row ?leftCol))
            (not (is-on ?row ?rightCol))
            (not (is-on ?downRow ?col)) ))

(:action turn-off-upEdge-leftOn-rightOff-downOff
:parameters (?row ?col ?leftCol ?rightCol ?downRow)
:precondition (and (is-on ?row ?col)
                (not (is-on ?row ?leftCol))
                (is-on ?row ?rightCol)
                (is-on ?downRow ?col)
                (next-row ?row ?downRow)
                (next-column ?leftCol ?col)
                (next-column ?col ?rightCol)
                (not (is-leftEdge ?row ?col))
                (not (is-rightEdge ?row ?col))
                (is-upEdge ?row ?col)
                (not (is-downEdge ?row ?col)))
:effect (and (not (is-on ?row ?col))
            (is-on ?row ?leftCol)
            (not (is-on ?row ?rightCol))
            (not (is-on ?downRow ?col)) ))

(:action turn-off-upEdge-leftOff-rightOn-downOff
:parameters (?row ?col ?leftCol ?rightCol ?downRow)
```

```
:precondition (and (is-on ?row ?col)
                   (is-on ?row ?leftCol)
                   (not (is-on ?row ?rightCol))
                   (is-on ?downRow ?col)
                   (next-row ?row ?downRow)
                   (next-column ?leftCol ?col)
                   (next-column ?col ?rightCol)
                   (not (is-leftEdge ?row ?col))
                   (not (is-rightEdge ?row ?col))
                   (is-upEdge ?row ?col)
                   (not (is-downEdge ?row ?col)))
:effect (and (not (is-on ?row ?col))
             (not (is-on ?row ?leftCol))
             (is-on ?row ?rightCol)
             (not (is-on ?downRow ?col)) ))

(:action turn-off-upEdge-leftOff-rightOff-downOn
:parameters (?row ?col ?leftCol ?rightCol ?downRow)
:precondition (and (is-on ?row ?col)
                   (is-on ?row ?leftCol)
                   (is-on ?row ?rightCol)
                   (not (is-on ?downRow ?col))
                   (next-row ?row ?downRow)
                   (next-column ?leftCol ?col)
                   (next-column ?col ?rightCol)
                   (not (is-leftEdge ?row ?col))
                   (not (is-rightEdge ?row ?col))
                   (is-upEdge ?row ?col)
                   (not (is-downEdge ?row ?col)))
:effect (and (not (is-on ?row ?col))
             (not (is-on ?row ?leftCol))
             (not (is-on ?row ?rightCol))
             (is-on ?downRow ?col) ))


(:action turn-off-upEdge-leftOn-rightOn-downOff
:parameters (?row ?col ?leftCol ?rightCol ?downRow)
:precondition (and (is-on ?row ?col)
                   (not (is-on ?row ?leftCol))
                   (not (is-on ?row ?rightCol))
                   (is-on ?downRow ?col)
                   (next-row ?row ?downRow)
                   (next-column ?leftCol ?col)
                   (next-column ?col ?rightCol)
                   (not (is-leftEdge ?row ?col))
                   (not (is-rightEdge ?row ?col))
                   (is-upEdge ?row ?col)
                   (not (is-downEdge ?row ?col)))
:effect (and (not (is-on ?row ?col))
             (is-on ?row ?leftCol)
```

```
            (is-on ?row ?rightCol)
            (not (is-on ?downRow ?col)) ))

(:action turn-off-upEdge-leftOn-rightOff-downOn
:parameters (?row ?col ?leftCol ?rightCol ?downRow)
:precondition (and (is-on ?row ?col)
                (not (is-on ?row ?leftCol))
                (is-on ?row ?rightCol)
                (not (is-on ?downRow ?col))
                (next-row ?row ?downRow)
                (next-column ?leftCol ?col)
                (next-column ?col ?rightCol)
                (not (is-leftEdge ?row ?col))
                (not (is-rightEdge ?row ?col))
                (is-upEdge ?row ?col)
                (not (is-downEdge ?row ?col)))
:effect (and (not (is-on ?row ?col))
            (is-on ?row ?leftCol)
            (not (is-on ?row ?rightCol))
            (is-on ?downRow ?col) ))

(:action turn-off-upEdge-leftOff-rightOn-downOn
:parameters (?row ?col ?leftCol ?rightCol ?downRow)
:precondition (and (is-on ?row ?col)
                (is-on ?row ?leftCol)
                (not (is-on ?row ?rightCol))
                (not (is-on ?downRow ?col))
                (next-row ?row ?downRow)
                (next-column ?leftCol ?col)
                (next-column ?col ?rightCol)
                (not (is-leftEdge ?row ?col))
                (not (is-rightEdge ?row ?col))
                (is-upEdge ?row ?col)
                (not (is-downEdge ?row ?col)))
:effect (and (not (is-on ?row ?col))
            (not (is-on ?row ?leftCol))
            (is-on ?row ?rightCol)
            (is-on ?downRow ?col) ))

