Name : Prasad Borkar
Roll No.: COTA59
Experiment No.: 03(AI)

PROGRAM CODE :

```
# Selection Sort
def selection_sort(arr):
    n = len(arr)
    for i in range(n - 1):
        min_index = i
        for j in range(i + 1, n):
            if arr[j] < arr[min_index]:
                min_index = j
        arr[i], arr[min_index] = arr[min_index], arr[i]
    return arr

input_array = []
total = int(input("Enter the total no. of element : "))
for k in range (total):
    num = int(input("Enter the Number {}: ".format(k + 1)))
    input_array.append(num)
sorted_array = selection_sort(input_array)
print("Sorted Array:", sorted_array)
```

OUTPUT :

```
Enter the total no. of element : 5
Enter the Number 1: 34
Enter the Number 2: 21
Enter the Number 3: 14
Enter the Number 4: 98
Enter the Number 5: 28
Sorted Array: [14, 21, 28, 34, 98]
```

PROGRAM CODE :

```
# Prim's Minimal Spanning Tree Algorithm

import sys

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

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

    def minKey(self, key, mstSet):

        # Initialize min value
        min = sys.maxsize

        for v in range(self.V):
            if key[v] < min and mstSet[v] == False:
                min = key[v]
                min_index = v

        return min_index

    def primMST(self):


        key = [sys.maxsize] * self.V
        parent = [None] * self.V

        key[0] = 0
        mstSet = [False] * self.V

        parent[0] = -1

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

```
                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)


# Driver's code
if __name__ == '__main__':
    g = Graph(5)
    g.graph = [[0, 2, 0, 6, 0],
               [2, 0, 3, 8, 5],
               [0, 3, 0, 0, 7],
               [6, 8, 0, 0, 9],
               [0, 5, 7, 9, 0]]

    g.primMST()
```

OUTPUT :

```
Edge    Weight

0 - 1    2
1 - 2    3
0 - 3    6
1 - 4    5
```

PROGRAM CODE :

```
# Prim's Minimal Spanning Tree Algorithm

INF = 9999999
N = 5
G = [[0, 2, 0, 6, 0],
     [2, 0, 3, 8, 5],
     [0, 3, 0, 0, 7],
     [6, 8, 0, 0, 9],
     [0, 5, 7, 9, 0]]

selected_node = [0, 0, 0, 0, 0]

no_edge = 0

selected_node[0] = True

print("Edge     Weight\n")
while (no_edge < N - 1):

    minimum = INF
    a = 0
    b = 0
    for m in range(N):
        if selected_node[m]:
            for n in range(N):
                if ((not selected_node[n]) and G[m][n]):
                    # not in selected and there is an edge
                    if minimum > G[m][n]:
                        minimum = G[m][n]
                        a = m
                        b = n
    print(str(a) + "-" + str(b) + "        " + str(G[a][b]))
    selected_node[b] = True
    no_edge += 1
```

OUTPUT :

```
Edge     Weight

0-1        2
1-2        3
1-4        5
0-3        6
```