## ADS 1 - homework 1

On a chessboard $N \times N$ lived a limp horse. This is a chess piece that, for even moves goes as knight while odd moves as pawn. Some cells of the chessboard are blocked by walls and can not be visited.

Design an algorithm to find a shortest path between two given positions in the chessboard. Your algorithm should expect input as follows:

1. Positive integers $N$ and $M$ giving the size of the chessboard.
2. Boolean array $wall[N][M]$ determining whether given cell is a wall.
3. Starting position $AX, AY$ and ending position $BX, BY$ within the chessboard.

The output is a sequence of positions $(X_0, Y_0), (X_1, Y_1), \ldots, (X_l, Y_l)$ minimising $l$ such that $(X_0, Y_0) = (AX, AY)$ and $(X_l, Y_l) = (BX, BY)$, for every $0 \leq i < l$ even it holds that,

1. $X_i + 1 = X_i \pm 2, Y_i + 1 = Y_i \pm 1$ or
2. $X_i + 1 = X_i \pm 1, Y_i + 1 = Y_i \pm 2$,

for every $0 \leq i < l$ odd it holds that $X_i + 1 = X_i, Y_i + 1 = Y_i + 1$. Moreover for every $0 \leq j \leq l$ it holds that $1 \leq X_j \leq N$, $1 \leq Y_j \leq M$ and $wall[X_j][Y_j]$=false.

Your algorithm should also detect the case when such sequence does not exist.

Describe the algorithm (either by words, pseudocode or a common programming language of your choice), determine the time and space complexity and prove its correctness.

Please hand in your solutions before **Friday March 13** either in paper form or by email to **hubicka@kam.mff.cuni.cz**. Please add **ADS-I** into the subject.