(:action turn-off-upEdge-leftOn-rightOn-downOn
:parameters (?row ?col ?leftCol ?rightCol ?downRow)
:precondition (and (is-on ?row ?col)
                (not (is-on ?row ?leftCol))
                (not (is-on ?row ?rightCol))
                (not (is-on ?downRow ?col))
                (next-row ?row ?downRow)
                (next-column ?leftCol ?col)
                (next-column ?col ?rightCol)
                (not (is-leftEdge ?row ?col))
                (not (is-rightEdge ?row ?col))
```

```
                (is-upEdge ?row ?col)
                (not (is-downEdge ?row ?col)))
:effect (and (not (is-on ?row ?col))
            (is-on ?row ?leftCol)
            (is-on ?row ?rightCol)
            (is-on ?downRow ?col) ))

(:action turn-off-downEdge-leftOff-rightOff-upOff
:parameters (?row ?col ?leftCol ?rightCol ?upRow)
:precondition (and (is-on ?row ?col)
                (is-on ?row ?leftCol)
                (is-on ?row ?rightCol)
                (is-on ?upRow ?col)
                (next-row ?upRow ?row)
                (next-column ?leftCol ?col)
                (next-column ?col ?rightCol)
                (not (is-leftEdge ?row ?col))
                (not (is-rightEdge ?row ?col))
                (not (is-upEdge ?row ?col))
                (is-downEdge ?row ?col))
:effect (and (not (is-on ?row ?col))
            (not (is-on ?row ?leftCol))
            (not (is-on ?row ?rightCol))
            (not (is-on ?upRow ?col)) ))

(:action turn-off-downEdge-leftOn-rightOff-upOff
:parameters (?row ?col ?leftCol ?rightCol ?upRow)
:precondition (and (is-on ?row ?col)
                (not (is-on ?row ?leftCol))
                (is-on ?row ?rightCol)
                (is-on ?upRow ?col)
                (next-row ?upRow ?row)
                (next-column ?leftCol ?col)
                (next-column ?col ?rightCol)
                (not (is-leftEdge ?row ?col))
                (not (is-rightEdge ?row ?col))
                (not (is-upEdge ?row ?col))
                (is-downEdge ?row ?col))
:effect (and (not (is-on ?row ?col))
            (is-on ?row ?leftCol)
            (not (is-on ?row ?rightCol))
            (not (is-on ?upRow ?col))))

(:action turn-off-downEdge-leftOff-rightOn-upOff
:parameters (?row ?col ?leftCol ?rightCol ?upRow)
:precondition (and (is-on ?row ?col)
                (is-on ?row ?leftCol)
                (not (is-on ?row ?rightCol))
                (is-on ?upRow ?col)
                (next-row ?upRow ?row)
```

```
                (next-column ?leftCol ?col)
                (next-column ?col ?rightCol)
                (not (is-leftEdge ?row ?col))
                (not (is-rightEdge ?row ?col))
                (not (is-upEdge ?row ?col))
                (is-downEdge ?row ?col))
:effect (and (not (is-on ?row ?col))
             (not (is-on ?row ?leftCol))
             (is-on ?row ?rightCol)
             (not (is-on ?upRow ?col))))

(:action turn-off-downEdge-leftOff-rightOff-upOn
:parameters (?row ?col ?leftCol ?rightCol ?upRow)
:precondition (and (is-on ?row ?col)
                (is-on ?row ?leftCol)
                (is-on ?row ?rightCol)
                (not (is-on ?upRow ?col))
                (next-row ?upRow ?row)
                (next-column ?leftCol ?col)
                (next-column ?col ?rightCol)
                (not (is-leftEdge ?row ?col))
                (not (is-rightEdge ?row ?col))
                (not (is-upEdge ?row ?col))
                (is-downEdge ?row ?col))
:effect (and (not (is-on ?row ?col))
             (not (is-on ?row ?leftCol))
             (not (is-on ?row ?rightCol))
             (is-on ?upRow ?col)))

(:action turn-off-downEdge-leftOn-rightOn-upOff
:parameters (?row ?col ?leftCol ?rightCol ?upRow)
:precondition (and (is-on ?row ?col)
                (not (is-on ?row ?leftCol))
                (not (is-on ?row ?rightCol))
                (is-on ?upRow ?col)
                (next-row ?upRow ?row)
                (next-column ?leftCol ?col)
                (next-column ?col ?rightCol)
                (not (is-leftEdge ?row ?col))
                (not (is-rightEdge ?row ?col))
                (not (is-upEdge ?row ?col))
                (is-downEdge ?row ?col))
:effect (and (not (is-on ?row ?col))
             (is-on ?row ?leftCol)
             (is-on ?row ?rightCol)
             (not (is-on ?upRow ?col))))

(:action turn-off-downEdge-leftOn-rightOff-upOn
:parameters (?row ?col ?leftCol ?rightCol ?upRow)
:precondition (and (is-on ?row ?col)
```

