CS 5483 Homework 1

1. The graph is not balanced. For a graph to be balanced, there cannot exist a cycle that is unbalanced. For a triangle of nodes to be balanced, we must have $w_{ij}w_{jk}w_{ik} \geq 0$, where w_{ij} represents the value of the edge between nodes *i* and *j* (in our case, a 1 or -1 depending on the positivity or negativity of the edge). Looking at nodes 2, 3, and 4, we can see that $w_{23}w_{34}w_{24} = (-1)(1)(1) = -1$ which is obviously not ≥ 0. Therefore, the graph is unbalanced.
2.
    a. Degree Centrality

    Degree Centrality can be calculated by $c_i = \frac{d_i}{n-1}$.

    1: 3/9 = 1/3

    2: 1/3

    3: 1/3

    4: 2/9

    5: 1/3

    6: 1/3

    7: 1/3

    8: 1/3

    9: 1/3

    10: 2/9

    For the following centrality calculations, I wrote scripts in Python using the NetworkX Library

```
G = nx.Graph()
G.add_edge(1, 3)
G.add_edge(1, 2)
G.add_edge(1, 10)
G.add_edge(2, 4)
G.add_edge(2, 7)
G.add_edge(3, 5)
G.add_edge(3, 8)
G.add_edge(4, 6)
G.add_edge(5, 6)
G.add_edge(5, 9)
G.add_edge(6, 7)
G.add_edge(7, 8)
G.add_edge(8, 9)
G.add_edge(9, 10)
```

b. Eigenvector Centrality:

```
nx.eigenvector_centrality(G)
```

```
{1: 0.3104883904772362,
 3: 0.3579907069964069,
 2: 0.3063031282514309,
 10: 0.2231838875316019,
 4: 0.21941202212603364,
 7: 0.34561011840798544,
 5: 0.35205192749011016,
 8: 0.3607164531280917,
 6: 0.3208440534144026,
 9: 0.3274463984152574}
```

c. PageRank Centrality:

```
nx.pagerank(G)
```

```
{1: 0.1073109558303194,
 3: 0.10491482880528645,
 2: 0.10745807640817177,
 10: 0.07562231436914846,
 4: 0.07572421188603344,
 7: 0.10542104280136726,
 5: 0.10522110721859218,
 8: 0.10481254791560028,
 6: 0.10686587981512946,
 9: 0.10664903495035094}
```

d. Katz Centrality:

```
nx.katz_centrality(G)
```

```
{1: 0.32055872140499675,
 3: 0.3238045584037487,
 2: 0.3205331921654917,
 10: 0.29115201149015374,
 4: 0.29114640100740213,
 7: 0.32352934554074503,
 5: 0.3235599185730798,
 8: 0.3238295214709142,
 6: 0.3208333110004352,
 9: 0.32086388926988524}
```

3. For this question, I again wrote a Python script and found the assortativity coefficient using the NetworkX library.

```
G = nx.Graph()
G.add_edge('a', 'b')
G.add_edge('a', 'c')
G.add_edge('a', 'd')
G.add_edge('a', 'e')
G.add_edge('a', 'f')
G.add_edge('b', 'g')
G.add_edge('b', 'i')
G.add_edge('e', 'i')
G.add_edge('f', 'h')
```

```
nx.degree_assortativity_coefficient(G)
```

```
-0.45945945945945965
```

4.

Adjacency List:

1: 2, 3

2: 1, 3

3: 1, 2, 4, 5

4: 3, 5, 6, 7

5: 3, 4, 6, 7

6: 4, 5, 7, 8

7: 4, 5, 6, 8

8: 6, 7, 9

9: 8

Edge List:

1 2

1 3

2 3

3 4

3 5

4 5

4 6

4 7

5 6

5 7

6 7

6 8

7 8

8 9

5.

a. BFS
   Parent array:
   v1: -
   v2: v1
   v3: v2
   v4: v1
   v5: v2
   v6: v4

v7: v5
v8: v6
v9: v5

b. DFS
Parent array:
v1: -
v2: v1
v3: v2
v4: v5
v5: v3
v6: v4
v7: v6
v8: v7
v9: v8

c. Dijkstra's
Parent array:
v1: -
v2: v1
v3: v2
v4: v1
v5: v2
v6: v4
v7: v6
v8: v6
v9: v5