ass 3

```
# Implement Greedy Search Algorithm for any of the following application: Prim's Minimal
# Spanning Tree Algorithm

import sys

class Graph:
    def __init__(self, vertices):          #v
        self.vertices = vertices
        self.graph = [[0 for column in range(vertices)]
                      for row in range(vertices)]

    def printMST(self, parent):
        print("Edge \tWeight")#1
        for i in range(1, self.vertices):
            print(parent[i], "-", i, "\t", self.graph[i][parent[i]])



    def minKey(self, key, mstSet):
        min = sys.maxsize
        for v in range(self.vertices):
            if key[v] < min and mstSet[v] == False:
                min = key[v]
                min_index = v
        return min_index    #a

    def primMST(self):
        key = [sys.maxsize] * self.vertices
        parent = [None] * self.vertices
        key[0] = 0
```

```
        mstSet = [False] * self.vertices
        parent[0] = -1

        for cout in range(self.vertices):
            u = self.minKey(key, mstSet)
            mstSet[u] = True

            for v in range(self.vertices):  #c
                if self.graph[u][v] > 0 and mstSet[v] == False and key[v] > self.graph[u][v]:
                    key[v] = self.graph[u][v]
                    parent[v] = u

        self.printMST(parent)



#k
num_vertices = int(input("Enter the number of vertices in the graph: "))

graph = Graph(num_vertices)

for i in range(num_vertices):
    for j in range(num_vertices):
        if i != j and graph.graph[i][j] == 0:
            weight = int(input(f"Enter the weight of edge ({i}, {j}): "))
            graph.graph[i][j] = weight
            graph.graph[j][i] = weight

graph.primMST()
```

 # ass 3 done