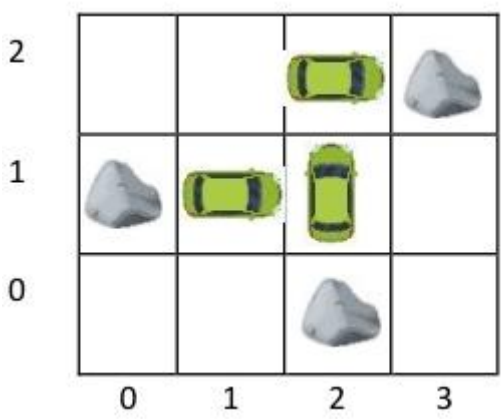
Problem Description



The diagram above presents a problem with a parking lot represented as an M×N grid (4×3 in the example), which can contain:
a) obstacles, like in cell (2,0)
b) cars with a North/South orientation, like in cell (2,1)
c) cars with an East/West orientation, like in cell (1,1).

The goal of this problem is to clear the parking lot of all cars, leaving all cells empty (except, of course, those that contain obstacles).

The allowed moves in this problem are:

1. Move a North/South car one step north
2. Move a North/South car one step south
3. Move an East/West car one step east
4. Move an East/West car one step west

Obviously, for a move to be possible, the destination cell must be empty (free of obstacles and other cars).
For example, the car in cell (2,1) cannot move north because there is another car in cell (2,2). Likewise, it cannot move south because cell (2,0) contains an obstacle. However, the car in cell (2,2) can move west since cell (1,2) is empty.

You are asked to implement in C++:
a) the modeling of the above problem
b) the Breadth-First Search algorithm to solve the problem