```
                (not (is-on ?row ?leftCol))
                (is-on ?row ?rightCol)
                (not (is-on ?upRow ?col))
                (next-row ?upRow ?row)
                (next-column ?leftCol ?col)
                (next-column ?col ?rightCol)
                (not (is-leftEdge ?row ?col))
                (not (is-rightEdge ?row ?col))
                (not (is-upEdge ?row ?col))
                (is-downEdge ?row ?col))
:effect (and (not (is-on ?row ?col))
             (is-on ?row ?leftCol)
             (not (is-on ?row ?rightCol))
             (is-on ?upRow ?col)))

(:action turn-off-downEdge-leftOff-rightOn-upOn
:parameters (?row ?col ?leftCol ?rightCol ?upRow)
:precondition (and (is-on ?row ?col)
                (is-on ?row ?leftCol)
                (not (is-on ?row ?rightCol))
                (not (is-on ?upRow ?col))
                (next-row ?upRow ?row)
                (next-column ?leftCol ?col)
                (next-column ?col ?rightCol)
                (not (is-leftEdge ?row ?col))
                (not (is-rightEdge ?row ?col))
                (not (is-upEdge ?row ?col))
                (is-downEdge ?row ?col))
:effect (and (not (is-on ?row ?col))
             (not (is-on ?row ?leftCol))
             (is-on ?row ?rightCol)
             (is-on ?upRow ?col)))

(:action turn-off-downEdge-leftOn-rightOn-upOn
:parameters (?row ?col ?leftCol ?rightCol ?upRow)
:precondition (and (is-on ?row ?col)
                (not (is-on ?row ?leftCol))
                (not (is-on ?row ?rightCol))
                (not (is-on ?upRow ?col))
                (next-row ?upRow ?row)
                (next-column ?leftCol ?col)
                (next-column ?col ?rightCol)
                (not (is-leftEdge ?row ?col))
                (not (is-rightEdge ?row ?col))
                (not (is-upEdge ?row ?col))
                (is-downEdge ?row ?col))
:effect (and (not (is-on ?row ?col))
             (is-on ?row ?leftCol)
             (is-on ?row ?rightCol)
             (is-on ?upRow ?col)))
```

```
(:action turn-off-leftUpCorner-rightOff-downOff
:parameters (?row ?col ?rightCol ?downRow)
:precondition (and (is-on ?row ?col)
                   (is-on ?row ?rightCol)
                   (is-on ?downRow ?col)
                   (next-row ?row ?downRow)
                   (next-column ?col ?rightCol)
                   (is-leftEdge ?row ?col)
                   (not (is-rightEdge ?row ?col))
                   (is-upEdge ?row ?col)
                   (not (is-downEdge ?row ?col)))
:effect (and (not (is-on ?row ?col))
             (not (is-on ?row ?rightCol))
             (not (is-on ?downRow ?col))))

(:action turn-off-leftUpCorner-rightOn-downOff
:parameters (?row ?col ?rightCol ?downRow)
:precondition (and (is-on ?row ?col)
                   (not (is-on ?row ?rightCol))
                   (is-on ?downRow ?col)
                   (next-row ?row ?downRow)
                   (next-column ?col ?rightCol)
                   (is-leftEdge ?row ?col)
                   (not (is-rightEdge ?row ?col))
                   (is-upEdge ?row ?col)
                   (not (is-downEdge ?row ?col)))
:effect (and (not (is-on ?row ?col))
             (is-on ?row ?rightCol)
             (not (is-on ?downRow ?col))))

(:action turn-off-leftUpCorner-rightOff-downOn
:parameters (?row ?col ?rightCol ?downRow)
:precondition (and (is-on ?row ?col)
                   (is-on ?row ?rightCol)
                   (not (is-on ?downRow ?col))
                   (next-row ?row ?downRow)
                   (next-column ?col ?rightCol)
                   (is-leftEdge ?row ?col)
                   (not (is-rightEdge ?row ?col))
                   (is-upEdge ?row ?col)
                   (not (is-downEdge ?row ?col)))
:effect (and (not (is-on ?row ?col))
             (not (is-on ?row ?rightCol))
             (is-on ?downRow ?col)))

(:action turn-off-leftUpCorner-rightOn-downOn
:parameters (?row ?col ?rightCol ?downRow)
:precondition (and (is-on ?row ?col)
                   (not (is-on ?row ?rightCol))
```

```
                (not (is-on ?downRow ?col))
                (next-row ?row ?downRow)
                (next-column ?col ?rightCol)
                (is-leftEdge ?row ?col)
                (not (is-rightEdge ?row ?col))
                (is-upEdge ?row ?col)
                (not (is-downEdge ?row ?col)))
:effect (and (not (is-on ?row ?col))
           (is-on ?row ?rightCol)
           (is-on ?downRow ?col)))

(:action turn-off-leftdownCorner-rightOff-upOff
:parameters (?row ?col ?rightCol ?upRow)
:precondition (and (is-on ?row ?col)
                (is-on ?row ?rightCol)
                (is-on ?upRow ?col)
                (next-row ?upRow ?row)
                (next-column ?col ?rightCol)
                (is-leftEdge ?row ?col)
                (not (is-rightEdge ?row ?col))
                (not (is-upEdge ?row ?col))
                (is-downEdge ?row ?col))
:effect (and (not (is-on ?row ?col))
           (not (is-on ?row ?rightCol))
           (not (is-on ?upRow ?col)) ))


