```
# Program to create a weighted graph, display it, and calculate path cost
def create_graph():
    graph = {}
    n = int(input("Enter number of vertices: "))
    e = int(input("Enter number of edges: "))
    print("\nEnter edges with costs in the format: u v cost")
    for _ in range(e):
        u, v, cost = input().split()
        cost = float(cost)
        # Add to adjacency list (undirected)
        if u not in graph:
            graph[u] = {}
        if v not in graph:
            graph[v] = {}
        graph[u][v] = cost
        graph[v][u] = cost  # comment this line if graph is directed
    return graph
def display_graph(graph):
    print("\nGraph structure (Adjacency List with costs):")
    for node, neighbors in graph.items():
```

```
        for nbr, cost in neighbors.items():
            print(f"{node} --({cost})--> {nbr}")
    print()
def calculate_path_cost(graph, path):
    total_cost = 0
    for i in range(len(path) - 1):
        u, v = path[i], path[i + 1]
        if u in graph and v in graph[u]:
            total_cost += graph[u][v]
        else:
            print(f"Path between {u} and {v} does not exist!")
            return None
    return total_cost
# ---------- Main Program ----------
if __name__ == "__main__":
    graph = create_graph()
    display_graph(graph)
    path_input = input("Enter path (e.g. A B C): ").split()
    total_cost = calculate_path_cost(graph, path_input)
    if total_cost is not None:
        print(f"\nTotal cost of path {' -> '.join(path_input)} =
{total_cost}")
```