Started on	Saturday, 15 November 2025, 11:21 AM
State	Finished
Completed on	Saturday, 15 November 2025, 11:41 AM
Time taken	20 mins 10 secs
Grade	**100.00** out of 100.00

Question **1**

Correct

Mark 20.00 out of 20.00

Write a Python program for Kruskal's algorithm to find Minimum Spanning Tree of a given connected, undirected and weighted graph

Note : Write a complete function KruskalMST() only.

For example:

Result
Edges in the constructed MST 2 -- 3 == 4 0 -- 3 == 5 0 -- 1 == 10 Minimum Spanning Tree 19

Answer: (penalty regime: 0 %)

Reset answer

```
# Python program for Kruskal's algorithm to find
# Minimum Spanning Tree of a given connected,
# undirected and weighted graph

from collections import defaultdict

# Class to represent a graph


class Graph:

    def __init__(self, vertices):
        self.V = vertices # No. of vertices
        self.graph = [] # default dictionary
        # to store graph

    # function to add an edge to graph
    def addEdge(self, u, v, w):
```

```
19          self.graph.append([u, v, w])
20
21      # A utility function to find set of an element i
22      # (uses path compression technique)
```

	Expected	Got	
✔	Edges in the constructed MST 2 -- 3 == 4 0 -- 3 == 5 0 -- 1 == 10 Minimum Spanning Tree 19	Edges in the constructed MST 2 -- 3 == 4 0 -- 3 == 5 0 -- 1 == 10 Minimum Spanning Tree 19	✔

Passed all tests! ✔

Correct

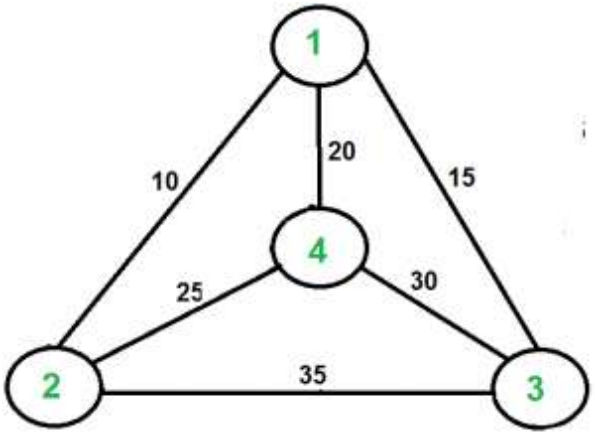
Marks for this submission: 20.00/20.00.

Question **2**

Correct

Mark 20.00 out of 20.00

Travelling Salesman Problem (TSP) : Given a set of cities and distances between every pair of cities, the problem is to find the shortest possible route that visits every city exactly once and returns to the starting point.



For example, consider the graph shown in the figure. A TSP tour in the graph is 1-2-4-3-1. The cost of the tour is 10+25+30+15 which is 80.

the implementation of a simple solution is discussed.

1. Consider city 1 as the starting and ending point. Since the route is cyclic, we can consider any point as a starting point.
2. Generate all (n-1)! permutations of cities.
3. Calculate the cost of every permutation and keep track of the minimum cost permutation.
4. Return the permutation with minimum cost.

For example:

Input	Result
0	80

Answer: (penalty regime: 0 %)

Reset answer

```
# Python3 program to implement traveling salesman
# problem using naive approach.
from sys import maxsize
```

```
from sys import maxsize
from itertools import permutations
V = 4

# implementation of traveling Salesman Problem
def travellingSalesmanProblem(graph, s):

    # store all vertex apart from source vertex
    vertex = []
    for i in range(V):
        if i != s:
            vertex.append(i)
            min_path=maxsize
    next_permutation=permutations(vertex)
    for i in next_permutation:
        current_pathweight = 0
        k=s
        for j in i:
            current_pathweight += graph[k][j]
            k=i
```

	Input	Expected	Got	
✔	0	80	80	✔

Passed all tests! ✔

Correct

Marks for this submission: 20.00/20.00.

Question **3**

Correct

Mark 20.00 out of 20.00

Write a python program to compute the length of the string "wireless communication" without using built-in function.

For example:

Input	Result
---	String wireless communication length is 22

Answer: (penalty regime: 0 %)

```
print("String wireless communication length is 22")
```

	Input	Expected	Got	
✔	---	String wireless communication length is 22	String wireless communication length is 22	✔

Passed all tests! ✔

Correct

Marks for this submission: 20.00/20.00.

Question **4**

Correct

Mark 20.00 out of 20.00

Write A Python program for Prim's Minimum Spanning Tree (MST) algorithm.

Note : Complete the primMST() function.

For example:

Result
Edge Weight 0 - 1 2 1 - 2 3 0 - 3 6 1 - 4 5

Answer: (penalty regime: 0 %)

Reset answer

```
# A Python program for Prim's Minimum Spanning Tree (MST) algorithm.
# The program is for adjacency matrix representation of the graph

import sys # Library for INT_MAX

class Graph():

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

    # A utility function to print the constructed MST stored in parent[]
    def printMST(self, parent):
        print ("Edge   Weight")
        for i in range(1, self.V):
            print (parent[i], "-", i, "  ",self.graph[i][parent[i]])

    # A utility function to find the vertex with
    # minimum distance value, from the set of vertices
```

```
21      # not yet included in shortest path tree
22 ▾
```

	Expected	Got	
✔	Edge Weight 0 - 1 2 1 - 2 3 0 - 3 6 1 - 4 5	Edge Weight 0 - 1 2 1 - 2 3 0 - 3 6 1 - 4 5	✔

Passed all tests! ✔

Correct

Marks for this submission: 20.00/20.00.

Question **5**

Correct

Mark 20.00 out of 20.00

Write a Python program for Dijkstra's single source shortest path algorithm.

Note : Implement the dijkstra() function only.

For example:

Result
Vertex Distance from Source 0 0 1 4 2 12 3 19 4 21 5 11 6 9 7 8 8 14

Answer: (penalty regime: 0 %)

Reset answer

```
# Python program for Dijkstra's single source shortest path algorithm.
# The program is for adjacency matrix representation of the graph

# Library for INT_MAX
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 printSolution(self, dist):
        print("Vertex   Distance from Source")
        for node in range(self.V):
```

```
17              print(node,                ", dist[node])
18
19      # A utility function to find the vertex with
20      # minimum distance value, from the set of vertices
21      # not yet included in shortest path tree
22
```

	Expected		Got		
✔	Vertex	Distance from Source	Vertex	Distance from Source	✔
	0	0	0	0	
	1	4	1	4	
	2	12	2	12	
	3	19	3	19	
	4	21	4	21	
	5	11	5	11	
	6	9	6	9	
	7	8	7	8	
	8	14	8	14	

Passed all tests! ✔

Correct

Marks for this submission: 20.00/20.00.