PROGRAM-6

VACUUM CLEANER PROBLEM

AIM:-

To write and execute the python program for the vacuum cleaner program.

PROCEDURE:-

- **Class Definition**:
  - Define the VacuumCleaner class, which represents the autonomous vacuum cleaner.
- **Helper Functions**:
  - Define helper functions:
    - is_valid_move(x, y): Checks if a move to coordinates (x, y) is valid within the grid boundaries.
    - is_dirty(x, y): Checks if the cell at coordinates (x, y) is dirty.
    - clean_cell(x, y): Cleans the cell at coordinates (x, y) if it's valid.
- **Depth-First Search (DFS)**:
  - Define the DFS algorithm to explore and clean the grid recursively.
  - Start DFS from the initial cell (0, 0) and mark visited cells..
- **Cleaning Function**:
  - Define the clean_grid method to initiate the cleaning process by calling the DFS algorithm.
- **Print Moves**:
  - Define the print_moves method to print the sequence of moves made by the vacuum cleaner to clean all dirty cells.
- **Example Usage**:
  - Define the grid representing the environment with clean ('C') and dirty ('D') cells.
  - Call the clean_grid method to clean the grid.
  - Print the sequence of moves made by the vacuum cleaner.

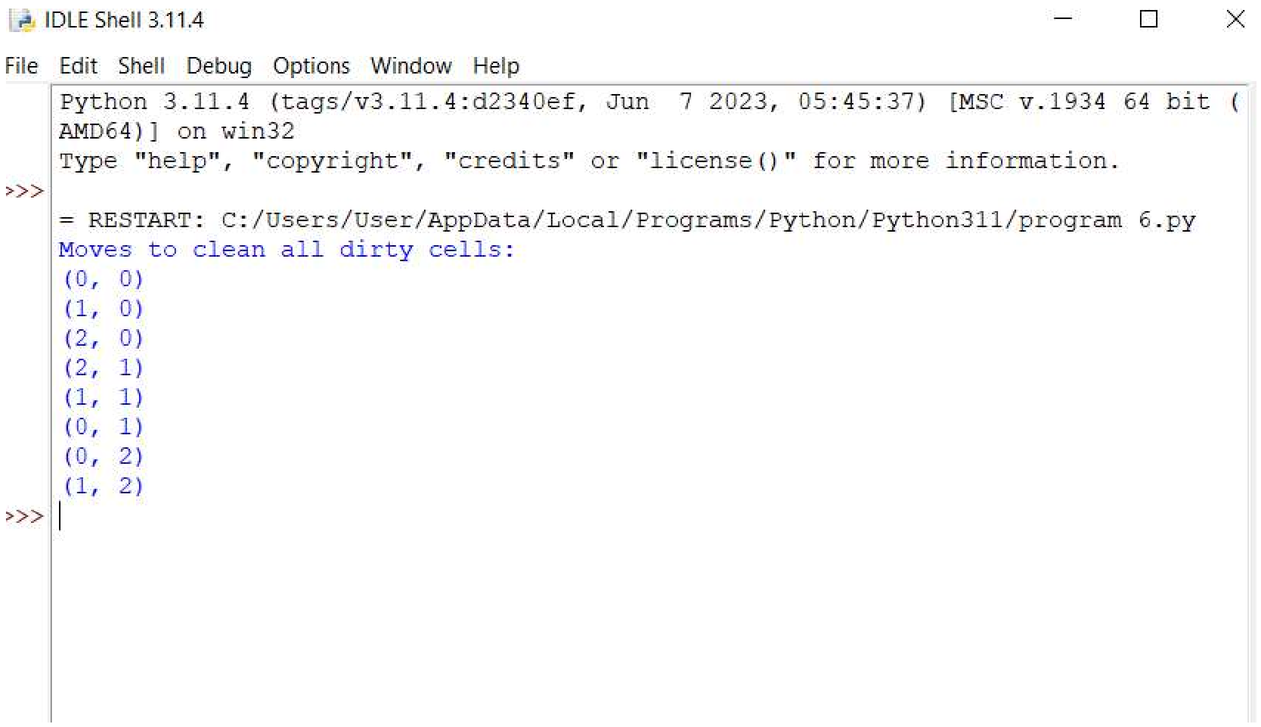
CODING:-

```
class VacuumCleaner:

    def __init__(self, grid):
        self.grid = grid

        self.rows = len(grid)

        self.cols = len(grid[0])

        self.visited = set()

        self.moves = []

    def is_valid_move(self, x, y):

        return 0 <= x < self.rows and 0 <= y < self.cols

    def is_dirty(self, x, y):

        return self.is_valid_move(x, y) and self.grid[x][y] == 'D'

    def clean_cell(self, x, y):

        if self.is_valid_move(x, y):

            self.grid[x][y] = 'C'

    def dfs(self, x, y):

        if not self.is_valid_move(x, y) or (x, y) in self.visited:

            return False

        self.visited.add((x, y))

        self.clean_cell(x, y)

        self.moves.append((x, y))

        if all(cell == 'C' for row in self.grid for cell in row):

            return True

        # Try moving in all four directions
        if self.dfs(x + 1, y) or self.dfs(x - 1, y) or self.dfs(x, y + 1) or self.dfs(x, y - 1):
            return True
        # If no solution found, backtrack
        self.clean_cell(x, y)
        self.moves.pop()
        return False
    def clean_grid(self):
        start_x, start_y = 0, 0
        self.dfs(start_x, start_y)
    def print_moves(self):
        print("Moves to clean all dirty cells:")
        for move in self.moves:
            print(move)
# Example usage:
grid = [
    ['C', 'D', 'C', 'C'],
    ['C', 'C', 'D', 'C'],
    ['C', 'C', 'C', 'C']
]

vacuum_cleaner = VacuumCleaner(grid)
```

vacuum_cleaner.clean_grid()

vacuum_cleaner.print_moves()

OUTPUT:-



```
Python 3.11.4 (tags/v3.11.4:d2340ef, Jun  7 2023, 05:45:37) [MSC v.1934 64 bit (
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Users/User/AppData/Local/Programs/Python/Python311/program 6.py
Moves to clean all dirty cells:
(0, 0)
(1, 0)
(2, 0)
(2, 1)
(1, 1)
(0, 1)
(0, 2)
(1, 2)
>>>
```

RESULT:-

Hence the program has been successfully executed and verified.