Name: **Insha Nadeem**

Roll Number: **220985**

Program: **BSAI-IV-A**

Submitted To: **Sir Tahir Akram**

Date: **29 February 2024**

Course: **Artificial Intelligence**

Uniform Cost Search

Code:

```
from queue import PriorityQueue

class Node:
    def _init_(self, state, cost, path):
        self.state = state
        self.cost = cost
        self.path = path

    def _lt_(self, other):
        return self.cost < other.cost

def uniform_cost_search(graph, start, goal):
    pq = PriorityQueue()
    visited = set()

    start_node = Node(start, 0, [start])
    pq.put(start_node)

    while not pq.empty():
        current_node = pq.get()
        current_state, current_cost, current_path = current_node.state,
current_node.cost, current_node.path

        if current_state == goal:
            return current_cost, current_path

        visited.add(current_state)

        for neighbor, edge_cost in graph.get(current_state, []):
            total_cost = current_cost + edge_cost

            if neighbor not in visited:
                new_path = current_path + [neighbor]
                pq.put(Node(neighbor, total_cost, new_path))

    return None, None
```

```
graph = {'A': [['B', 5],['D', 3]],
        'B': [['C', 1]],
        'C': [['G', 8],['E',6]],
        'D': [['F', 2],['E',2]],
        'E' : [['B',4]] ,
        'F': [['G', 3]],
        'G' : [['E',4]]}

start_node = 'A'
goal_node = 'G'

cost, path = uniform_cost_search(graph, start_node, goal_node)

if cost is not None and path is not None:
    print("Cost to reach goal:", cost)
    print("Path:", path)
else:
    print("Goal state not reachable.")
```

Output: