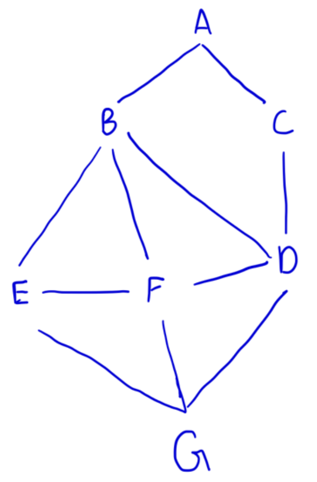
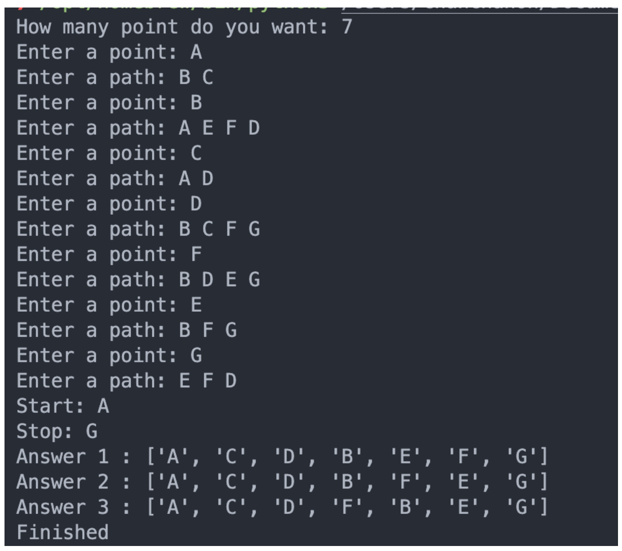
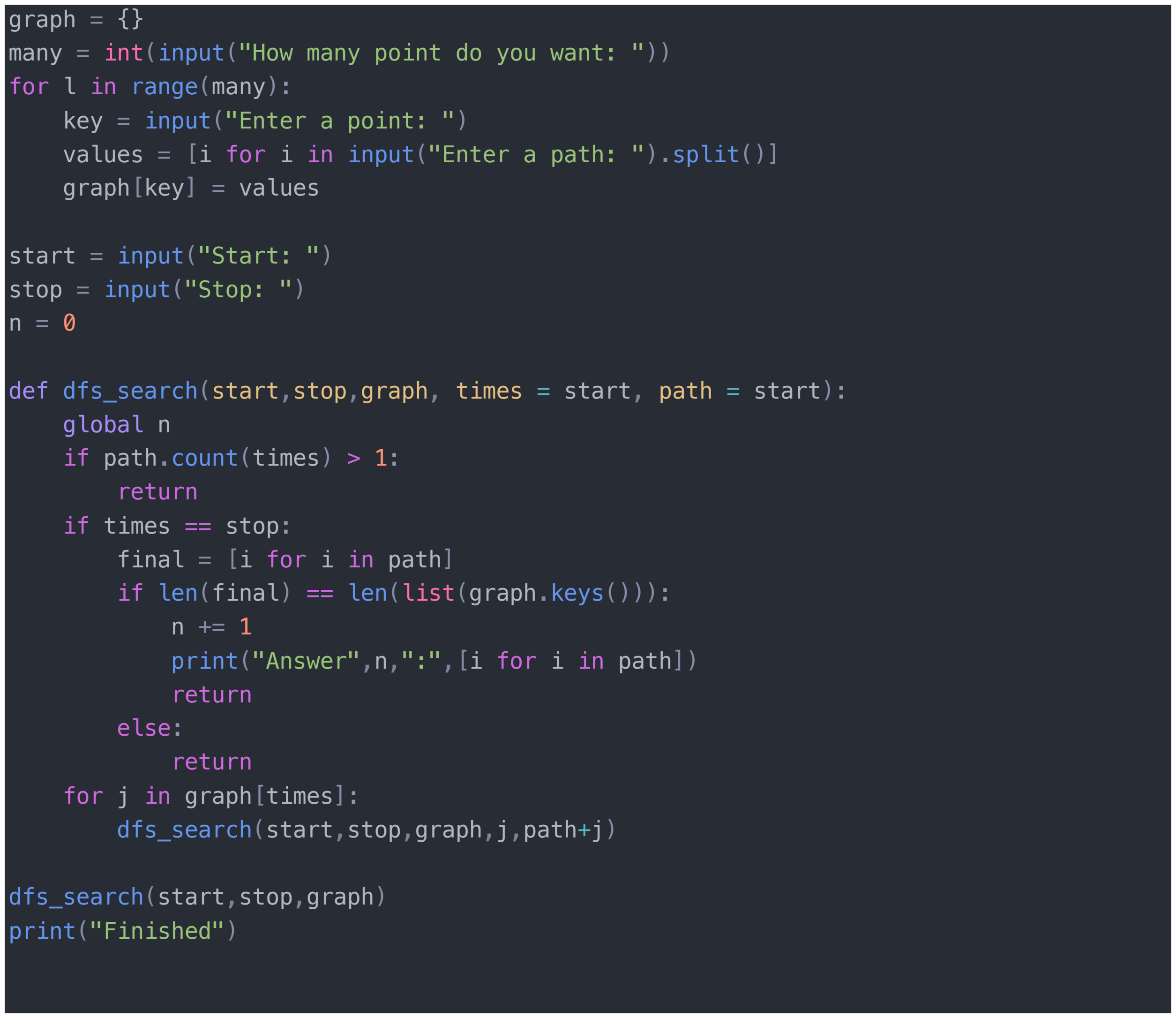
1)



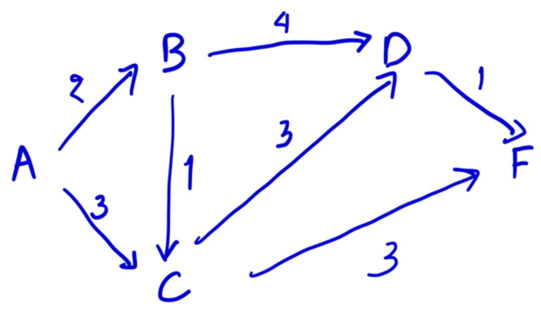


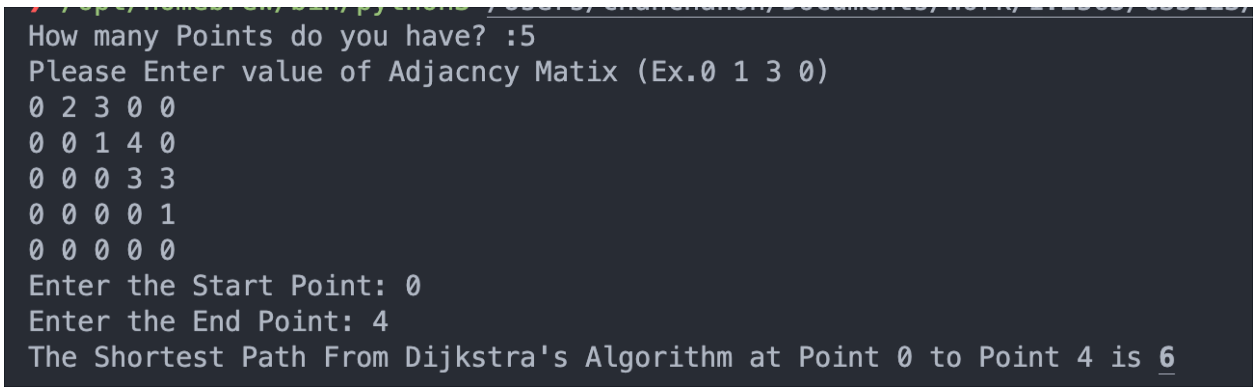
```
How many point do you want: 7
Enter a point: A
Enter a path: B C
Enter a point: B
Enter a path: A E F D
Enter a point: C
Enter a path: A D
Enter a point: D
Enter a path: B C F G
Enter a point: F
Enter a path: B D E G
Enter a point: E
Enter a path: B F G
Enter a point: G
Enter a path: E F D
Start: A
Stop: G
Answer 1 : ['A', 'C', 'D', 'B', 'E', 'F', 'G']
Answer 2 : ['A', 'C', 'D', 'B', 'F', 'E', 'G']
Answer 3 : ['A', 'C', 'D', 'F', 'B', 'E', 'G']
Finished
```



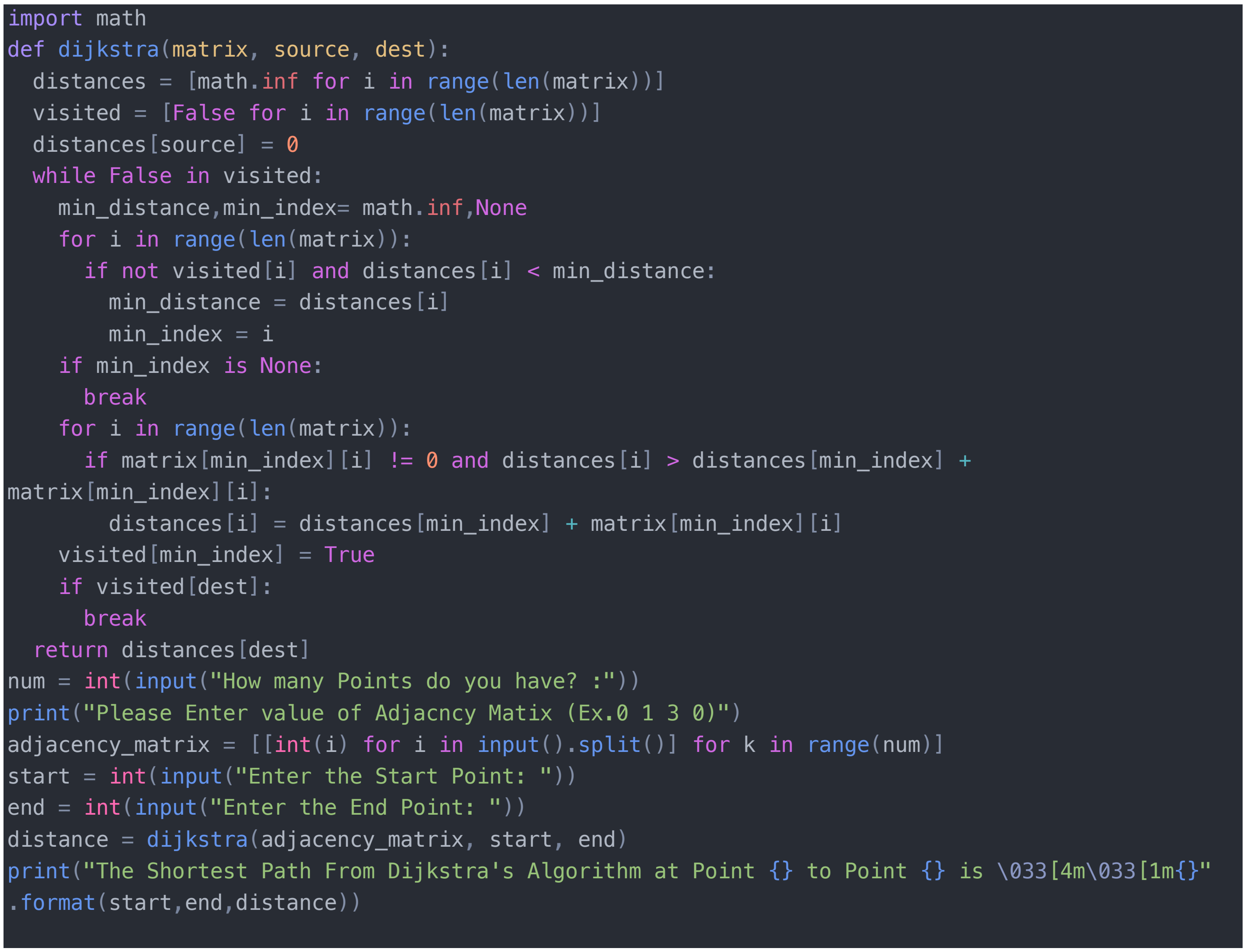
```python
graph = {}
many = int(input("How many point do you want: "))
for l in range(many):
    key = input("Enter a point: ")
    values = [i for i in input("Enter a path: ").split()]
