Java-2022. HomeWork-2

26 Dec 2022, 00:42:03
start: 2 Dec 2022, 18:55:16
finish: 23 Dec 2022, 18:55:16
duration: 21d.
start: 14 Nov 2022, 14:30:00

A. Problem A. City traffic

Time limit	1 second
Memory limit	64Mb
Input	standard input or input.txt
Output	standard output or output.txt

Problem A. City traffic
There is a city map represented as a two-dimensional array of the size M * N. Each cell is either occupied or empty. A cell can be occupied by a vehicle or by a building. You are given a sample of that city map. Also one or more movements of vehicles are given. You should compute the final positions of vehicles or report that some movement is incorrect.

It is guaranteed that the map length and width are $2 < M, N < 200$. Amount of buildings is $0 < K < 50$. Amount of vehicles is $0 < T < 50$. Amount of moves is $0 < D < 1000$.

Movement rules:

- Each vehicle occupies exactly one cell. During a movement each vehicle can travel to the adjacent cell only.
- There are two types of vehicles: car and truck. Car needs one standard unit of fuel to perform one movement, a truck needs two. A vehicle cannot move if it requires more fuel than the vehicle has.
- Vehicles cannot leave the map. More formally, if (x, y) are coordinates of a vehicle, $0 < x <= M$, $0 < y <= N$ at any time.
- Two vehicles cannot occupy two adjacent cells. More formally, if (x_1, y_1), (x_2, y_2) are coordinates of two vehicles, max(|x_1 - x_2|, |y_1 - y_2|) > 1 .

Movement is said to be correct if the movement satisfies movement rules and the state that the city map enters after the movement satisfies the state of the map rules.

If some rule is violated, movement is incorrect and an exception is to be thrown.

Exception types:

- **out of bound** – vehicle left the city (x < 1 or x > M or y < 1 or y > N)
- **out of fuel** – vehicle needs more fuel to move than it has
- **road accident** – target cell is occupied by a building
- **vehicles are too close to each other** – distance between vehicles is smaller than required
- **error** – other rule is violated

It is guaranteed that **the input position is correct**.

Input format

1. M N
2. Coordinates of buildings. Each building is a rectangle. Left-bottom and right-top corners are given. So, in this line there are 4 * K numbers in the format x_1 y_1 x_2 y_2, where (x_1, y_1) – left-bottom corner, (x_2, y_2) – right-top corner. x_2 ≥ x_1, y_2 ≥ y_1.
3. Descriptions of vehicles. Each vehicle is represented using four numbers. Type (0 stands for car, 1 stands for truck), X-coordinate, Y-coordinate, amount of fuel. So, in this line there are 4 * T numbers. *For example, 0 3 1 10 1 1 3 10 means that there are two vehicles and the first one is a car with initial coordinates [3, 1] and 10 units of fuel. The second one is a truck with initial coordinates [1, 3] and also 10 units of fuel.*
4. Description of moves. Each move is represented using four numbers. x_1 y_1 x_2 y_2, where (x_1, y_1) – from, (x_2, y_2) – to. *For example, 1 3 2 3 means that the vehicle that occupies cell [1, 3] moves to the cell [2, 3].*

Output format

Type and coordinates of the vehicle that has the greatest amount of fuel remaining or string with error description (see Exception types). It is guaranteed that there is only one vehicle with the greatest amount of fuel.

For example, 0 3 2 means that in the end the vehicle with the greatest amount of fuel is a car that occupies cell [3, 2]

Input and output examples

in	out

3 3
1 1 2 2
0 3 1 10 1 1 3 10
1 3 2 3 2 3 1 3 3 1 3 2
	0 3 2

3 3
1 1 2 2 3 3 3 3
0 3 1 10 1 1 3 10
1 3 2 3 2 3 3 3 3 1 3 2
	road accident

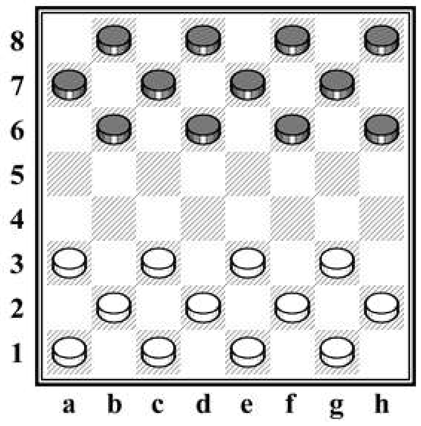
Select File is not chosen

Submit

B. Problem B. Russian checkers

Time limit	1 second
Memory limit	64Mb
Input	standard input or input.txt
Output	standard output or output.txt

Problem B. Russian checkers



A position from russian checkers (https://en.wikipedia.org/wiki/Russian_draughts) match is given. Also one or more moves description is given. You should compute the final positions of pieces or or report that some move is incorrect. In this implementation **pieces can move backwards**.

Exception types:

- **busy cell** – target cell is occupied by other piece
- **white cell** – target cell is white
- **invalid move –** player was able to capture opponent's piece but made a move without capture
- **error –** some other rule is violated

It is guaranteed that **the input position is valid**.

Notation

- The vertical columns of squares are labeled from a to h. The horizontal rows of squares are numbered from 1 to 8 starting from White's side of the board. Thus each square of the board has a unique identification of file letter followed by rank number.
- Moves without capture are denoted with dash (move from e3 to d4 is recorded as "e3-d4").
- Moves with capture are denoted with colon. (The piece from c5 captured the opponent's piece on d4 and jumped to c3 is recorded as "c5:e3"). If more than one capture is performed in one move then move is recorded using more than one colon. (for example "b6:d4:f2").
- King's moves are denoted with capital letters (for example "D4" instead of "d4").
- We don't use notes ('!' – perfect move, '?' – weak move) in our notation (for example, you will not see a description like "g3-f4??", only "g3-f4" is allowed).

Input format

1. String with coordinates of white pieces
2. String with coordinates of black pieces
3. List of moves. Each line contains a pair of moves (White's move, Black's move)

Output format

1. String with coordinates of white pieces (sorted lexicographically)
2. String with coordinates of black pieces (sorted lexicographically)
3. or string with error message

Input and output examples

in	out
a1 a3 b2 c1 c3 d2 e1 e3 f2 g1 g3 h2	
a7 b6 b8 c7 d6 d8 e7 f6 f8 g7 h6 h8	
g3 – f4 f6 – e5	
c3 – d4 e5 : c3	
b2 : d4 d6 – c5	a1 a3 b2 c3 d4 e1 e3 e5 f2 g1 h4
d2 – c3 g7 – f6	a7 b6 b8 c5 c7 d8 e7 f6 g5 g7 h6
h2 – g3 h8 – g7	
c1 – b2 f6 – g5	
g3 – h4 g7 – f6	
f4 – e5 f8 – g7	
a1 a3 b2 c1 c3 d2 e1 e3 f2 g1 g3 h2	
a7 b6 b8 c7 d6 d8 e7 f6 f8 g7 h6 h8	white cell
g3 – f5 f6 – e5	

Select File is not chosen

Submit