Task:** Implement a simple robotic control algorithm.

Task: Implement a simple robotic control algorithm.

Description:
You have a simulated robot with differential drive, equipped with two wheels. The robot is placed in a grid-based environment with dimensions 10 by 10. The task is to write a program that takes in commands from a user interface (e.g., keyboard inputs) and moves the robot accordingly. The robot can move forward, backward, turn left, and turn right. The environment is represented as a grid, where the robot can move in four directions (up, down, left, right), and it cannot move beyond the grid boundaries.

Requirements:

- Write a function/method called `move_robot` that takes parameters:
  - `commands`: a list of characters representing the commands (F for forward, B for backward, L for turn left, R for turn right).
  - `initial_position`: a tuple `(x, y, orientation)` representing the initial position of the robot, where `(x, y)` are coordinates on the grid and `orientation` is one of the cardinal directions (N, S, E, W).
- The function should return the final position of the robot after executing all the commands.
- You can assume the grid has predefined dimensions of 10 by 10.
- The robot should execute the commands sequentially, updating its position based on each command.
- The solution can be delivered in any programming language, but Python or C++ is preferred.
- The solution should ideally but not necessarily be delivered to work on a Linux based operating system. You can use a Virtual Machine for development.

User Interface Instructions:

- Implement a user interface for inputting commands (e.g., keyboard inputs).
- After entering the initial position `(x, y, orientation)` and the list of commands, display the final position of the robot.

Example:

```
# Example usage of the function
initial_position = (0, 0, 'N')
commands = ['F', 'R', 'F', 'L', 'B']
```

```
final_position = move_robot(commands, initial_position)
print(final_position)  # Output: (1, 1, 'N')
```

Evaluation Criteria:

- Correctness of the implementation: Does the robot move correctly according to the given commands?
- Handling boundary conditions: Does the robot stay within the grid boundaries?
- Efficiency of the algorithm: Is the solution optimized?
- User interface implementation: Is the user interface intuitive and functional?
- Code readability and style: Is the code well-structured and easy to understand?

Submission Instructions:

- Provide the code solution along with instructions on how to compile/run it.
- Clearly specify how to input commands and initial position through the user interface.