High-level Strategy

Preliminary Thoughts:

- Robots all share the same program, thus without sufficient randomization (i.e. flip's nondeterminism) all the robots will act in the exact same way.
- Strategy can be divided into three-parts: food collection, offense, defense.
- Using (leaving and sensing transponders for the same team, perhaps sensing opposition robot transponders?)

1. Efficient Food Collection

Goal: Maximize the amount of food collected and returned to your base.

Implementation:

- **Exploration**: Deploy robots to scout the terrain and identify food sources quickly. Use a randomized movement strategy to cover more area and avoid predictable patterns that could be exploited by enemy robots.
- **Food Detection and Collection**:
    - Use the `Sense` instruction to detect food directly ahead. If food is sensed, move towards it and use `PickUp`.
- **Return Path Optimization**: Once food is picked up:
    - Use `Sense` directed at the home base to find the shortest safe route back.
    - Consider marking safe and quick return paths using `Mark`, which later robots can follow back with the food more safely.
    - ***To minimize recharge time (idle time) movements should be minimal?*

**Looping states*

Considerations:

1. Efficient Exploration

- ~~Divide and Conquer: Split the map into sectors and assign robots to specific sectors to avoid overlap and ensure complete coverage.~~
- Boundary Recognition: Utilize the 'Sense' instruction to recognize and avoid obstacles and map boundaries.
- ~~Path Optimization: Implement algorithms like Breadth-First Search (BFS) for robots to find the shortest path to food. (*Should robots find the robots first?)~~

2. Food Collection and Delivery

- Priority to Food: When food is detected ('Sense Food'), robots should prioritize reaching and picking it up over other actions.
- Direct Delivery: Once food is collected, robots should take the most direct route back to the base, using pathfinding algorithms.
- Drop Off Coordination: Ensure that when a robot reaches the base to drop off food, it does so efficiently and quickly returns to collecting more food. → Implemented with TurnAround

3. Communication and Teamwork

- Transponder Usage: Use transponders to mark paths that have already been explored or to indicate food locations to other robots.
- Avoiding Traffic: Implement rules for robots to avoid or manage encounters with other robots, reducing idle time and blockages.

4. Conflict Avoidance and Management

- Enemy Detection: Use the sensing capability to detect nearby enemy robots and avoid them.
- ~~Safe Routing: Route paths to minimize encounters with enemy robots, especially when carrying food.~~
- Group Tactics: In scenarios where confrontation is unavoidable, use numbers to your advantage, surrounding enemy robots when you have superior numbers.

5. Energy and Recharging Management

- Energy Efficient Routing: Choose routes that minimize movement, as moving costs more energy.
- Recharge Planning: Plan routes that allow robots to recharge efficiently, possibly near the base or safe zones.

2. Defensive and Offensive Tactics

Goal: Protect your robots and their cargo from enemy attacks, and disrupt enemy operations when advantageous.

Implementation:

- **Situational Awareness:** Constantly use `Sense` to check for nearby enemy robots. ~~If an enemy is detected near a robot carrying food, divert other robots to help escort the carrier home.~~
- **Ambush and Defense:**

    - If multiple robots from our team are close together *how would we know, use them to set up ambushes near high-value areas like dense food locations or known paths enemies use.
    - Implement conditional behaviors based on enemy proximity: if an enemy is detected and your robot is not carrying food, attempt to surround and disable the enemy.
- **Retreat and Save:** If a robot is carrying food and encounters an enemy, use `Turn` and `Move` commands to retreat towards the base or towards nearby allies.

3. Communication and Coordination

Goal: Use transponders to create an efficient communication network among your robots.

Implementation:

- **Path Marking:** Use `Mark` and `Unmark` strategically to indicate safe paths or areas with high food concentration. This can guide other robots in your fleet to these areas or warn them of dangerous zones.
- **Coordination for Defense and Attack:** Develop a simple signaling system with `Mark` to request help or indicate an emergency. This could coordinate movements for defense (e.g., when carrying food) or for grouping to attack.

4. Adaptability and Learning

Goal: Adapt strategies based on outcomes of encounters and environmental variables.

Implementation:

- **Feedback Loops:** If possible, implement a basic learning mechanism or adjust strategies based on past successes and failures. This might involve choosing different paths if certain strategies lead consistently to ambush or high enemy encounters.
- **Environment Sensitivity:** Adjust robot behavior based on the density of obstacles, food availability, and observed enemy patterns. For example, if certain areas are too risky due to frequent enemy attacks, re-route robots to safer, though possibly longer, paths.

How many things to sense: 10 *Actually its 9 + 12 for all types of transponders so **move** can be more efficient in some cases
Recharge after incorrect move: 14
~~So wouldn't it be better to sense everything first before making a "Move", which is a very expensive instruction?~~

Transponders: use sparingly as setting it (and then not clearing it), and creating logic around transponders can make the robots move according to false information.

