Project: Maze Safety Classification and Path Planning using Perceptron and A* Algorithm

Objective: In this project, you will create a maze where the goal is to find the shortest safe path from a start point to an end point using the A* algorithm. You will incorporate a Perceptron to classify tiles as "safe" or "unsafe" based on their type and other factors. Your task is to implement pathfinding while ensuring that the algorithm only navigates through safe tiles (grass or safe terrain), and avoids unsafe tiles.

Project Tasks:

1. Maze Generation:
    - Generate a maze based on user-defined size
    - Each tile in the maze will have the following properties:
        - Tile Type: Either grass, water, or obstacle.
        - Elevation: A numerical value (0-10) that represents the height of the tile.
    - Define a start and end position within the maze.
2. Perceptron for Safety Classification:
    - You will implement a Perceptron to classify each tile as safe or unsafe. The classifier should use the following features:
        - Tile Type: Grass (0), Water (1).
        - Elevation: A numerical value (0-10).
        - Manhattan Distance to Nearest Obstacle: The distance from the tile to the closest obstacle, calculated using the Manhattan distance formula.
    - The output label from the Perceptron will be:
        - 1 for safe
        - 0 for unsafe
    - Training data will be uploaded for the training process. (See Data.xlsx)
3. A* Algorithm Implementation:
    - Use the A* algorithm to find the shortest path from the start to the end point.
    - During the A* search, each time a tile is expanded (or a neighbour is considered):
        - Check if the neighbour tile is safe: Pass the tile's features (type, elevation, distance to the nearest obstacle) to the Perceptron. Only safe tiles (classified by the Perceptron as 1) should be add to the list.
        - Skip unsafe tiles (classified as 0) and do not add them to the list.
    - Calculate the Manhattan distance as the heuristic for the A* algorithm to determine the cost of moving between tiles.

4. Pathfinding Process:
    - The algorithm should find the shortest path from the start to the end, while adhering to the safety constraints provided by the Perceptron.
    - If no valid path exists (i.e., the safe path is blocked), the algorithm should return an appropriate message (e.g., "No safe path found").

Requirements:

1. Grid Representation:
    - Represent the maze as a grid (e.g., a 2D array) where each cell contains information about the tile (type, elevation, distance to nearest obstacle).
    - Define a start and end point in the grid.
2. Perceptron:
    - Implement a Perceptron that can take the tile features (type, elevation, and distance to the nearest obstacle) and predict whether the tile is safe or unsafe.
3. A* Algorithm:
    - Implement the A* algorithm that calculates the shortest path from the start to the end point.
    - Modify the A* search to only consider safe tiles based on the Perceptron output.
4. Manhattan Distance:
    - Implement the Manhattan distance formula to calculate the distance between tiles. Use this as the heuristic in the A* algorithm.
5. Final Output:
    - Display the maze along with the path found by the A* algorithm, marking the start, end, and path tiles.