# Numeric Klotski game

## 1. Introduction

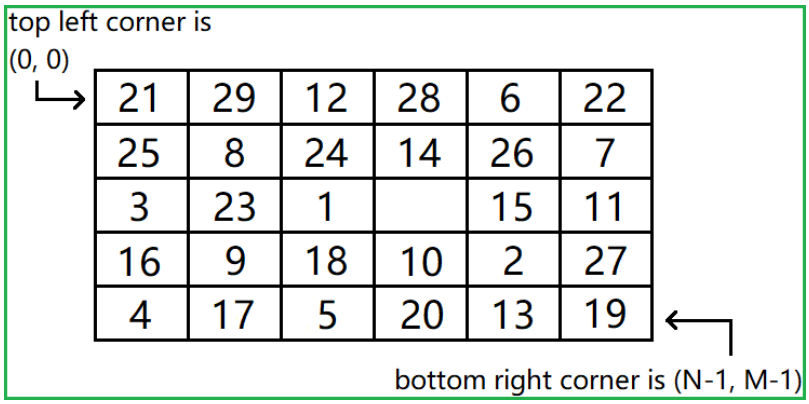
### 1.1 Original Klotski

Klotski is an ancient Chinese folk puzzle game, which, together with Rubik's cube and independent diamond, is called "the three incredible puzzles in the field of puzzle games" by some experts. In this project, you are going to design a Numeric Klotski game. You should write a program to automate the game.

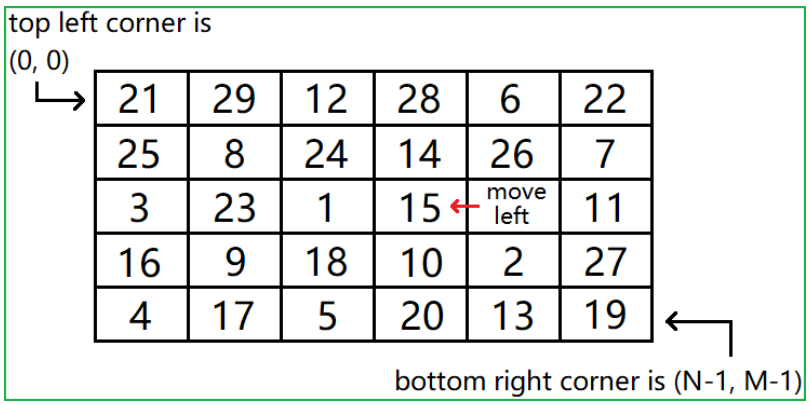
The original numeric Klotski game is as follows:

1. The game is played on a $N$ x $M$ grid, the grid is numbered from $1$ to $N \times M - 1$ and there is an empty grid. The grids are shuffled.
2. The player can moves a number to its adjacent grid (a grid is adjacent to the four closest grids that's on its left, right, up and down directions) if that grid is emtpy.
3. The player keep doing the above until all the numbers are in order (ascending from left to right, from top to bottom, row major) and the empty grid is on the right bottom corner. Then the player wins.

Here's an example
the init state:




move one number to adjacent grid:



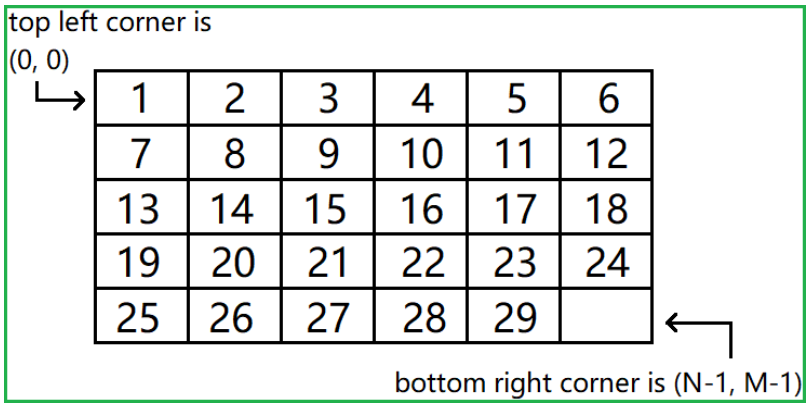

many steps here.....

...

