Assignment 6 - Rearranging cars

If anything in the assignment is unclear, you have two options. You can ask for clarifications in the #assignments channel on Slack. It is also great if you can also make assumptions: real-world problems are always unclear, and as engineers we want to move on and make progress, even if we need to re-adjust later. If you do make assumptions, please try to identify them and document them as comments in your code.

There is a parking lot with N spaces and N-1 cars in it. Your task is to write an algorithm to rearrange the cars in a given way. Only one car can be moved at a time to the empty slot.

You can assume each parking space has a unique identifier (of your choosing) and each car has a unique identifier (of your choosing as well). The state of the parking lot is then represented by the location of each car in a parking lot.

You should first define which data structure to use to represent the state of a parking lot.

You should then define a function that, given a start state and an end state for the parking lot, generates a sequence of moves from the start state to the end state.

For example, given cars #1, #2, #3 and parking spaces A, B, C, D, if the start state is:

- car #1 in space A
- car #2 in space B
- car #3 in space D

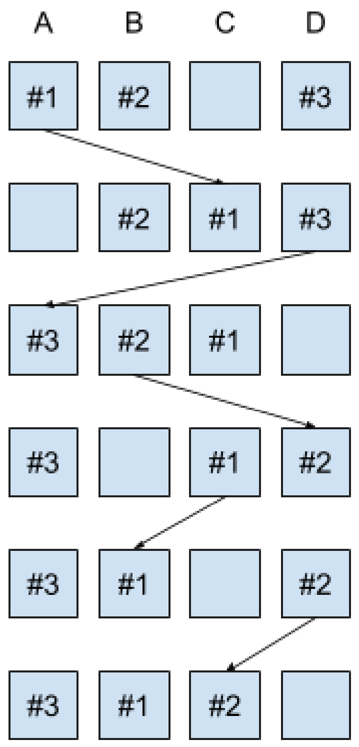
and the end state is:

- car #1 in space B
- car #2 in space C
- car #3 in space A

the following set of moves could be generated:

- move car #1 from space A to C
- move car #3 from space D to A
- move car #2 from space B to D
- move car #1 from space C to B
- move cat #2 from space D to C

Same example visually:



Hint (highlight to reveal):

Hint (highlight to reveal):

Hint (highlight to reveal):

Optional challenge #1:

Define the simplest data structure to represent the start and end states and the sequence of moves.

Optional challenge #2:

Compute the sequence of moves to go from the start to the end state with fewer moves.

Optional challenge #3:

Consider the problem of computing a sequence of moves that enforces a set of constraints, where a constraint is the fact that a certain parking place is reserved only for certain cars. For example, space D is reserved for car #1 or #2, which means that car #3 can never be parked in space D, which can only be empty, or contain car #1 or #2.

Optional challenge #4:

Compute all the possible sequence of moves that lead from the start to the end state without ever repeating the same configuration more than once.