Task 01: Move the Robot Forward

First Coding Task: Move the Robot Forward

Now that everyone is set up in GitHub Codespaces, it's time to dive into your first coding task!

Objective

Write a function in Rust that sends a command to move the robot forward in the virtual maze. This function will interact with the "black box" API that simulates the robot's motor.

Detailed Code Explanation

The function you'll write is structured as follows:

```rust
    // This function is public, meaning it can be accessed from other modules.
    pub fn move_forward(speed: u8) {
        // 'speed' is a parameter that sets how fast the robot moves forward.
        // 'u8' is an unsigned 8-bit integer, allowing values from 0 to 255.

        // Here, you'll replace this comment with code that sends a command
        // to the black box API, instructing the virtual robot to move forward.
    }
```

Listing 1: Rust code for moving the robot forward

What the Code Does:

- `pub fn move_forward(speed: u8)`: This line defines a **public function** in Rust, meaning it can be called from other parts of the program.
- `speed: u8`: The function takes a parameter called `speed`, which represents the speed at which the robot should move forward. The `u8` type stands for an **unsigned 8-bit integer**, meaning it can take values from 0 to 255.
- The function body currently contains a comment that indicates where you should implement the code for moving the robot forward.

What You Should Do to Complete Task 01

To complete this task, you need to add logic inside the `move_forward` function to interact with the "black box" API. Follow these steps:

1. **Understand the API Interaction**: The "black box API" acts as a simulated motor controller, translating commands (e.g., moving forward, turning) into robot actions. Your task is to write Rust code that calls this API to make the virtual robot move forward.
2. **Modify the `move_forward` Function**: Replace the placeholder comment inside the function with code that sends a forward command to the black box API, using the given `speed` parameter.

3. **Example Implementation**: If the API has a function called `send_command`, you could implement the `move_forward` function as follows:

```rust
pub fn move_forward(speed: u8) {
    // Simulate sending a command to the black box API to move forward
    send_command("MOVE_FORWARD", speed);
}

// This is a mock function representing the black box API command sender
fn send_command(command: &str, value: u8) {
    // In a real implementation, this would send a command to the simulated motor
    println!("Sending command: {} with speed {}", command, value);
}
```

Listing 2: Example implementation of move_forward

How Students Should Test Their Code

After completing the `move_forward` function, students should test it by following these steps:

1. **Add the function to the main file** (`src/main.rs`): Open the `src/main.rs` file and include the `move_forward` function definition.

2. **Call the function in the `main()` function**: Add the following code to the `main()` function:

```rust
fn main() {
    move_forward(5); // Example call to move the robot forward at speed 5
}
```

Listing 3: Calling move_forward in the main function

3. **Run the code in the terminal**:
   - In the top menu of the Codespace, click on the **Terminal** menu and select **New Terminal**.
   - In the terminal, navigate to the project directory (if not already there) by typing:

   ```
   cd ~/workspace/virtual-robot-maze
   ```

   - Compile and run your Rust code using the following command:

   ```
   cargo run
   ```

4. **Check the output**: If the implementation is correct, the terminal should display a message like:

```
Sending command: MOVE_FORWARD with speed 5
```

Tips

- Focus on getting the basic functionality working first. Adjust the speed parameter and observe how it changes the robot's behavior.
- If you encounter errors, read the error messages carefully—they often provide clues about what went wrong.
- Remember that the goal is to make the robot move forward in the virtual maze, so don't hesitate to experiment with the function's logic.