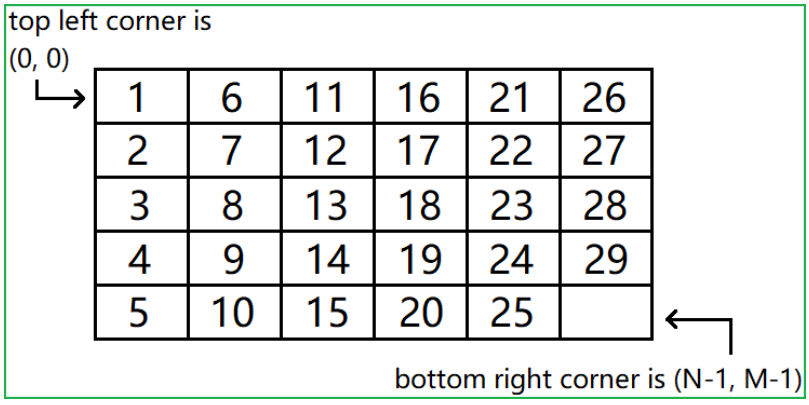
...

...

final state:




Note that the state below is **NOT** a proper final state, since it is not row major.



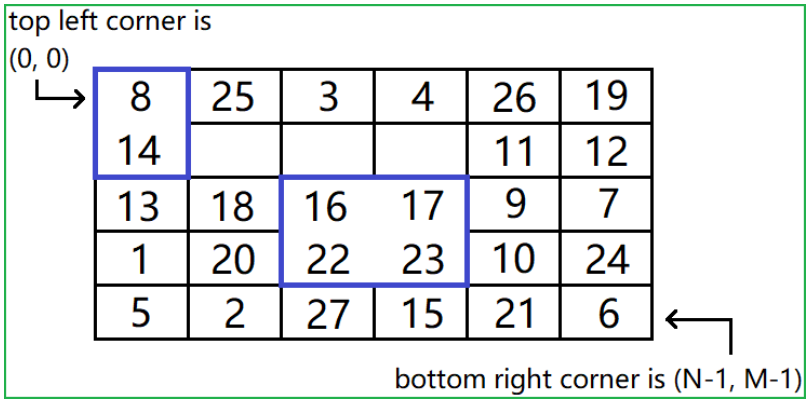

## 1.2 Modifications

In this project, the Klotski you need to implement is slightly different:

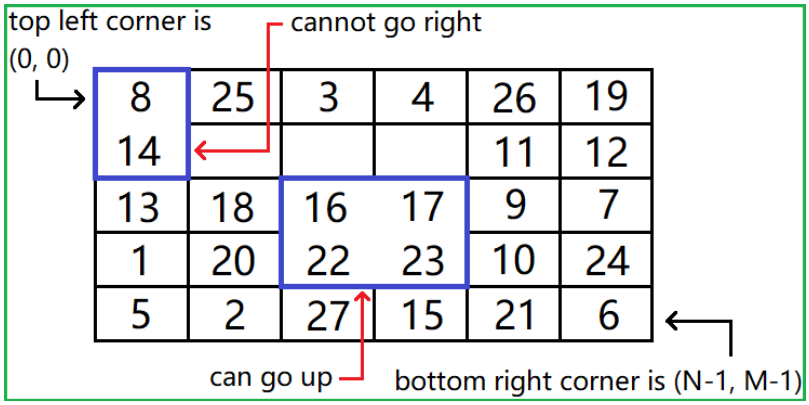
1. There could be more than one empty grid.
2. Some numbers may be tied together with its neighbors to form larger blocks like 2x2 blocks, 1x2 blocks, 2x1 blocks.
3. When moving a number, if it is tied to some other numbers, then they should move together. If a 2x2 block need to go up, for example, then there should be two empty blocks above.

4. The player wins in the same way as before: put all numbers in ascending order, from left to right, top to bottom, followed by the empty spaces.