(:action turn-off-leftdownCorner-rightOn-upOff
:parameters (?row ?col ?rightCol ?upRow)
:precondition (and (is-on ?row ?col)
                (not (is-on ?row ?rightCol))
                (is-on ?upRow ?col)
                (next-row ?upRow ?row)
                (next-column ?col ?rightCol)
                (is-leftEdge ?row ?col)
                (not (is-rightEdge ?row ?col))
                (not (is-upEdge ?row ?col))
                (is-downEdge ?row ?col))
:effect (and (not (is-on ?row ?col))
           (is-on ?row ?rightCol)
           (not (is-on ?upRow ?col)) ))

(:action turn-off-leftdownCorner-rightOff-upOn
:parameters (?row ?col ?rightCol ?upRow)
:precondition (and (is-on ?row ?col)
                (is-on ?row ?rightCol)
                (not (is-on ?upRow ?col))
                (next-row ?upRow ?row)
                (next-column ?col ?rightCol)
                (is-leftEdge ?row ?col)
```

```
                (not (is-rightEdge ?row ?col))
                (not (is-upEdge ?row ?col))
                (is-downEdge ?row ?col))
:effect (and (not (is-on ?row ?col))
           (not (is-on ?row ?rightCol))
           (is-on ?upRow ?col) ))

(:action turn-off-leftdownCorner-rightOn-upOn
:parameters (?row ?col ?rightCol ?upRow)
:precondition (and (is-on ?row ?col)
                (not (is-on ?row ?rightCol))
                (not (is-on ?upRow ?col))
                (next-row ?upRow ?row)
                (next-column ?col ?rightCol)
                (is-leftEdge ?row ?col)
                (not (is-rightEdge ?row ?col))
                (not (is-upEdge ?row ?col))
                (is-downEdge ?row ?col))
:effect (and (not (is-on ?row ?col))
           (is-on ?row ?rightCol)
           (is-on ?upRow ?col) ))

(:action turn-off-rightUpCorner-leftOff-downOff
:parameters (?row ?col ?leftCol ?downRow)
:precondition (and (is-on ?row ?col)
                (is-on ?row ?leftCol)
                (is-on ?downRow ?col)
                (next-row ?row ?downRow)
                (next-column ?leftCol ?col)
                (not (is-leftEdge ?row ?col))
                (is-rightEdge ?row ?col)
                (is-upEdge ?row ?col)
                (not (is-downEdge ?row ?col)))
:effect (and (not (is-on ?row ?col))
           (not (is-on ?row ?leftCol))
           (not (is-on ?downRow ?col))))

(:action turn-off-rightUpCorner-leftOn-downOff
:parameters (?row ?col ?leftCol ?downRow)
:precondition (and (is-on ?row ?col)
                (not (is-on ?row ?leftCol))
                (is-on ?downRow ?col)
                (next-row ?row ?downRow)
                (next-column ?leftCol ?col)
                (not (is-leftEdge ?row ?col))
                (is-rightEdge ?row ?col)
                (is-upEdge ?row ?col)
                (not (is-downEdge ?row ?col)))
:effect (and (not (is-on ?row ?col))
           (is-on ?row ?leftCol)
```

```
            (not (is-on ?downRow ?col)) ))

(:action turn-off-rightUpCorner-leftOff-downOn
:parameters (?row ?col ?leftCol ?downRow)
:precondition (and (is-on ?row ?col)
                   (is-on ?row ?leftCol)
                   (not (is-on ?downRow ?col))
                   (next-row ?row ?downRow)
                   (next-column ?leftCol ?col)
                   (not (is-leftEdge ?row ?col))
                   (is-rightEdge ?row ?col)
                   (is-upEdge ?row ?col)
                   (not (is-downEdge ?row ?col)))
:effect (and (not (is-on ?row ?col))
             (not (is-on ?row ?leftCol))
             (is-on ?downRow ?col) ))

(:action turn-off-rightUpCorner-leftOn-downOn
:parameters (?row ?col ?leftCol ?downRow)
:precondition (and (is-on ?row ?col)
                   (not (is-on ?row ?leftCol))
                   (not (is-on ?downRow ?col))
                   (next-row ?row ?downRow)
                   (next-column ?leftCol ?col)
                   (not (is-leftEdge ?row ?col))
                   (is-rightEdge ?row ?col)
                   (is-upEdge ?row ?col)
                   (not (is-downEdge ?row ?col)))
:effect (and (not (is-on ?row ?col))
             (is-on ?row ?leftCol)
             (is-on ?downRow ?col) ))

(:action turn-off-rightDownCorner-leftOff-upOff
:parameters (?row ?col ?leftCol ?upRow)
:precondition (and (is-on ?row ?col)
                   (is-on ?row ?leftCol)
                   (is-on ?upRow ?col)
                   (next-row ?upRow ?row)
                   (next-column ?leftCol ?col)
                   (not (is-leftEdge ?row ?col))
                   (is-rightEdge ?row ?col)
                   (not (is-upEdge ?row ?col))
                   (is-downEdge ?row ?col))
:effect (and (not (is-on ?row ?col))
             (not (is-on ?row ?leftCol))
             (not (is-on ?upRow ?col)) ))

