Planner Algorithm Flow:

- The world is represented using matplotlib and is generated through the ‘World’ class.
- Expand(): Uses kinematic constraints to find the neighbours of the current node and returns them as a list of coordinates. This function is the primary point of difference in the 3 codes
- Collision_check() : Checks collision between the robot and obstacles by treating them as polygons. Nodes with active collisions are considered invalid and are not searched.
- Cost_function and calc_heuristic(): These calculate the cost and heuristic respectively for the current node
- A_star() : Applies the hybrid A* algorithm

Pseudocode for Hybrid A:*

```
1. Initialize the start and goal positions.
2. Generate a graph of motion primitives.
3. Initialize the closed and open lists.
4. Insert the start position into the open list with f-cost = heuristic(start).
5. while the open list is not empty do
    a. Pop the node with the lowest f-cost from the open list.
    b. Generate the successors of the current node.
    c. For each successor node, do the following:
        i. Compute the cost to move from the current node to the successor node.
        ii. If the successor is the goal position, compute the continuous path using numerical integration and optimization techniques.
        iii. If the successor is not in the closed list or has a lower cost than its previous cost, add it to the open list with f-cost = g-cost + heuristic(successor).
    d. Add the current node to the closed list.
6. If no path is found, return failure.
7. Compute the continuous path using the final node from the closed list.
8. Apply the path to the vehicle by generating a sequence of control inputs (e.g., steering angles, velocities) that follow the trajectory.
```