CSE 422
Assignment 01

Name: Tahmid Iqbal
ID: 21201701
Sec: 06

```python
import heapq

def inpFunc(file_path_01):
    with open(file_path_01, 'r') as file:
        graph = {}
        h = {}

        for line in file:
            parts = line.split()
            city = parts[0]
            heuristic = int(parts[1])
            neighbors = parts[2:]

            h[city] = heuristic
            graph[city] = {}

            for i in range(0, len(neighbors), 2):
                neighbor = neighbors[i]
                distance = int(neighbors[i+1])
                graph[city][neighbor] = distance

    return graph, h

# The a_star_search function finds the shortest path from a start node to
a goal node using a priority queue

def a_star_search(graph, h, start, goal):
    priority_Q = []   # Priority queue to store nodes to be explored
    heapq.heappush(priority_Q, (0 + h[start], 0, start, [start]))

    visited_nodes = set()  # tracking da visited nodes

    while priority_Q:
        f_score, g_score, current_node, path = heapq.heappop(priority_Q)
#lowest f_score node

        if current_node in visited_nodes:
            continue

        if current_node == goal:
```

```
            return path, g_score

        visited_nodes.add(current_node)  # Marking visited

        for neighbor, distance in graph[current_node].items():

            if neighbor in visited_nodes:
                continue

            tentative_g_score = g_score + distance
            f_score = tentative_g_score + h[neighbor]

            heapq.heappush(priority_Q, (f_score, tentative_g_score,
neighbor, path + [neighbor]))

    return None, float('inf')

def main():
    file_path_01 = 'input.txt'
    file_path_02 = 'output.txt'
    graph, h = inpFunc(file_path_01)

    start = input("Start node: ")
    goal = input("Destination node: ")

    path, total_dist = a_star_search(graph, h, start, goal)

    with open(file_path_02, 'w') as output_file:
        if path:
            output_file.write(f"Path: {' -> '.join(path)}\n")
            output_file.write(f"Total distance: {total_dist} km\n")


        else:
            output_file.write("NO PATH FOUND\n")

if __name__ == "__main__":
    main()
```