EXPERIMENT 9:

Write the python program to implement A* algorithm

AIM:

The aim is to implement A* algorithm using python program.

PROGRAM:

```
import heapq

class Node:
    def __init__(self, row, col, cost, heuristic):
        self.row = row
        self.col = col
        self.cost = cost
        self.heuristic = heuristic
        self.parent = None

    def __lt__(self, other):
        return (self.cost + self.heuristic) < (other.cost + other.heuristic)

def astar(grid, start, goal):
    rows, cols = len(grid), len(grid[0])
    open_set = [Node(start[0], start[1], 0, heuristic(start, goal))]
    closed_set = set()

    while open_set:
        current_node = heapq.heappop(open_set)

        if (current_node.row, current_node.col) == goal:
            path = []
            while current_node:
                path.append((current_node.row, current_node.col))
                current_node = current_node.parent
```

```
            return path[::-1]
        closed_set.add((current_node.row, current_node.col))
        for neighbor in get_neighbors(current_node.row, current_node.col, rows, cols):
            if neighbor not in closed_set:
                neighbor_row, neighbor_col = neighbor
                cost = current_node.cost + grid[neighbor_row][neighbor_col]
                heuristic_val = heuristic((neighbor_row, neighbor_col), goal)
                new_node = Node(neighbor_row, neighbor_col, cost, heuristic_val)
                new_node.parent = current_node
                heapq.heappush(open_set, new_node)
    return None
def heuristic(a, b):
    return abs(a[0] - b[0]) + abs(a[1] - b[1])
def get_neighbors(row, col, max_row, max_col):
    neighbors = []
    if row > 0:
        neighbors.append((row - 1, col))
    if row < max_row - 1:
        neighbors.append((row + 1, col))
    if col > 0:
        neighbors.append((row, col - 1))
    if col < max_col - 1:
        neighbors.append((row, col + 1))
    return neighbors
grid = [
    [1, 3, 1, 2, 4],
    [2, 5, 3, 1, 2],
    [1, 1, 2, 5, 3],
    [3, 2, 4, 1, 4],
    [2, 4, 2, 3, 3]
]
```

```
start_node = (0, 0)
goal_node = (4, 4)
result = astar(grid, start_node, goal_node)
if result:
    print("Shortest path:", result)
else:
    print("No path found.")
```

RESULT:

A*algorithm program executed successfully.