10.Write the python program to implement A algorithm

import heapq

def heuristic(a, b):

return abs(a[0] - b[0]) + abs(a[1] - b[1])

def a\_star(grid, start, goal):

rows, cols = len(grid), len(grid[0])

open\_set = []

heapq.heappush(open\_set, (0 + heuristic(start, goal), 0, start))

came\_from = {}

g\_score = {start: 0}

while open\_set:

\_, current\_cost, current = heapq.heappop(open\_set)

if current == goal:

# Reconstruct path

path = []

while current in came\_from:

path.append(current)

current = came\_from[current]

path.append(start)

path.reverse()

return path

for dx, dy in [(-1,0), (1,0), (0,-1), (0,1)]:

neighbor = (current[0]+dx, current[1]+dy)

x, y = neighbor

if 0 <= x < rows and 0 <= y < cols and grid[x][y] == 0:

tentative\_g\_score = g\_score[current] + 1

if neighbor not in g\_score or tentative\_g\_score < g\_score[neighbor]:

came\_from[neighbor] = current

g\_score[neighbor] = tentative\_g\_score

f\_score = tentative\_g\_score + heuristic(neighbor, goal)

heapq.heappush(open\_set, (f\_score, tentative\_g\_score, neighbor))

return None # No path found

# Example grid (0 = free, 1 = obstacle)

grid = [

[0, 1, 0, 0, 0],

[0, 1, 0, 1, 0],

[0, 0, 0, 1, 0],

[1, 1, 0, 0, 0]

]

start = (0, 0)

goal = (3, 4)

path = a\_star(grid, start, goal)

if path:

print("Path found:", path)

else:

print("No path found.")\

OUTPUT:

