**TASK 1**

**Grid Robotics Pathfinding with A\* Algorithm**

## Viveka G - 21PW40

**Overview**

This project is a simple grid-based pathfinding simulation using the A\* algorithm, implemented in Python with Pygame for visualization. The grid contains randomly placed obstacles, and the user can set start and end points to visualize the shortest path found by the A\* algorithm.

**Requirements**

* Python 3.x
* Pygame library

## Overall Working

This project implements a grid-based pathfinding simulation using the A\* algorithm, visualized with Pygame. Upon running the script, a window displays a grid with randomly placed obstacles. The user can click on empty cells to set a start point (highlighted in blue) and an end point (highlighted in red). Once both points are set, the A\* algorithm calculates the shortest path between them, avoiding obstacles. The path, if found, is displayed in green. The grid is continuously updated to reflect user interactions, showing the grid structure, obstacles, start and end points, and the computed path in real-time. The program handles user input to dynamically reset and recalculate paths, providing an interactive demonstration of pathfinding algorithms.

**TASK 2**

**Robot Navigation Simulation with Pygame**

**Overview**

This project simulates a robot navigating towards the center of a screen with a large pillar in the middle, implemented in Python with Pygame for visualization. The robot starts from a random position outside the pillar and moves towards the center while avoiding the obstacle. The user can restart the simulation by clicking a button once the robot reaches the center.

**Requirements**

* Python 3.x
* Pygame library

## Overall Working

This project is a simulation of a robot navigating toward the center of a screen while avoiding a large rectangular pillar in the middle, visualized using Pygame. The simulation initializes with the robot starting from a random position outside the pillar. The robot's objective is to reach the center of the screen, which is obstructed by the pillar. The movement of the robot is controlled by a method that calculates the direction towards the center and moves the robot incrementally, checking for potential collisions with the pillar. If a collision is detected, the robot adjusts its path to move along an unobstructed axis. The user can restart the simulation by clicking a "Restart" button that appears once the robot successfully reaches the center. The main loop continuously handles events such as quitting the application or restarting the simulation, updates the screen by redrawing the robot and the pillar, and manages the robot's movement towards the center. The simulation provides a dynamic and interactive demonstration of obstacle avoidance and pathfinding in a confined space.