Original Description

- The application is a simulation of a toy robot moving on a square tabletop.
- There are no other obstructions on the table surface.
- The robot is free to roam around the surface of the table but must be prevented from falling to destruction. Any movement that would result in the robot falling from the table must be prevented, however further valid movement commands must still be allowed.
- Include a mechanism to report where the robot currently is on the table.
- Input can be from a file, or from standard input, as the developer chooses.
- Provide test data to exercise the application.

Original Constraints

- The toy robot must not fall off the table during movement. This also includes the initial placement of the toy robot.
- Any move that would cause the robot to fall must be ignored.

Initial Discussion / Assumptions Made:

- Toy Robot is to be Represented as "X"
- Square Table Top can be 2 x 2 grid. Empty spaces can be represented as "O". The squares in the grid could be labelled "1", "2", "3", and "4" for convenience
- Could use arrow or WASD keys to limit the kind of input the user can do. This is to prevent erroneous inputs. Or the number of the square in which they want to go to can be entered instead.
- Although it would be helpful to let the user know if their next move would cause the robot to fall although the constraints state that these moves must be ignored
- Console.WriteLine can be used to report where the robot is on the table.
- Will choose standard input
- Test data can be read from a txt file or a separate class that contains the preset values
- The user can enter "1", "2", "3", or "4" to choose which square they want to place the robot
- The description and constraints mentions that any movement that would result in the robot falling must be prevented and ignored. However there is also no requirement that the user should be presented with moves that make the toy robot fall off.
- There is also no requirement on how the robot should roam about i.e. getting input using arrow keys

Plan:

- Toy Robot to be represented as X in a 2 by 2 "square" grid. The data structure that is to store the information of the robot and the grid does not have to be square like (but will be displayed as if it were a square)
- The square grid will have numbers i.e. 1 for the top left, 2 for the top right, 3 for the bottom left and 4 for the bottom right
- Only "1", "2", "3", "4" are valid commands for moving / initialising the position of the robot as there is no constraint about how the robot can move and what commands it should take. This also avoids any movement that would cause it to fall off. There is also no requirement that there should be movements that can make the robot fall off.

- Log the grid number that the robot is currently positioned on for reporting.
- Create a console application where the user can start the robot simulation and move the robot, end the simulation, and run some test data on it
- Create unit tests for important functionality i.e. displaying the current data, and moving the robot

Final Implementation Notes:

- A console application is made which contains the main menu, the game itself (i.e. moving the robot), and running a “test” situation where inputs are “simulated
- When the user enters the game mode, the game session class is initiated. The game session contains the game itself as well as the necessary logic to take input from the user. This contains the important functionality of the game such as moving the robot, displaying the current state of the table, and reporting where the robot is.
- The game class itself also contains the table data i.e. information about which spots are occupied and not occupied by the robot
- The methods in the Game class are unit tested
- The PlayGame method in the game session class has one unit test that simulates a real interaction between the user and the application.