```
import heapq

def astar(graph, start, goal, heuristic):
    visited = set()
    priority_queue = [(heuristic[start], 0, start)]  # 
Priority queue sorted by f-value (heuristic + cost)
    path_cost = {start: 0}
    path = {start: None}

    while priority_queue:
        print("OPEN LIST:", priority_queue)
        _, current_cost, current_node = heapq.heappop(
priority_queue)

        if current_node == goal:
            return construct_path(path, start, goal)

        visited.add(current_node)
        print("CLOSED LIST:",visited)
        for n, edge_cost in graph[current_node].items
():
            total_cost = path_cost[current_node] + 
edge_cost
            if n not in visited or total_cost < 
path_cost[n]:
                path_cost[n] = total_cost
                heapq.heappush(priority_queue, (
total_cost + heuristic[n], total_cost, n))
                path[n] = current_node

    return None


def construct_path(path, start, goal):
    current_node = goal
    path_sequence = []
    while current_node:
        path_sequence.insert(0, current_node)
        current_node = path[current_node]
    return path_sequence
```

```
# Example usage Graph
# graph = {
#     'A': {'B': 1, 'C': 2},
#     'B': {'D': 3, 'E': 4},
#     'C': {'F': 1},
#     'D': {},
#     'E': {},
#     'F': {}
# }

# graph = {
#     'A': {'B': 1, 'C': 2},
#     'B': {'D': 3, 'E': 4},
#     'C': {'F': 1},
#     'D': {},
#     'E': {},
#     'F': {}
# }

graph = {
    'A': {'B': 2, 'E': 3},
    'B': {'C': 1, 'G': 9},
    'C': {},
    'D': {'G':1},
    'E': {'D':6},
    'G': {}
}

start_node = input("Enter the start node: ").upper()
goal_node = input("Enter the goal node: ").upper()

# heuristic = {
#     'A': 3,
#     'B': 2,
#     'C': 4,
#     'D': 1,
#     'E': 1,
#     'F': 0
# }
```

```
heuristic = {
    'A': 11,
    'B': 6,
    'C': 99,
    'D': 1,
    'E': 7,
    'G': 0
}

astar_path = astar(graph, start_node, goal_node,
heuristic)
print()
if astar_path:
    print('Path:', astar_path)
    print(f"Goal '{goal_node}' found using A*.")
else:
    print(f"Goal '{goal_node}' not found using A*.")
```