(:action turn-off-rightDownCorner-leftOn-upOff
:parameters (?row ?col ?leftCol ?upRow)
:precondition (and (is-on ?row ?col)
```

```
                    (not (is-on ?row ?leftCol))
                    (is-on ?upRow ?col)
                    (next-row ?upRow ?row)
                    (next-column ?leftCol ?col)
                    (not (is-leftEdge ?row ?col))
                    (is-rightEdge ?row ?col)
                    (not (is-upEdge ?row ?col))
                    (is-downEdge ?row ?col))
    :effect (and (not (is-on ?row ?col))
                (is-on ?row ?leftCol)
                (not (is-on ?upRow ?col)) ))

    (:action turn-off-rightDownCorner-leftOff-upOn
    :parameters (?row ?col ?leftCol ?upRow)
    :precondition (and (is-on ?row ?col)
                    (is-on ?row ?leftCol)
                    (not (is-on ?upRow ?col))
                    (next-row ?upRow ?row)
                    (next-column ?leftCol ?col)
                    (not (is-leftEdge ?row ?col))
                    (is-rightEdge ?row ?col)
                    (not (is-upEdge ?row ?col))
                    (is-downEdge ?row ?col))
    :effect (and (not (is-on ?row ?col))
                (not (is-on ?row ?leftCol))
                (is-on ?upRow ?col) ))

    (:action turn-off-rightDownCorner-leftOn-upOn
    :parameters (?row ?col ?leftCol ?upRow)
    :precondition (and (is-on ?row ?col)
                    (not (is-on ?row ?leftCol))
                    (not (is-on ?upRow ?col))
                    (next-row ?upRow ?row)
                    (next-column ?leftCol ?col)
                    (not (is-leftEdge ?row ?col))
                    (is-rightEdge ?row ?col)
                    (not (is-upEdge ?row ?col))
                    (is-downEdge ?row ?col))
    :effect (and (not (is-on ?row ?col))
                (is-on ?row ?leftCol)
                (is-on ?upRow ?col) ))

)
```

A.1.1 3x3 Table Problem 1

```
(define (problem allOut3x3)
  (:domain allOut)
  (:objects
```

```
    row1 row2 row3
    col1 col2 col3)
  (:init
    (next-row row1 row2) (next-column col1 col2)
    (next-row row2 row3) (next-column col2 col3)
    (not (is-on row1 col1)) (is-leftEdge row1 col1) (is-upEdge row1 col1)
    (is-on row1 col2) (is-upEdge row1 col2)
    (not (is-on row1 col3)) (is-upEdge row1 col3) (is-rightEdge row1 col3)
    (is-on row2 col1) (is-leftEdge row2 col1)
    (is-on row2 col2)
    (is-on row2 col3) (is-rightEdge row2 col3)
    (not (is-on row3 col1)) (is-leftEdge row3 col1) (is-downEdge row3 col1)
    (is-on row3 col2) (is-downEdge row3 col2)
    (not (is-on row3 col3)) (is-rightEdge row3 col3) (is-downEdge row3 col3)


   )
  (:goal
   (and
    (not (is-on row1 col1))
    (not (is-on row1 col2))
    (not (is-on row1 col3))
    (not (is-on row2 col1))
    (not (is-on row2 col2))
    (not (is-on row2 col3))
    (not (is-on row3 col1))
    (not (is-on row3 col2))
    (not (is-on row3 col3))
   )
   )
)
```

A.1.1.1 Plan

```
(turn-off-center-leftoff-rightoff-upoff-downoff row2 col2 col1 col3 row1 row3)
```

A.1.2 3x3 Table Problem 2

```
(define (problem allOut3x3)
  (:domain allOut)
  (:objects
    row1 row2 row3
    col1 col2 col3)
  (:init
    (next-row row1 row2) (next-column col1 col2)
    (next-row row2 row3) (next-column col2 col3)

    (is-on row1 col1) (is-leftEdge row1 col1) (is-upEdge row1 col1)
    (is-on row1 col2) (is-upEdge row1 col2)
```

```
    (is-on row1 col3) (is-upEdge row1 col3) (is-rightEdge row1 col3)

    (is-on row2 col1) (is-leftEdge row2 col1)
    (is-on row2 col2)
    (is-on row2 col3) (is-rightEdge row2 col3)

    (is-on row3 col1) (is-leftEdge row3 col1) (is-downEdge row3 col1)
    (not (is-on row3 col2)) (is-downEdge row3 col2)
    (not (is-on row3 col3)) (is-rightEdge row3 col3) (is-downEdge row3 col3)


   )
  (:goal
   (and
    (not (is-on row1 col1))
    (not (is-on row1 col2))
    (not (is-on row1 col3))
    (not (is-on row2 col1))
    (not (is-on row2 col2))
    (not (is-on row2 col3))
    (not (is-on row3 col1))
    (not (is-on row3 col2))
    (not (is-on row3 col3))
   )
   )
)
```

A.1.2.1 Plan

