How to run code & verify solution:

- Install Python 3.9
- Pull the main.py file, two folders (problems, solutions).
- Delete all files from the solutions folder (for verification purposes).
- Run the main.py file.
- Check the solution of problems in the solutions folder (stored according to the problem number).

Approach towards Solving Problems:

- First task to solve the problem was to load the data from a problem file into a representation that is easy to work with.
  Approach:
  1. Load the problem files from the directory.
  2. To store solution to files develop a solution path for each problem.
  3. Convert the data to a list.
  4. Based on the first element of data differentiate between two problems namely, Find Plan or Check the Plan.
  5. Pass the problem data and solution path to the functions.
- After the data is loaded in the required representation, the two separate parts of the problem solved are explained below:

a) Check the Plan

- Implement function:

```
check_plan(problem_list, final_solution_path)
```

- To check plan there are three different types of problems:
  1. Orientation and starting position is known.
  2. Orientation unknown and starting position known.
  3. Orientation and starting position both unknown.

- Implement get_starting_position(final_map) function to fetch the initial orientation and starting position(coordinates)

```
orientation, x, y = get_starting_position(final_map)
```

- Implement function try_plan() to overall iterate through the plan and check if it is good plan or bad plan:

```
try_plan(orientation, x, y, plan, final_map)
```

- Implement function find_dirty_spaces(final_map) to fetch the dirty spaces (i.e ' ') from the problem map.

```
find_dirty_spaces(final_map)
```

- For the case of known Orientation and starting position:
  1. Implement three functions for instructions R, L and M respectively:

```
move_to_r(orientation, x, y, plan, final_map)
move_to_l(orientation, x, y, plan, final_map)
move_to_m(orientation, x, y, plan, final_map,
dirty_spaces)
```

- For the case of unknown Orientation and starting position is known:
    1. Implement function:

```
move_for_unknown_orientation(orientation, x, y, plan, final_map)
```

    2. This function tries all four initial orientations:

```
["^", "<", ">", "v"]
```

    3. After trying the possible initial orientations it replies with the overall list of not cleaned spaces by the plan.
- For the case of unknown Orientation and starting position:
    1. Implement function:

```
unknown_orientation_and_position(plan, final_map, dirty_spaces):
```

    2. This function tries setting one by one each dirty space as starting position and sets orientation to S so to try in all four orientations.
    3. After trying combinations returns the final list of unvisited dirty spaces.

b) Find Plan

- Implement function:

```
find_plan(problem_list, final_solution_path):
```

- To Find plan there are three different types of problems:
    1. Orientation and starting position is known.
    2. Orientation unknown and starting position known.
    3. Orientation and starting position both unknown.
- Implement function get_neighbours to fetch the neighbours of the position coordinate.

```
get_neighbours(x, y, final_map)
```

- Implement function next_move() which provides with the functionality to return the new position and instruction to move to new position as a sub plan on each iteration when function is called.

```
next_move(orientation, x, y, i[0], i[1], final_map)
```

-
- Also uses get_starting_position(final_map) function to fetch the initial orientation and starting position(coordinates).
- Implement function create_plan() to overall iterate through the different three problem types:

```
create_plan(orientation, x, y, plan, final_map):
```

- For the case of known Orientation and starting position:
    1. Call function

```
clean_spaces(dirty_spaces, x, y, orientation, final_map, origin, cleaned_spaces):
```

    2. In this function there are two cases:
        a. There is a position in the neighbours of the current position which is dirty. The robot moves to that position.

b. If there is no dirty position in neighbourhood, then it iterates through the already cleaned_spaces list and starts traversing and checking if any new neighbor in those places is left dirty.

- For the case of unknown Orientation and known starting position:
  1. Set default orientation for this problem to:

     ```
     orientation = "^"
     ```

     Note Possible Improvements: (Can try all possible orientations and compare plans) to finalize. Did not try all the possible orientations.
  2. Call the clean_spaces() function. Rest of functionality is same.
- For the case of unknown Orientation and starting position:

  1. Set the starting position for this problem to the first dirty space in the list(default).
     Note Possible Improvements: (Can try all possible starting positions and compare plans) to finalize.
  2. Set default orientation for this problem to:

     ```
     orientation = "^"
     ```

     Note Possible Improvements: (Can try all possible orientations and compare plans) to finalize. Did not try all the possible orientations.
  3. Call the clean_spaces() function. Rest of functionality is same.
- Implement write_to_file() to store the solution to the respective solution file in directory:

```
write_to_file(plan_type, dirty_spaces_not_cleaned, plan,
final_solution_path)
```