IARE

IARE
INSTITUTE OF AERONAUTICAL ENGINEERING
(An Autonomous Institute affiliated to JNTUH, Hyderabad)
Dundigal, Hyderabad - 500 043

LABORATORY WORK SHEET

Name of the Student : Muhd. fauzan zohaib

Class : CSE-C Semester : II

Roll Number									
:	3	9	5	1	A	0	5	5	H

Course Code : ACSD06 Course Name : PPS Lab

Name of the Course Faculty : Dr. M. Madhusudhan reddy Faculty ID : IARE 10881

Exercise Number : Week Number : Date :

DAY TO DAY EVALUATION:

Marks	Aim / Preparation	Algorithm / Procedure / Performance in the Lab	Source Code / Calculations and Graphs	Program Execution / Results and Error Analysis	Viva - Voce	Total
Max. Marks	4	4	4	4	4	20
Obtained	4	4	4	4	4	20

Signature of Faculty 26/6/24

START WRITING FROM HERE :

Kruskal's Algorithm:

```
Class disjoint_set:
    def __init__(self, vertices):
        self.parent = {v:v for v in vertices}
    def find_root(self,v):
        if self.parent[v] != v:
            self.parent[v] = self.find_root[self.parent[v]]
        return self.parent[v]
    def union(self, v1, v2):
        p1 = self.find_root(v1)
        p2 = self.find_root(v2)
        self.parent[p1] = p2
```

ROLL NUMBER : 23ITA0557

```
def kruskal (vertices ,edges):
    edge_list . sort ()
    mst = []
    obj = disjoint - set (vertices)
    for edge in edge_list:
        w, s, d = edge
        P1 = obj. find_root (s)
        P2 = obj. find_root (d)
        if P1 != P2 :
            mst .append (edge)
            obj.union (s,d)
    return mst
nv, ne = list (map (int, input () split (' ' )))
vertices = list (range (nv))
edges = []
for i in range (ne):
    edge = tuple (map(int ,input () . split (',')))
    edges. append(edge)
mst = kruskal (vertices ,edges)
print (mst)
for edge in mst:
    tot = tot + edge [0]
print (tot).
```

ROLL NUMBER: 2395IA05571

Prim's Algorithm

```
import heapq
def prim(adj_list, start, dest):
    mst = []
    visited = set()
    heap = [(0, start)]
    while heap:
        w, d = heapq.heappop(heap)
        if d is not in visited:
            visited.add(d)
            mst.append((w,d))
            for w,n in adj-list[d]:
                if n not in visited:
                    heap.heappush(heap)(w, n))
    return mst
nv, ne = map(int, input().split(' '))
edges = []
for i in range(ne):
    edge = tuple(map(int, input().split(' ')))
    edges = append(edge)
adj_list = {v:[] for v in range(nv)}
for w, v1, v2 in edges
        adj_list[v1].append((w, v2))
        adj_list[v2].append((w, v1))
```

ROLL NUMBER :

```
start = 0
mst = prim(adj_list, start, start)
tot = 0
for e in mst:
    tot += e[0]
print(tot)
```