```
(turn-off-leftedge-rightoff-upoff-downoff row2 col1 col2 row1 row3)
(turn-off-rightupcorner-leftoff-downoff row1 col3 col2 row2)
```

A.1.3 3x3 Table Problem 3

```
(define (problem allOut3x3)
  (:domain allOut)
  (:objects
     row1 row2 row3
     col1 col2 col3)
  (:init
     (next-row row1 row2) (next-column col1 col2)
     (next-row row2 row3) (next-column col2 col3)

     (not (is-on row1 col1)) (is-leftEdge row1 col1) (is-upEdge row1 col1)
     (is-on row1 col2) (is-upEdge row1 col2)
     (not (is-on row1 col3)) (is-upEdge row1 col3) (is-rightEdge row1 col3)

     (is-on row2 col1) (is-leftEdge row2 col1)
     (not (is-on row2 col2))
```

```
    (is-on row2 col3) (is-rightEdge row2 col3)

    (not (is-on row3 col1)) (is-leftEdge row3 col1) (is-downEdge row3 col1)
    (is-on row3 col2) (is-downEdge row3 col2)
    (not (is-on row3 col3)) (is-rightEdge row3 col3) (is-downEdge row3 col3)


   )
  (:goal
   (and
    (not (is-on row1 col1))
    (not (is-on row1 col2))
    (not (is-on row1 col3))
    (not (is-on row2 col1))
    (not (is-on row2 col2))
    (not (is-on row2 col3))
    (not (is-on row3 col1))
    (not (is-on row3 col2))
    (not (is-on row3 col3))
   )
   )
)
```

A.1.3.1 Plan

```
(turn-on-center-leftoff-rightoff-upoff-downoff row2 col2 col1 col3 row1 row3)
(turn-on-leftedge-rightoff-upon-downon row2 col1 col2 row1 row3)
(turn-on-rightdowncorner-lefton-upon row3 col3 col2 row2)
(turn-on-center-leftoff-rightoff-upon-downoff row2 col2 col1 col3 row1 row3)
(turn-off-upedge-leftoff-righton-downoff row1 col2 col1 col3 row2)
(turn-on-rightedge-lefton-upoff-downoff row2 col3 col2 row1 row3)
(turn-on-downedge-leftoff-righton-upoff row3 col2 col1 col3 row2)
(turn-off-rightdowncorner-leftoff-upoff row3 col3 col2 row2)
```

A.1.4 5x5 Table Problem 1

```
(define (problem allOut5x5)
  (:domain allOut)
  (:objects
     row1 row2 row3 row4 row5
     col1 col2 col3 col4 col5)
  (:init
     (next-row row1 row2) (next-column col1 col2)
     (next-row row2 row3) (next-column col2 col3)
     (next-row row3 row4) (next-column col3 col4)
     (next-row row4 row5) (next-column col4 col5)


     (not (is-on row1 col1)) (is-leftEdge row1 col1) (is-upEdge row1 col1)
```

```
    (not (is-on row1 col2)) (is-upEdge row1 col2)
    (not (is-on row1 col3)) (is-upEdge row1 col3)
    (not (is-on row1 col4)) (is-upEdge row1 col4)
    (not (is-on row1 col5)) (is-upEdge row1 col5) (is-rightEdge row1 col5)

    (not (is-on row2 col1)) (is-leftEdge row2 col1)
    (not (is-on row2 col2))
    (not (is-on row2 col3))
    (not (is-on row2 col4))
    (not (is-on row2 col5)) (is-rightEdge row2 col5)

    (not (is-on row3 col1)) (is-leftEdge row3 col1)
    (not (is-on row3 col2))
    (not (is-on row3 col3))
    (not (is-on row3 col4))
    (not (is-on row3 col5)) (is-rightEdge row3 col5)

    (not (is-on row4 col1)) (is-leftEdge row4 col1)
    (not (is-on row4 col2))
    (is-on row4 col3)
    (is-on row4 col4)
    (is-on row4 col5) (is-rightEdge row4 col5)

    (not (is-on row5 col1)) (is-leftEdge row5 col1) (is-downEdge row5 col1)
    (is-on row5 col2) (is-downEdge row5 col2)
    (not (is-on row5 col3)) (is-downEdge row5 col3)
    (is-on row5 col4) (is-downEdge row5 col4)
    (not (is-on row5 col5)) (is-downEdge row5 col5) (is-rightEdge row5 col5)


  )
 (:goal
  (and
    (not (is-on row1 col1))
    (not (is-on row1 col2))
    (not (is-on row1 col3))
    (not (is-on row1 col4))
    (not (is-on row1 col5))
    (not (is-on row2 col1))
    (not (is-on row2 col2))
    (not (is-on row2 col3))
    (not (is-on row2 col4))
    (not (is-on row2 col5))
    (not (is-on row3 col1))
    (not (is-on row3 col2))
    (not (is-on row3 col3))
    (not (is-on row3 col4))
    (not (is-on row3 col5))
    (not (is-on row4 col1))
    (not (is-on row4 col2))
```

