```
#genrate adjacency list
from collections import defaultdict
v = int(input("enter number of vertices: "))
e = int(input("enter number of edges: "))
graph = defaultdict(list)
for i in range(e):
    u1,u2=input("enter vertices").split()
    graph[u1].append(u2)
    if u1!=u2:
        graph[u2].append(u1)
for i in graph:
    print(i,graph[i])
```

```
enter number of vertices: 4
enter number of edges: 5
enter verticesv1 v2
enter verticesv1 v3
enter verticesv1 v4
enter verticesv3 v4
enter verticesv2 v3
v1 ['v2', 'v3', 'v4']
v2 ['v1', 'v3']
v3 ['v1', 'v4', 'v2']
v4 ['v1', 'v3']
```

```
from collections import defaultdict

def createGraph():
    v = int(input("Enter the number of vertices: "))
    e = int(input("Enter the number of edges: "))
    graph = defaultdict(list)
    for i in range(e):
        u1, u2 = input(f"Enter edge {i + 1} (two vertices separated by space): ").s
        graph[u1].append(u2)
        if u1 != u2:  # Avoid duplicate entry for self-loops
            graph[u2].append(u1)
    print("Graph representation (Adjacency List):")
    for vertex in graph:
        print(f"{vertex}: {graph[vertex]}")
    return graph

def verticesDict(graph):
    vertices_list = list(graph.keys())
    vertices_dict = {vertices_list[i]: i for i in range(len(vertices_list))}
    return vertices_dict

def adjacencyMatrix(graph, vertices_dict):
    n = len(vertices_dict)  # Number of vertices
    adjacency_matrix = [[0] * n for _ in range(n)]  # Initialize n x n matrix with

    for key in graph:
        i = vertices_dict[key]
        for edge in graph[key]:
            j = vertices_dict[edge]
```

```
            if i == j:
                adjacency_matrix[i][j] = 2  # Self-loop
            else:
                adjacency_matrix[i][j] = 1  # Regular edge

    print("\nAdjacency Matrix:")
    for row in adjacency_matrix:
        print(*row)

# Main execution
graph = createGraph()
vertices_dict = verticesDict(graph)
adjacencyMatrix(graph, vertices_dict)
```

```
Enter the number of vertices: 4
Enter the number of edges: 6
Enter edge 1 (two vertices separated by space): a b
Enter edge 2 (two vertices separated by space): a c
Enter edge 3 (two vertices separated by space): a d
Enter edge 4 (two vertices separated by space): b c
Enter edge 5 (two vertices separated by space): c d
Enter edge 6 (two vertices separated by space): a a
Graph representation (Adjacency List):
a: ['b', 'c', 'd', 'a']
b: ['a', 'c']
c: ['a', 'b', 'd']
d: ['a', 'c']

Adjacency Matrix:
2 1 1 1
1 0 1 0
1 1 0 1
1 0 1 0
```

```
# Import required libraries
import pandas as pd
from collections import defaultdict
import numpy as np

# Load the dataset
file_path = "teams_data.csv"  # Replace with your file's path
df = pd.read_csv(file_path)

# Create a dictionary mapping each country to its teams
country_to_teams = defaultdict(list)
for _, row in df.iterrows():
    country_to_teams[row['country']].append(row['team'])

# Flatten this structure into edges
edges = []
for teams in country_to_teams.values():
    for i in range(len(teams)):
        for j in range(i + 1, len(teams)):
            edges.append((teams[i], teams[j]))

# Get a list of unique team names
```

```
team_names = sorted(set(df['team']))
team_index = {team: idx for idx, team in enumerate(team_names)}

# Initialize an empty adjacency matrix
adj_matrix = np.zeros((len(team_names), len(team_names)), dtype=int)

# Populate the adjacency matrix
for team1, team2 in edges:
    i, j = team_index[team1], team_index[team2]
    adj_matrix[i, j] = 1
    adj_matrix[j, i] = 1  # Undirected graph

# Convert to DataFrame for better visualization
adj_df = pd.DataFrame(adj_matrix, index=team_names, columns=team_names)

# Print the adjacency matrix
print(adj_df)
```

```
                          AC Milan  AC Sparta Praha  AS Monaco  Arsenal FC  \
AC Milan                         0                0          0           0
AC Sparta Praha                  0                0          0           0
AS Monaco                        0                0          0           0