    graph[key] = values

start = input("Start: ")
stop = input("Stop: ")
n = 0

def dfs_search(start,stop,graph, times = start, path = start):
    global n
    if path.count(times) > 1:
        return
    if times == stop:
        final = [i for i in path]
        if len(final) == len(list(graph.keys())):
            n += 1
            print("Answer",n,":",[i for i in path])
            return
        else:
            return
    for j in graph[times]:
        dfs_search(start,stop,graph,j,path+j)

dfs_search(start,stop,graph)
print("Finished")
```

2)



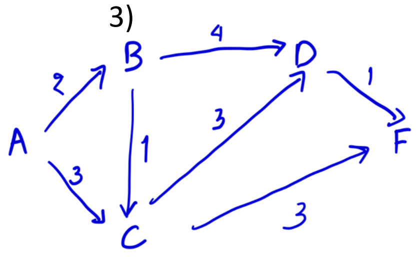


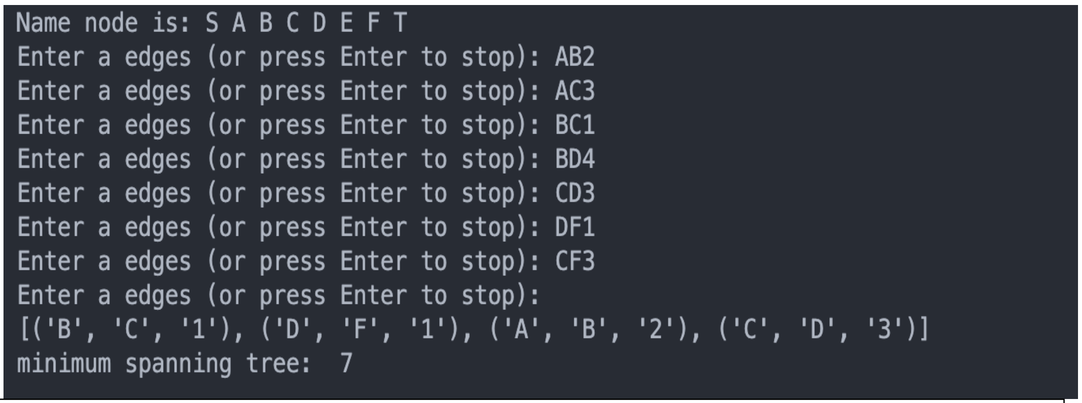
```
How many Points do you have? :5
Please Enter value of Adjacncy Matix (Ex.0 1 3 0)
0 2 3 0 0
0 0 1 4 0
0 0 0 3 3
0 0 0 0 1
0 0 0 0 0
Enter the Start Point: 0
Enter the End Point: 4
The Shortest Path From Dijkstra's Algorithm at Point 0 to Point 4 is 6