```
    (not (is-on row4 col3))
    (not (is-on row4 col4))
    (not (is-on row4 col5))
    (not (is-on row5 col1))
    (not (is-on row5 col2))
    (not (is-on row5 col3))
    (not (is-on row5 col4))
    (not (is-on row5 col5))
   )
   )
)
```

A.1.4.1 Plan

```
(turn-on-downedge-leftoff-rightoff-upoff row5 col3 col2 col4 row4)
(turn-on-downedge-leftoff-righton-upoff row5 col4 col3 col5 row4)
(turn-off-rightdowncorner-leftoff-upoff row5 col5 col4 row4)
```

A.1.5 5x5 Table Problem 2

```
(define (problem allOut5x5_p02)
  (:domain allOut)
  (:objects
    row1 row2 row3 row4 row5
    col1 col2 col3 col4 col5)
  (:init
    (next-row row1 row2) (next-column col1 col2)
    (next-row row2 row3) (next-column col2 col3)
    (next-row row3 row4) (next-column col3 col4)
    (next-row row4 row5) (next-column col4 col5)


    (not (is-on row1 col1)) (is-leftEdge row1 col1) (is-upEdge row1 col1)
    (not (is-on row1 col2)) (is-upEdge row1 col2)
    (not (is-on row1 col3)) (is-upEdge row1 col3)
    (not (is-on row1 col4)) (is-upEdge row1 col4)
    (not (is-on row1 col5)) (is-upEdge row1 col5) (is-rightEdge row1 col5)

    (not (is-on row2 col1)) (is-leftEdge row2 col1)
    (not (is-on row2 col2))
    (is-on row2 col3)
    (not (is-on row2 col4))
    (not (is-on row2 col5)) (is-rightEdge row2 col5)

    (not (is-on row3 col1)) (is-leftEdge row3 col1)
    (is-on row3 col2)
    (not (is-on row3 col3))
    (is-on row3 col4)
    (not (is-on row3 col5)) (is-rightEdge row3 col5)
```

```
    (is-on row4 col1) (is-leftEdge row4 col1)
    (not (is-on row4 col2))
    (is-on row4 col3)
    (not (is-on row4 col4))
    (is-on row4 col5) (is-rightEdge row4 col5)

    (not (is-on row5 col1)) (is-leftEdge row5 col1) (is-downEdge row5 col1)
    (is-on row5 col2) (is-downEdge row5 col2)
    (not (is-on row5 col3)) (is-downEdge row5 col3)
    (is-on row5 col4) (is-downEdge row5 col4)
    (not (is-on row5 col5)) (is-downEdge row5 col5) (is-rightEdge row5 col5)


   )
  (:goal
   (and
    (not (is-on row1 col1))
    (not (is-on row1 col2))
    (not (is-on row1 col3))
    (not (is-on row1 col4))
    (not (is-on row1 col5))
    (not (is-on row2 col1))
    (not (is-on row2 col2))
    (not (is-on row2 col3))
    (not (is-on row2 col4))
    (not (is-on row2 col5))
    (not (is-on row3 col1))
    (not (is-on row3 col2))
    (not (is-on row3 col3))
    (not (is-on row3 col4))
    (not (is-on row3 col5))
    (not (is-on row4 col1))
    (not (is-on row4 col2))
    (not (is-on row4 col3))
    (not (is-on row4 col4))
    (not (is-on row4 col5))
    (not (is-on row5 col1))
    (not (is-on row5 col2))
    (not (is-on row5 col3))
    (not (is-on row5 col4))
    (not (is-on row5 col5))
   )
   )
)
```

A.1.5.1 Plan

```
(turn-on-center-leftoff-rightoff-upoff-downoff row3 col3 col2 col4 row2 row4)
(turn-on-rightdowncorner-leftoff-upoff row5 col5 col4 row4)
```

```
(turn-on-leftdowncorner-rightoff-upoff row5 col1 col2 row4)
(turn-on-leftedge-righton-upon-downoff row4 col1 col2 row3 row5)
(turn-on-center-leftoff-rightoff-upon-downoff row3 col2 col1 col3 row2 row4)
(turn-on-leftedge-rightoff-upon-downoff row3 col1 col2 row2 row4)
(turn-off-leftedge-rightoff-upon-downoff row2 col1 col2 row1 row3)
(turn-on-rightedge-lefton-upon-downoff row4 col5 col4 row3 row5)
(turn-off-rightedge-lefton-upon-downoff row3 col5 col4 row2 row4)
(turn-off-center-lefton-righton-upon-downoff row3 col4 col3 col5 row2 row4)
(turn-off-rightedge-leftoff-upon-downoff row2 col5 col4 row1 row3)
(turn-on-upedge-leftoff-righton-downon row1 col2 col1 col3 row2)
(turn-on-center-leftoff-righton-upoff-downoff row2 col3 col2 col4 row1 row3)
(turn-on-upedge-leftoff-righton-downoff row1 col3 col2 col4 row2)
(turn-off-upedge-leftoff-rightoff-downoff row1 col4 col3 col5 row2)
```

A.1.6 5x5 Table Problem 3

