Assignment No:- 3 4

Code:-

```
import heapq


class Graph:
    def __init__(self, vertices):
        self.vertices = vertices
        self.graph = [[] for _ in range(vertices)]

    def add_edge(self, u, v, weight):
        self.graph[u].append((v, weight))
        self.graph[v].append((u, weight))

    def prim_mst(self):
        pq = []
        mst = []
        visited = set()

        start_vertex = int(input("Enter the starting vertex (0 to {}): ".format(self.vertices - 1)))
        if start_vertex < 0 or start_vertex >= self.vertices:
            print("Invalid starting vertex.")
            return []

        for neighbor, weight in self.graph[start_vertex]:
            heapq.heappush(pq, (weight, start_vertex, neighbor))

        visited.add(start_vertex)
```

```
        while pq:
            weight, u, v = heapq.heappop(pq)
            if v not in visited:
                visited.add(v)
                mst.append((u, v, weight))
                for neighbor, weight in self.graph[v]:
                    heapq.heappush(pq, (weight, v, neighbor))

        return mst

# Get the number of vertices from the user
num_vertices = int(input("Enter the number of vertices: "))

# Create a graph with the specified number of vertices
g = Graph(num_vertices)

# Get edges and weights from the user
num_edges = int(input("Enter the number of edges: "))
for _ in range(num_edges):
    u, v, weight = map(int, input("Enter edge (u v weight): ").split())
    g.add_edge(u, v, weight)

# Find and print the Minimum Spanning Tree
minimum_spanning_tree = g.prim_mst()

if minimum_spanning_tree:
```

```
    print("\nMinimum Spanning Tree:")
    for edge in minimum_spanning_tree:
        print(f"Edge: {edge[0]} - {edge[1]}, Weight: {edge[2]}")
else:
    print("Invalid input or graph disconnected.")
```

Output:-

```
PS F:\3.Programming\.vscode\AI Assignments> &
C:/Users/hp/AppData/Local/Microsoft/WindowsApps/python3.10.exe
"f:/3.Programming/.vscode/AI Assignments/Assignment No.3 Code.py"
Enter the number of vertices: 5
Enter the number of edges: 7
Enter edge (u v weight): 0 1 2
Enter edge (u v weight): 0 3 4
Enter edge (u v weight): 1 2 3
Enter edge (u v weight): 1 3 2
Enter edge (u v weight): 2 4 1
Enter edge (u v weight): 3 4 5
Enter edge (u v weight): 2 3 1
Enter the starting vertex (0 to 4): 0

Minimum Spanning Tree:
Edge: 0 - 1, Weight: 2
Edge: 1 - 3, Weight: 2
```

Edge: 3 - 2, Weight: 1

Edge: 2 - 4, Weight: 1