Building up our program:

1. Get the robots to move as sparsely as possible (based on the notion that in an unknown world, the best initial move is to spread out as far as possible):

   Move (if you can't move, then flip left or right turn) otherwise keep moving
   Problem:
   - Need the robots to come back to home base →
   - Robots crawl along the wall → Turn back option? (Turn L/R 3 times)
     **Notice it is always better to sense a wall rather than attempting to move then fail as it will take up 14 turns

2. Sense and Pickup Food → **Enhanced**: Check LeftAhead and RightAhead

   - What if there is no food? (sensed it first but another robot has taken it)
   - Or food is being held by the robot
     *Decide on a state for the robots in such cases
     In the 1st case, it's best to go back to sense ahead and in the 2nd, it's best to turn around and return to the homebase
     ** No way of differentiating so flip for it
3. Return home: Suppose we have all the robots use Transponder 0 as path to home →
4. Dropping food at the homebase, then **turn around → move forward for some rounds → flip direction**

Once food is found, we can leave a rough trail to the food using

Our O/D priority: Instead of focusing on offense, where we surround enemy robots to kill them, focus on defensive maneuvers and food collection.

Offensive Maneuvers:

- Robot that finds enemy: Sense for enemies nearby → Found: mark current cell with 3 → Move to either the left or right side of the enemy
- Robot that finds current cell marked with transponder 3: Sense for enemy in ahead → if enemy ahead: stay and keep sensing enemy ahead (loop) → Otherwise, sense leftahead and rightahead → if enemy ahead go back to loop → If not enemy found: flip a turn and keep moving
- Sense FoeWithFood → Attempt to surround and destroy it?

Defensive Maneuvers:

1. Immediate Detection: Constantly sense the immediate area for enemy presence.
   Sense Ahead Foe → Turn Around → Attempt to Move
   No Foes Sensed → Keep Moving Forward
2. Escape Maneuvers: If an enemy is detected ahead, execute a series of turns and movements designed to disorient or evade the enemy.

3. Repositioning: Find a safer location or rejoin friendly units to strengthen defensive positioning.
4. *Use of Environment: Leverage obstacles and terrain to prevent enemies from surrounding.
   a. *Building walls are strategic safe points as if we have a robot up against a wall, the minimum required **5 enemy robots** cannot surround it to destroy it.

#Update on movement: instead of random sparse movements for robots, we tried to keep the movement deterministic (i.e. a predictive movements to the food will allow us to keep the movement back to base predictive)

Reflection

Unexplored strategies in no particular order:

Swarm Intelligence

- Strategy: Implement algorithms based on swarm intelligence principles, mimicking the behaviors of biological systems like ant colonies or bee swarms. This could involve complex, coordinated actions such as formation flying, area sweeping, and collective problem-solving.
- Application: Use this for area control, efficient resource collection, and dynamic adaptation to environmental changes or enemy tactics.

Machine Learning Integration

- Strategy: Train machine learning models based on reinforcement learning to adapt robot behavior based on the outcome of their actions. Robots could learn the most effective strategies for exploration, combat, and resource management.
- Application: Enable robots to predict enemy movements, optimize paths for resource collection, and dynamically adjust tactics during encounters based on past successes or failures.

Decentralized Command

- Strategy: Rather than using a central control structure, employ a decentralized approach where each robot operates independently but follows a shared set of objectives and rules. This can enhance resilience against disruptions that would disable a central controller.
- Application: Particularly useful in hostile environments where communication might be compromised or where robots need to operate over large, dispersed areas.

Dynamic Role Assignment

- Strategy: Allow robots to change roles dynamically based on the current situation. A scout could become a harvester upon finding a resource, or a defender when an enemy is spotted.
- Application: This flexibility would allow a more fluid response to changing conditions, optimizing resource use and defensive capabilities without the need for reprogramming or manual intervention.

Advanced Obstacle Navigation

- Strategy: Develop more sophisticated algorithms for obstacle avoidance that also take into account strategic considerations, such as using obstacles for cover or to funnel enemies into traps.
- Application: Enhance survival rates in complex terrains and create opportunities to ambush or escape from enemies.

Energy Management Tactics

- Strategy: Implement energy management tactics that optimize the robots' battery or fuel usage based on their tasks, travel distance, and the necessity of high-power actions like combat or rapid movement.
- Application: Increase operational longevity and efficiency, especially important in scenarios with limited recharging opportunities or prolonged missions.

Psychological Warfare

- Strategy: Use actions that could confuse or demoralize enemy robots, such as mimicking enemy signals, creating false resource trails, or simulating larger forces.
- Application: Useful in competitive scenarios to gain a psychological edge, disrupting enemy strategies and cohesion.