```



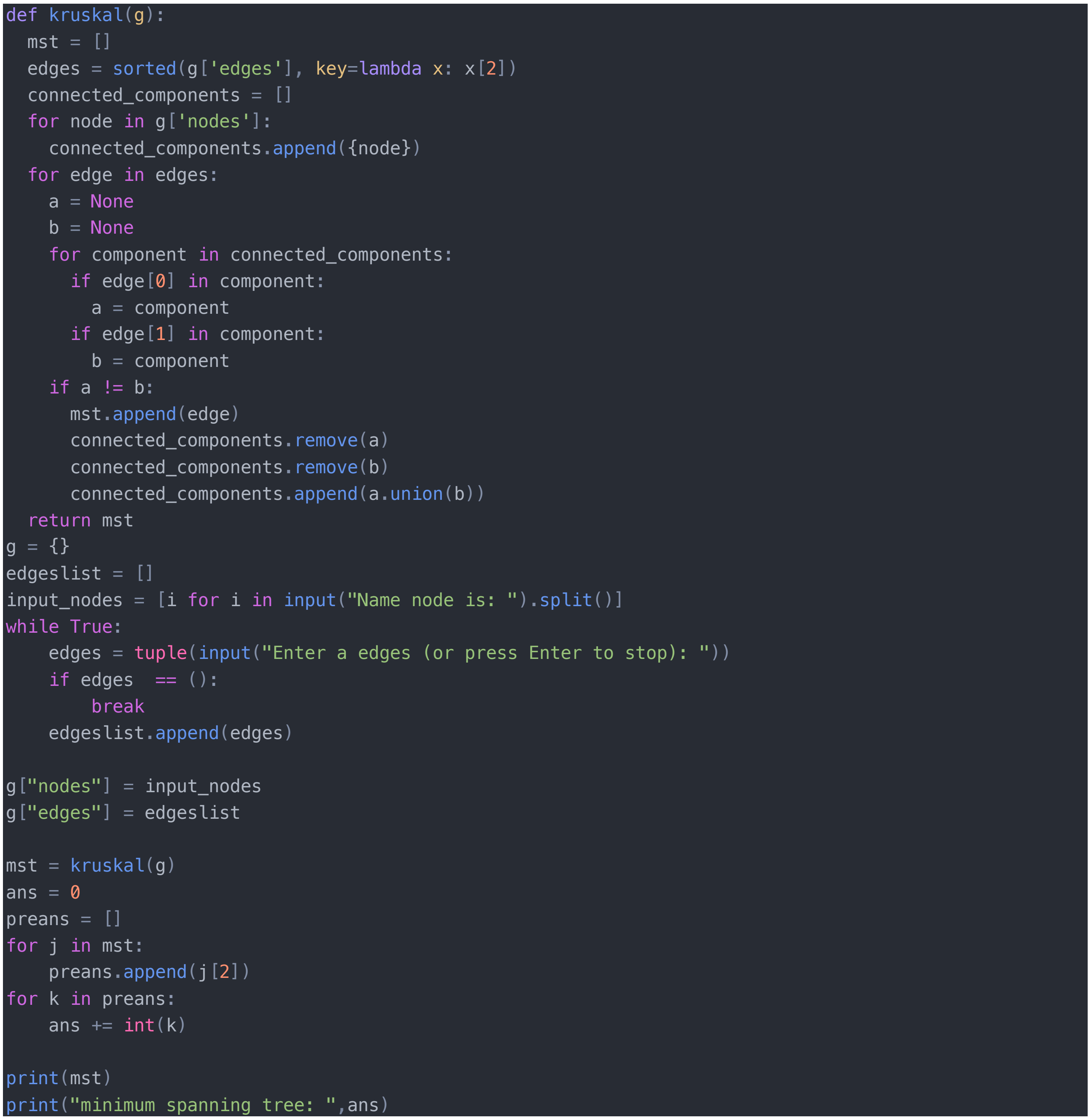
```python
import math
def dijkstra(matrix, source, dest):
  distances = [math.inf for i in range(len(matrix))]
  visited = [False for i in range(len(matrix))]
  distances[source] = 0
  while False in visited:
    min_distance,min_index= math.inf,None
    for i in range(len(matrix)):
      if not visited[i] and distances[i] < min_distance:
        min_distance = distances[i]
        min_index = i
    if min_index is None:
      break
    for i in range(len(matrix)):
      if matrix[min_index][i] != 0 and distances[i] > distances[min_index] +
matrix[min_index][i]:
        distances[i] = distances[min_index] + matrix[min_index][i]
    visited[min_index] = True
    if visited[dest]:
      break
  return distances[dest]
num = int(input("How many Points do you have? :"))
print("Please Enter value of Adjacncy Matix (Ex.0 1 3 0)")
