# Escape Mines

A turtle must walk through a minefield. Write a program (console application) that will read the initial game settings and one or more sequences of moves. For each move sequence, the program will output whether the sequence leads to the success or failure of the little turtle.

The program should also handle the scenarios where the turtle doesn't reach the exit point or doesn't hit a mine.

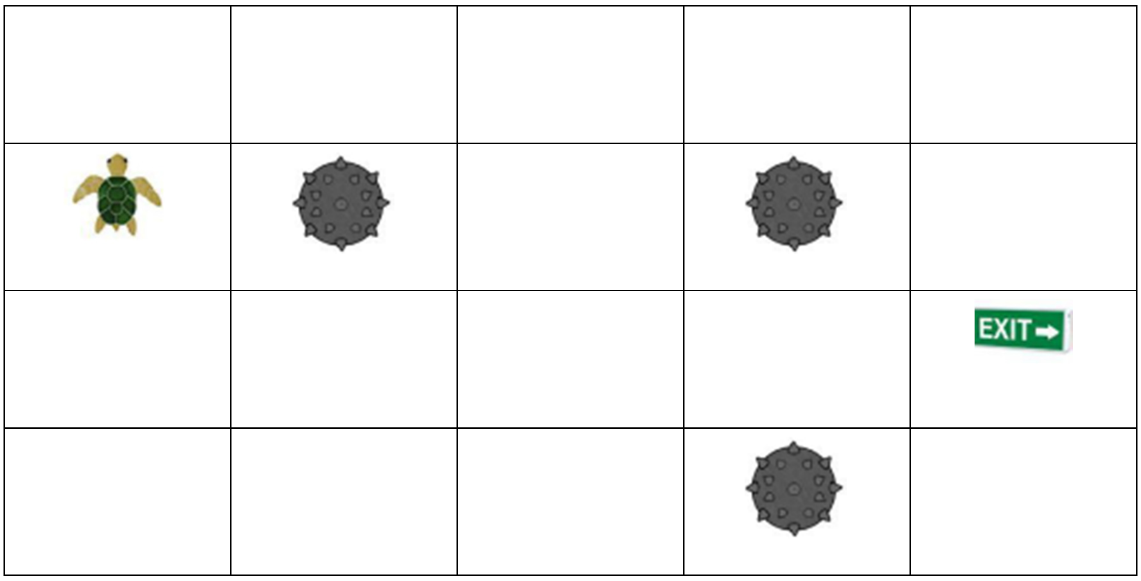
## Setup

- The board (or minefield) is a grid of N by M number of tiles.
- The starting position is a tile, represented by a set of zero based co-ordinates (x, y) and the initial direction (i.e.: N, S, W or E).
- The exit point is a tile (x, y)
- The mines are defined as a list of tiles (x, y)

Example 5 X 4 board:

Starting position: x=1, y=0 dir=N

Exit point: x=2, y=4



## Inputs

The game settings are to be loaded from a text file, which should follow this format:

- The first line should define the board size
- The second line should contain a list of mines (i.e. list of co-ordinates separated by a space)
- The third line of the file should contain the exit point.
- The fourth line of the file should contain the starting position of the turtle.
- The fifth line to the end of the file should contain a series of moves.

| Example: | Where |
|---|---|
| 5 4<br>1,1 1,3 3,3<br>4 2<br>0 1 N<br>R M L M M<br>R M M M | • R = Rotate 90 degrees to the right<br>• L = Rotate 90 degrees to the left<br>• N = North direction<br>• S = South direction<br>• W = West direction<br>• E = East direction<br>• M = Move |

Turtle actions can be either:

- A move to the next neighbouring tile
- A rotation (90 degrees Right or Left)

## Results

Results can be:

- Success – if the turtle finds the exit point
- Mine Hit – if the turtle hits a mine
- Still in Danger – it the turtle has not yet found the exit or hit a mine

## Deliverables

You are required to submit a working solution written in C# .Net (or .Net Core) modelling the above problem. The submitted solution will be assessed upon the following points:

- Ability to keep the code simple and readable
- Ability to decompose the problem into appropriate models. This includes modularity and separation of concerns.
- Ability to use the most appropriate structures that would guarantee optimal performance even on a larger scale
- Proper validations – where deemed necessary
- Proper unit testing

Plus points:

- Avoid the use of mutation
- Aim for high unit test coverage

Any use of 3rd party frameworks, other than for unit testing purposes, is strongly discouraged.