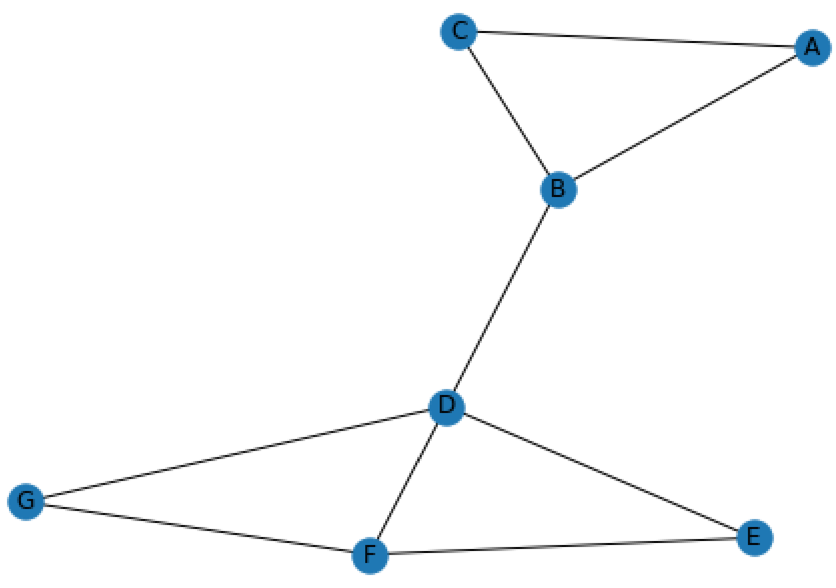
Question a

In [12]:

```
import networkx as nx
import numpy as np
# To create an empty undirected graph
G = nx.Graph()

# To add a node
G.add_node('A')
G.add_node('B')
G.add_node('C')
G.add_node('D')
G.add_node('E')
G.add_node('F')
G.add_node('G')

# To add an edge
# Note graph is undirected
# Hence order of nodes in edge doesn't matter
G.add_edges_from([('A','B'),('A','C'),('B','C'),('B','D'),('D','E'),('D','F'),('D','G'),('E','F'),('F','G')])

node_list = G.nodes()
print("Nodes")
print(node_list)

# To get all the edges of a graph
edge_list = G.edges()
print("Edges")
print(edge_list)

nx.draw(G, with_labels = True)
```

```
Nodes
['A', 'B', 'C', 'D', 'E', 'F', 'G']
Edges
[('A', 'B'), ('A', 'C'), ('B', 'C'), ('B', 'D'), ('D', 'E'), ('D', 'F'), ('D', 'G'), ('E', 'F'), ('F', 'G')]
```



Question b

In [22]:

```
A = nx.adjacency_matrix(G)
print(A.todense())
```

```
[[0 1 1 0 0 0 0]
 [1 0 1 1 0 0 0]
 [1 1 0 0 0 0 0]
 [0 1 0 0 1 1 1]
 [0 0 0 1 0 1 0]
 [0 0 0 1 1 0 1]
 [0 0 0 1 0 1 0]]
```

In [23]:

```
D = [G.degree[node] for node in G.nodes()]
D = np.diag(D)
D
```

Out[23]:

```
array([[2, 0, 0, 0, 0, 0, 0],
       [0, 3, 0, 0, 0, 0, 0],
       [0, 0, 2, 0, 0, 0, 0],
       [0, 0, 0, 4, 0, 0, 0],
       [0, 0, 0, 0, 2, 0, 0],
       [0, 0, 0, 0, 0, 3, 0],
       [0, 0, 0, 0, 0, 0, 2]])
```

Question c

In [85]:

```
D = G.degree
D = dict(D)
sort_orders = sorted(D.items(), key=lambda x: x[1], reverse=True)

for i in sort_orders:
    print(i[0], i[1])
```

```
D 4
B 3
F 3
A 2
C 2
E 2
G 2
```

Question d

In [117…

```
A = nx.adjacency_matrix(G).todense()
D = [G.degree[node] for node in G.nodes()]
D = np.diag(D).tolist()
L = D-A
```

In [118…

```
L    # 2 marks
```

Out[118…

```
matrix([[ 2, -1, -1,  0,  0,  0,  0],
        [-1,  3, -1, -1,  0,  0,  0],
        [-1, -1,  2,  0,  0,  0,  0],
        [ 0, -1,  0,  4, -1, -1, -1],
        [ 0,  0,  0, -1,  2, -1,  0],
        [ 0,  0,  0, -1, -1,  3, -1],
        [ 0,  0,  0, -1,  0, -1,  2]])
```