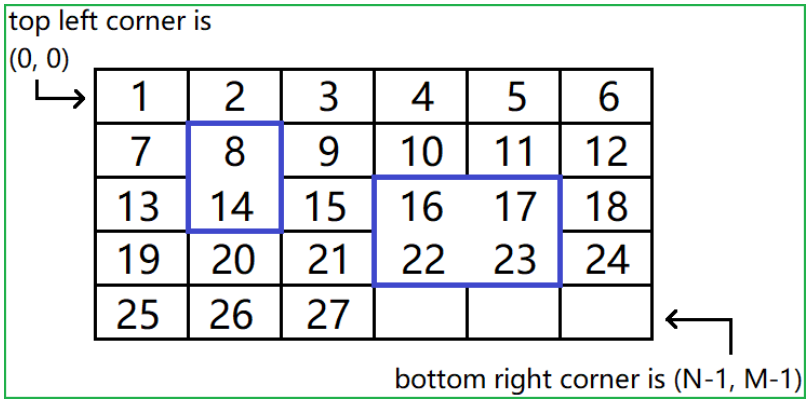
Here's an example:
An initial state, note that 8 and 14 are combined to form a 2x1 block. And 16, 17, 22, 23 are combined to form a 2x2 block.




In this situation the 2x1 block containing 8 and 14 cannot go right because number 25 stops them. The 2x2 block contains 16, 17, 22, 23 can go up.

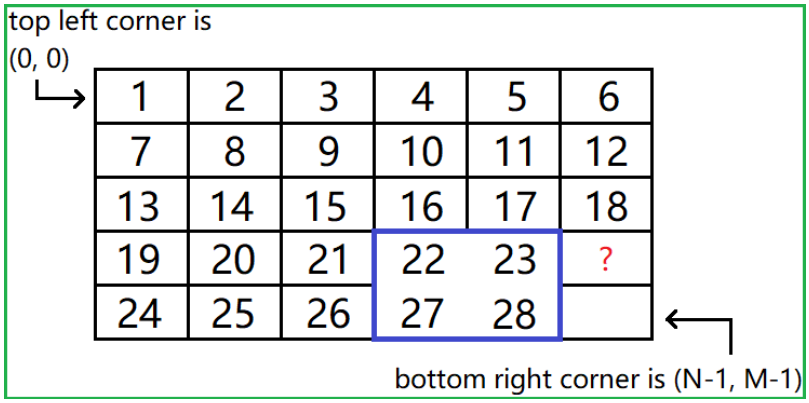



Moving all the pieces and reach the final state as below(if possible), then the player wins.




However, the following is **NOT** a final state that leads to win since an empty block (the one with the question mark) appears before the number 24. The empty blocks should

appear after the last number.




In this project, when numbers are tied together, they may form 2x2 block (4 numbers), 2x1 block (2 numbers), 1x2 block (2 numbers). Other blocks are single numbers. There will not be blocks of other shape.

Your program should display the result using a graphic window. This window displays the movement of every number at each step.

***Your program shoule read initial state from standard input.***
***Your program shoule read initial state from standard input.***
***Your program shoule read initial state from standard input.***

## 2. Input and Output

When the program starts running, you need to ***pass an argument to the program***.

There are 2 choices of the argument: `terminal` and `gui`.

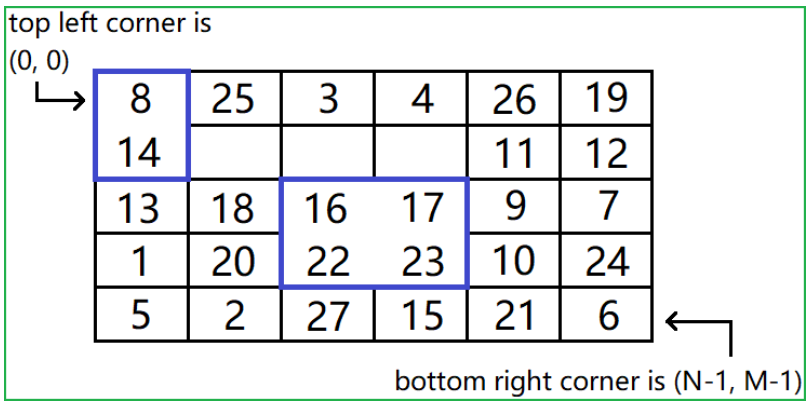
`terminal` means you should print the required information in the terminal.

`gui` means you should show a window.