```
(define (problem allOut5x5_p03)
  (:domain allOut)
  (:objects
     row1 row2 row3 row4 row5
     col1 col2 col3 col4 col5)
  (:init
     (next-row row1 row2) (next-column col1 col2)
     (next-row row2 row3) (next-column col2 col3)
     (next-row row3 row4) (next-column col3 col4)
     (next-row row4 row5) (next-column col4 col5)


     (not (is-on row1 col1)) (is-leftEdge row1 col1) (is-upEdge row1 col1)
     (not (is-on row1 col2)) (is-upEdge row1 col2)
     (not (is-on row1 col3)) (is-upEdge row1 col3)
     (not (is-on row1 col4)) (is-upEdge row1 col4)
     (not (is-on row1 col5)) (is-upEdge row1 col5) (is-rightEdge row1 col5)

     (not (is-on row2 col1)) (is-leftEdge row2 col1)
     (not (is-on row2 col2))
     (not (is-on row2 col3))
     (not (is-on row2 col4))
     (not (is-on row2 col5)) (is-rightEdge row2 col5)

     (not (is-on row3 col1)) (is-leftEdge row3 col1)
     (not (is-on row3 col2))
     (is-on row3 col3)
     (is-on row3 col4)
     (is-on row3 col5) (is-rightEdge row3 col5)

     (not (is-on row4 col1)) (is-leftEdge row4 col1)
     (not (is-on row4 col2))
     (is-on row4 col3)
```

```
    (is-on row4 col4)
    (not (is-on row4 col5)) (is-rightEdge row4 col5)

    (not (is-on row5 col1)) (is-leftEdge row5 col1) (is-downEdge row5 col1)
    (not (is-on row5 col2)) (is-downEdge row5 col2)
    (is-on row5 col3) (is-downEdge row5 col3)
    (not (is-on row5 col4)) (is-downEdge row5 col4)
    (not (is-on row5 col5)) (is-downEdge row5 col5) (is-rightEdge row5 col5)


   )
  (:goal
   (and
    (not (is-on row1 col1))
    (not (is-on row1 col2))
    (not (is-on row1 col3))
    (not (is-on row1 col4))
    (not (is-on row1 col5))
    (not (is-on row2 col1))
    (not (is-on row2 col2))
    (not (is-on row2 col3))
    (not (is-on row2 col4))
    (not (is-on row2 col5))
    (not (is-on row3 col1))
    (not (is-on row3 col2))
    (not (is-on row3 col3))
    (not (is-on row3 col4))
    (not (is-on row3 col5))
    (not (is-on row4 col1))
    (not (is-on row4 col2))
    (not (is-on row4 col3))
    (not (is-on row4 col4))
    (not (is-on row4 col5))
    (not (is-on row5 col1))
    (not (is-on row5 col2))
    (not (is-on row5 col3))
    (not (is-on row5 col4))
    (not (is-on row5 col5))
   )
   )
)
```

A.1.6.1 Plan

```
(turn-off-center-lefton-rightoff-upoff-downoff row4 col3 col2 col4 row3 row5)
(turn-off-rightedge-leftoff-upon-downon row3 col5 col4 row2 row4)
(turn-on-rightupcorner-lefton-downoff row1 col5 col4 row2)
(turn-off-upedge-lefton-rightoff-downon row1 col4 col3 col5 row2)
(turn-on-center-lefton-rightoff-upoff-downon row2 col3 col2 col4 row1 row3)
(turn-on-center-lefton-rightoff-upoff-downoff row3 col2 col1 col3 row2 row4)
```

```
(turn-off-leftedge-rightoff-upon-downon row3 col1 col2 row2 row4)
(turn-on-leftupcorner-righton-downoff row1 col1 col2 row2)
(turn-on-rightdowncorner-lefton-upoff row5 col5 col4 row4)
(turn-off-leftupcorner-rightoff-downon row1 col1 col2 row2)
(turn-on-leftdowncorner-righton-upoff row5 col1 col2 row4)
(turn-off-downedge-leftoff-righton-upon row5 col2 col1 col3 row4)
(turn-off-downedge-leftoff-rightoff-upon row5 col4 col3 col5 row4)
(turn-on-center-leftoff-rightoff-upon-downon row4 col3 col2 col4 row3 row5)
(turn-off-center-lefton-righton-upoff-downoff row3 col3 col2 col4 row2 row4)
(turn-on-leftedge-rightoff-upoff-downon row3 col1 col2 row2 row4)
(turn-off-leftedge-righton-upoff-downon row4 col1 col2 row3 row5)
(turn-on-downedge-leftoff-rightoff-upoff row5 col2 col1 col3 row4)
(turn-on-downedge-leftoff-righton-upon row5 col3 col2 col4 row4)
(turn-on-center-leftoff-righton-upoff-downoff row4 col4 col3 col5 row3 row5)
(turn-on-downedge-leftoff-righton-upoff row5 col4 col3 col5 row4)
(turn-off-rightdowncorner-leftoff-upoff row5 col5 col4 row4)
```