Pracical No. 3

Name : Mohan Kadambande
Roll No. : 13212
Aim : Implement Prim's Algorithm usng Greedy search Alogorihm.
Code :

```
import sys

# Function to find the vertex with the minimum key value
def min_key(key, mst_set, V):
    min_val = sys.maxsize
    min_index = -1

    for v in range(V):
        if key[v] < min_val and not mst_set[v]:
            min_val = key[v]
            min_index = v

    return min_index

# Function to implement Prim's algorithm
def prim_mst(graph, V):
    parent = [-1] * V
    key = [sys.maxsize] * V
    mst_set = [False] * V

    # Start from the first vertex
    key[0] = 0

    for _ in range(V):
        u = min_key(key, mst_set, V)

        # Add u to the MST
        mst_set[u] = True

        # Update the key values and parent index of the adjacent vertices
        for v in range(V):
            if graph[u][v] != 0 and not mst_set[v] and graph[u][v] < key[v]:
                key[v] = graph[u][v]
                parent[v] = u
    # Print the MST
    print("Edge \tWeight")
    for i in range(1, V):
        print(f"{parent[i]} - {i} \t{graph[i][parent[i]]}")

# Main function to take user input
def main():
    V = int(input("Enter the number of vertices: "))

    # Create an empty graph with zeros
    graph = [[0 for _ in range(V)] for _ in range(V)]

    print("Enter the adjacency matrix (enter 0 for no edge between vertices):")
    for i in range(V):
        for j in range(i + 1, V):
            weight = int(input(f"Enter the weight of edge ({i}, {j}): "))
            graph[i][j] = weight
            graph[j][i] = weight

    # Call Prim's algorithm
    prim_mst(graph, V)

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

Output :

```
Enter the number of vertices: 5
Enter the adjacency matrix (enter 0 for no edge between vertices):
Enter the weight of edge (0, 1): 2
Enter the weight of edge (0, 2): 3
Enter the weight of edge (0, 3): 0
Enter the weight of edge (0, 4): 0
Enter the weight of edge (1, 2): 5
Enter the weight of edge (1, 3): 6
Enter the weight of edge (1, 4): 0
Enter the weight of edge (2, 3): 7
Enter the weight of edge (2, 4): 0
Enter the weight of edge (3, 4): 0
Edge    Weight
0 - 1   2
0 - 2   3
1 - 3   6
-1 - 4  0
```