adjacency_matrix = [[int(i) for i in input().split()] for k in range(num)]
start = int(input("Enter the Start Point: "))
end = int(input("Enter the End Point: "))
distance = dijkstra(adjacency_matrix, start, end)
print("The Shortest Path From Dijkstra's Algorithm at Point {} to Point {} is \033[4m\033[1m{}"
.format(start,end,distance))
```

3)





```
Name node is: S A B C D E F T
Enter a edges (or press Enter to stop): AB2
Enter a edges (or press Enter to stop): AC3
Enter a edges (or press Enter to stop): BC1
Enter a edges (or press Enter to stop): BD4
Enter a edges (or press Enter to stop): CD3
Enter a edges (or press Enter to stop): DF1
Enter a edges (or press Enter to stop): CF3
Enter a edges (or press Enter to stop):
[('B', 'C', '1'), ('D', 'F', '1'), ('A', 'B', '2'), ('C', 'D', '3')]
minimum spanning tree:  7
```



```python
def kruskal(g):
  mst = []
  edges = sorted(g['edges'], key=lambda x: x[2])
  connected_components = []
  for node in g['nodes']:
    connected_components.append({node})
  for edge in edges:
    a = None
    b = None
    for component in connected_components:
      if edge[0] in component:
        a = component
      if edge[1] in component:
        b = component
    if a != b:
      mst.append(edge)
      connected_components.remove(a)
      connected_components.remove(b)
      connected_components.append(a.union(b))
  return mst
g = {}
edgeslist = []
input_nodes = [i for i in input("Name node is: ").split()]
while True:
    edges = tuple(input("Enter a edges (or press Enter to stop): "))
    if edges  == ():
        break
    edgeslist.append(edges)

g["nodes"] = input_nodes
g["edges"] = edgeslist

mst = kruskal(g)
ans = 0
preans = []
for j in mst:
    preans.append(j[2])
for k in preans:
    ans += int(k)

print(mst)
print("minimum spanning tree: ",ans)
```