ROS & Robotics Assessment

Instructions:

1. **Answer all questions within this document.**
2. **For coding questions:**
   - Include the complete code snippets within the document.
   - Include screenshots of the terminal output and Gazebo simulation (if applicable).
     - **Screenshot Guidelines:**
       - Place the screenshot of the terminal output on the **left side** of the page.
       - Place the screenshot of the Gazebo simulation (if applicable) on the **right side** of the page.
   - **Ensure clear and concise explanations for your code and observations.**
3. **Submit this completed document through the provided Google Form.**
4. **Time Limit:** This assessment has a strict time limit of **24 hours**.

Questions: 1. Simple Publisher/Subscribe

- **Scenario:** You are tasked with creating a basic ROS communication system between two nodes.
- **Objective:**
  - Create a ROS node (publisher_node.py) that publishes a string message to a topic named "/robot_status" at a rate of 10 Hz. The message should be: "Robot is running."
  - Create another ROS node (subscriber_node.py) that subscribes to the "/robot_status" topic and prints the received messages to the console.
- **Task:**
  - Write the complete Python code for both the publisher_node.py and subscriber_node.py nodes.
  - Include the code snippets within this document.
  - Include a screenshot of the terminal window showing both the publisher and subscriber nodes running with the expected output.

Expected Terminal Output (Publisher Node):

[INFO] [publisher_node-1] Robot is running
[INFO] [publisher_node-1] Robot is running
[INFO] [publisher_node-1] Robot is running
... (continues to publish)

Expected Terminal Output (Subscriber Node):

[INFO] [listener_node-2] I heard Robot is running
[INFO] [listener_node-2] I heard Robot is running
[INFO] [listener_node-2] I heard Robot is running
... (continues to receive messages)

2. Robot State Publishing

- **Scenario:** You are tasked with creating a ROS node to simulate the position of a robot in a 2D environment.
- **Objective:**
  - Create a ROS node that publishes the current simulated position (x, y, theta) of a robot to the topic "/robot_pose".
  - Use the geometry_msgs/PoseStamped message type to represent the robot's pose.
- **Task:**
  - Write the complete Python code for the robot state publisher node.
  - Include the code snippet within this document.
  - **(Optional)** If you have access to a simulator like Gazebo or RViz, run the node and visualize the robot's position. Include a screenshot of the visualization.

3. TurtleBot3 Teleoperation

- **Scenario:** You are working with a simulated TurtleBot3 in Gazebo.
- **Objective:**
  - Launch the TurtleBot3 simulation in Gazebo using the turtlebot3_gazebo launch file.
  - Launch the teleop_twist_keyboard node.
  - Teleoperate the TurtleBot3 using the keyboard commands.
- **Task:**
  - Include a screenshot of the Gazebo simulation showing the TurtleBot3.
  - Include a screenshot of the teleop_twist_keyboard terminal window.
  - Briefly describe the keyboard commands used and the observed robot behavior.

4. Gazebo Robot Collision Avoidance

- **Scenario:** You have two robots (e.g., two TurtleBots) already running in a Gazebo simulation.
- **Objective:**
  - Describe a simple approach to enable the robots to avoid colliding with each other.
  - Consider how you would use robot position information (e.g., from /gazebo/model_states) to implement this avoidance behavior.
- **Task:**
  - Explain your approach in detail.
  - **(Optional)** If possible, provide a conceptual outline of how you would implement this in a ROS node (e.g., subscribing to robot positions, calculating distances, adjusting robot velocities).

5. Dynamic Node Handling

- **Scenario:** You are working on a ROS system where the number of active nodes may change dynamically.
- **Objective:**
  - Describe a mechanism to dynamically discover and interact with new nodes that appear on the ROS network.
  - Consider how you would use this mechanism to, for example, automatically subscribe to topics published by newly discovered nodes.
- **Task:**
  - Explain your approach in detail.
  - Briefly discuss any relevant ROS tools or APIs that could be used to implement this mechanism (e.g., master API, parameter server).

Submission Guidelines:

1. **Answer all questions within this document.**
2. **Include code snippets, terminal output, and screenshots as instructed.**
3. **Submit this completed document through the provided Google Form.**
4. **Time Limit:** This assessment has a strict time limit of **24 hours**.

Note:

- This assessment is designed to evaluate your understanding of ROS fundamentals, basic robotics concepts, and your ability to work with simulated environments.
- Focus on providing clear, concise, and well-explained answers.