In [130…

```
from numpy import linalg as LA
eigenvalues, eigenvectors = LA.eig(L)
```

In [131…

```
eigenvalues
```

Out[131…

```
array([-2.77555756e-16,  3.98320868e-01,  5.26180225e+00,  3.00000000e+00,
        3.33987689e+00,  4.00000000e+00,  2.00000000e+00])
```

In [132…

```
eigenvectors
```

Out[132…

```
matrix([[-3.77964473e-01, -4.92886500e-01, -1.06502348e-01,
          7.07106781e-01, -3.20722630e-01, -4.78450776e-17,
          5.64378438e-17],
        [-3.77964473e-01, -2.96559521e-01,  4.53891948e-01,
         -2.02752687e-15,  7.50451469e-01,  2.93408927e-16,
          1.93282527e-16],
        [-3.77964473e-01, -4.92886500e-01, -1.06502348e-01,
         -7.07106781e-01, -3.20722630e-01, -1.97718772e-16,
         -1.93282527e-16],
        [-3.77964473e-01,  2.14220282e-01, -8.13609130e-01,
         -7.65809099e-16,  3.86384151e-01,  1.02028617e-16,
          8.04068397e-17],
        [-3.77964473e-01,  3.56037413e-01,  1.90907293e-01,
          5.71937510e-16, -1.65130120e-01,  4.08248290e-01,
         -7.07106781e-01],
        [-3.77964473e-01,  3.56037413e-01,  1.90907293e-01,
          5.14722129e-16, -1.65130120e-01, -8.16496581e-01,
          1.55422026e-17],
        [-3.77964473e-01,  3.56037413e-01,  1.90907293e-01,
          5.14722129e-16, -1.65130120e-01,  4.08248290e-01,
          7.07106781e-01]])
```

In [126…

```
max(eigenvalues)
```

Out[126…

```
5.261802245259971
```

In [134…

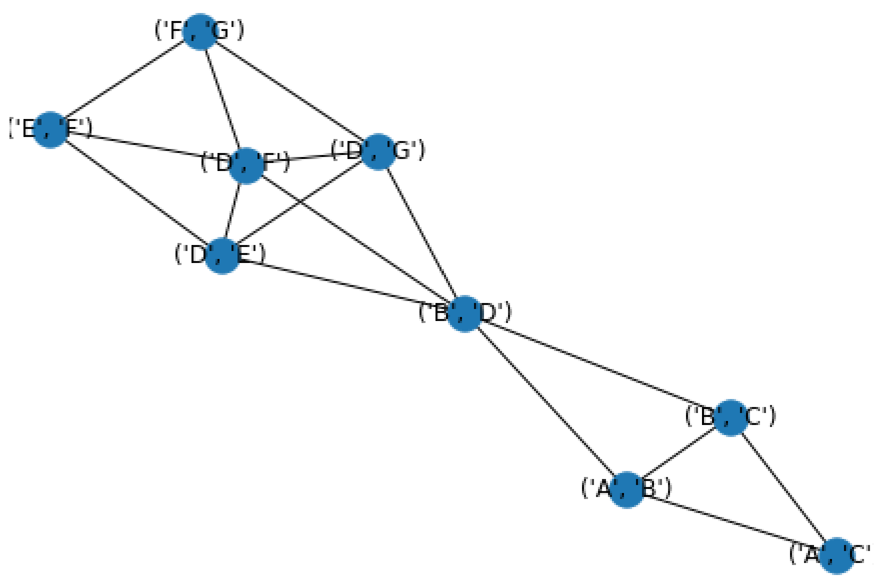
```
# node C
```

Question e

G' is the dual graph of G. The graph G' constructed from graph G has a node for each edge in G and an edge joining those nodes if the two edges in G share a common node.

In [89]:

```
L=nx.line_graph(G)
# nx.draw(G, with_labels=True)
nx.draw(L, with_labels=True)
```



Question f

The given graph G can be converted to a tri-partite graph (K=3). For a k-partite graph, we have k types of nodes and edges between unlike nodes only. Here, in G we have 3 triangles (3 nodes connected with each other). So, k cannot be 2. The possible sets of nodes are: [A,D];[B,G,E];[C,F] or, [A,E,G];[B,F];[C,D] or, [A,F];[B,E,G];[C,D] or, [A,D];[B,F];[C,E,G].

In [ ]: