Programming exercise

Brief

The purpose of this test is to cover general practices, design and structure as well as algorithmic solutions for a smaller project.

You can make assumptions if nothing is specifically stated, but be sure to document these.

The end result should be a command line based program that will perform simple simulations of a moving object. The program will read from `stdin` and write to `stdout` according to a certain protocol (see below).

The task

The task is to accept a set of commands and then simulate whether an object can move according to these commands without falling off the table it stands on.

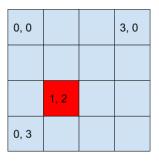The table can be seen as a matrix where the object will have an x and y position as drawn below:

Figure 1: Red box represents object at position [1, 2] on a 4 x 4 table.

The object always occupies exactly one cell and can be seen as a point without mass. Origo is at the top left.

Protocol

It is important to follow the protocol correctly so that your final output can be easily tested and verified.

First, your solution reads a header from `stdin` like this:

- The size of the table as two 16-bit little-endian integers [width, height]
- The objects starting position as two 16-bit little-endian integers [x, y]

This is followed by an arbitrarily long stream of commands of 8-bit integers.

When the simulation is complete, your program outputs the answer to `stout` as per below:

- If the simulation succeeded: The objects final position as two 16-bit little-endian integers [x, y].
- If the simulation failed (the object falls off the table): Position [-1, -1] should be returned

Commands

The object always has a direction (north, east, south or west). A simulation always starts with direction north. North means that if the object sits on [2, 4] and moves forward one step, the object will now stand on [2, 3].

The commands are:
        0 = quit simulation and print results to `stout`
        1 = move forward one step
        2 = move backwards one step
        3 = rotate clockwise 90 degrees (eg north to east)
        4 = rotate counterclockwise 90 degrees (eg west to south)

Inputs are "binary" integers, not letters. You should therefore not interpret the letter ZERO ("0") or the letter ONE ("1").

Example

If the program gets `4,4,2,2` as input, the table is initiated to size 4 x 4 with the object in position [2, 2] with direction north. Then, commands `1,4,1,3,2,3,2,4,1,0` are read from `stdin` and executed. The final output would then be the end position of the object, in this case `[0, 1]`.

Things to keep in mind

It is always possible to solve the task without any real structure, but the point here is to use a well known object oriented and/or functional architecture.

A good code structure should also allow for expanded functionality in the future. For example, would it be easy to:

- Handle a different shapes other than a rectangle
- Add more commands
- Change the binary form of the protocol to JSON

Timing

Estimated time to complete this task is around 8 hours and we need your solution before sunday the 12th of May at 23:59.