Arsenal FC                       0                0          0           0
Aston Villa FC                   0                0          0           1
Atalanta BC                      1                0          0           0
Atlético de Madrid               0                0          0           0
BSC Young Boys                   0                0          0           0
Bayer 04 Leverkusen              0                0          0           0
Bologna FC 1909                  1                0          0           0
Borussia Dortmund                0                0          0           0
Celtic FC                        0                0          0           0
Club Brugge KV                   0                0          0           0
FC Barcelona                     0                0          0           0
FC Bayern München                0                0          0           0
FC Internazionale Milano         1                0          0           0
FC Salzburg                      0                0          0           0
FC Shakhtar Donetsk              0                0          0           0
FK Crvena Zvezda                 0                0          0           0
Feyenoord                        0                0          0           0
GNK Dinamo                       0                0          0           0
Girona FC                        0                0          0           0
Juventus                         1                0          0           0
LOSC Lille                       0                0          1           0
Liverpool FC                     0                0          0           1
Manchester City                  0                0          0           1
PSV Eindhoven                    0                0          0           0
Paris Saint-Germain              0                0          1           0
RB Leipzig                       0                0          0           0
Real Madrid C.F.                 0                0          0           0
SK Sturm Graz                    0                0          0           0
SL Benfica                       0                0          0           0
Sporting Clube de Portugal       0                0          0           0
Stade Brestois 29                0                0          1           0
VfB Stuttgart                    0                0          0           0
ŠK Slovan Bratislava             0                0          0           0

                          Aston Villa FC  Atalanta BC  Atlético de Madrid  \
```

AC Milan	0	1	0
AC Sparta Praha	0	0	0
AS Monaco	0	0	0
Arsenal FC	1	0	0
Aston Villa FC	0	0	0
Atalanta BC	0	0	0
Atlético de Madrid	0	0	0
BSC Young Boys	0	0	0
Bayer 04 Leverkusen	0	0	0
Bologna FC 1909	0	1	0
Borussia Dortmund	0	0	0
Celtic FC	0	0	0
Club Brugge KV	0	0	0
FC Barcelona	0	0	1
FC Bayern München	0	0	0
FC Internazionale Milano	0	1	0
FC Salzburg	0	0	0
FC Shakhtar Donetsk	0	0	0
FK Crvena Zvezda	0	0	0

```
# Function to print the adjacency matrix
def printMatrix(matrix):
    r, c = len(matrix), len(matrix[0])
    for i in range(r):  # Corrected row iteration
        for j in range(c):  # Corrected column iteration
            print(matrix[i][j], end=" ")
        print()

# Input for number of vertices and edges
v, e = map(int, input("Enter number of vertices and edges: ").split())

# Initialize adjacency matrix for directed weighted graph
matrix = [[0] * v for _ in range(v)]

# Input edges and their weights
for _ in range(e):
    u, v, w = input("Enter edge (u v) and weight w: ").split()
    u = ord(u) - ord('a')  # Convert vertex 'a'-based indexing to integer
    v = ord(v) - ord('a')  # Same for target vertex
    w = int(w)
    matrix[u][v] = w  # Set weight for directed edge

# Print the adjacency matrix
printMatrix(matrix)
```

```
Enter number of vertices and edges: 3 3
Enter edge (u v) and weight w: a b 2
Enter edge (u v) and weight w: b c 3
Enter edge (u v) and weight w: c a 4
0 2 0
0 0 3
4 0 0
```

Start coding or generate with AI.