### Sample Input

```
5 6
8 25 3 4 26 19
14 0 0 0 11 12
13 18 16 17 9 7
1 20 22 23 10 24
5 2 27 15 21 6
2
8 2*1
16 2*2
```

The sample represents a game scenario as shown in this figure.




The two numbers `n` and `m` in the second line represent the number of rows and columns on the chessboard.

For each of the next `n` lines, there are `m` numbers represent the initial state of the chessboard.

- The number indicates the value of the location.
- `0` means the blank space.

There are no repeated values and the value ranges from `1` to `n*m-num(blank_space)`.

The blank space(s) do not always appear in the last position. It can appear in any position.

Then, the number `w` in the line n+2 represents the number of squares whose size is not 1*1.

For each of the next `w` lines, there are a number and a String representing the state of the squares whose size is not 1*1.

The number represents the top left number in the square.

The String represents the size of the square. `2*1` means this square has 2 rows and 1 column.

## Sample Output

```
Yes
step_num
x S
……
```

or

```
No
```

You need to decide whether the problem is solvable.

If you find a solution, output `Yes` and the number `step_num` of steps required.

Then it prints the `step_num` lines with a number and a letter.

The number `x` represents the number that was moved in that step (If a block large than 1x1 needs to move, then prints the top left number).

In other words, if a solution exists, we use the smallest number `x` in each square to represent the square, and this number indicates that the square was moved in that step.

The letter `S` only appear `L`, `R`, `U`, and `D`, representing the square moves left, right, up, and down respectively.

If the problem is unsolvable, the output is `No`.

Please follows strictly the above requirement. When we test your program (as mentioned in the last section, 30 points), we will run your program with our input data and compare your result with the standard output data. If your program does not follow the above restriction and prints something else, the comparison will fail and it will not be good.

If you have any questions about the input/output format, you can ask the student assistants or teacher.

# 3. Project Requirement

The actual work of the project contains many parts:

- Write the system described above. (50 Points)
- Write your report. (20 points)

## 3.1 Write the system described above

First of all, you should make such a system.

- (10 points) The program should be able to read from standard input the initial state.
- (10 points) The program should be able to search for a solution (in a reasonable time). Or tell us if that doesn't exist.
- (10 points) The program should be able to run in terminal mode and show result with ***standard output***.
- (10 points) The program should be able to deal with 2x2, 1x2, 2x1 blocks as well as single numbers.
- (10 points) The program should be able to run in GUI mode and show the movement of the numbers with a window. **Note:** GUI requires at least one game window where shows the current state of board, and one button which can be clicked to switch the board to the next state.

### 3.2 Write your report

Include everything you want us to know about your project. Such as the basic structure of your project, the data structures you have used in the project, the way you optimize the performance, and anything that makes your project different from others.

***Include the running result on some data in your report.*** Show us it can produce the right answer.

Clearly state how to run your program in your report. It's necessary when we test your program.

However, we have no requirement on the length of the report. Do **NOT** try to make it unnecessarily long. Remember to include group members and student numbers. Also remember to upload your report in "pdf" format.

Do **NOT** put complaint about your teammate in your report!

### 3.3 Extra test

You may noticed that the above parts add up to 70 points. In the last we will run some test we have prepared on your program. That is the last 30 points of your project. This can only be done after your code is submitted. We'll test your program by some cases. By specifying the initial state, your program's result should be the same with the standard answer. The difficulty of each test cases is different. Some examples contains only a few numbers. However, some cases contains much more than that. You can try to minimize the time spent by your algorithm. Each test case has some score, and your score depends on the number of test cases you pass. Your algorithm has to be able to show us the process through the GUI.

### There are restrictions on what you can use and cannot use in this project

You could use various tools to advance your project, but third party physics engines, game engines, AI tools are not allowed. For instance, you could use OpenGL, and OpenCL, but Unity, Unreal or Torch are not allowed to use in the project.

You are allowed to use algs4 library. If you want to use some library and you are not sure about it, you can ask.

## 4. Bonus

- Generate the solvable input state randomly. (10 points at most)
- Use different searching methods to solve this problem, compare and analyze the performance of them. Show the conclusion comprehensively to us in your own way.

(10 points at most)

**You must not copy the code from Internet or other groups' classmates, or we will deal with this situation in accordance with the CS Department plagiarism policy.**