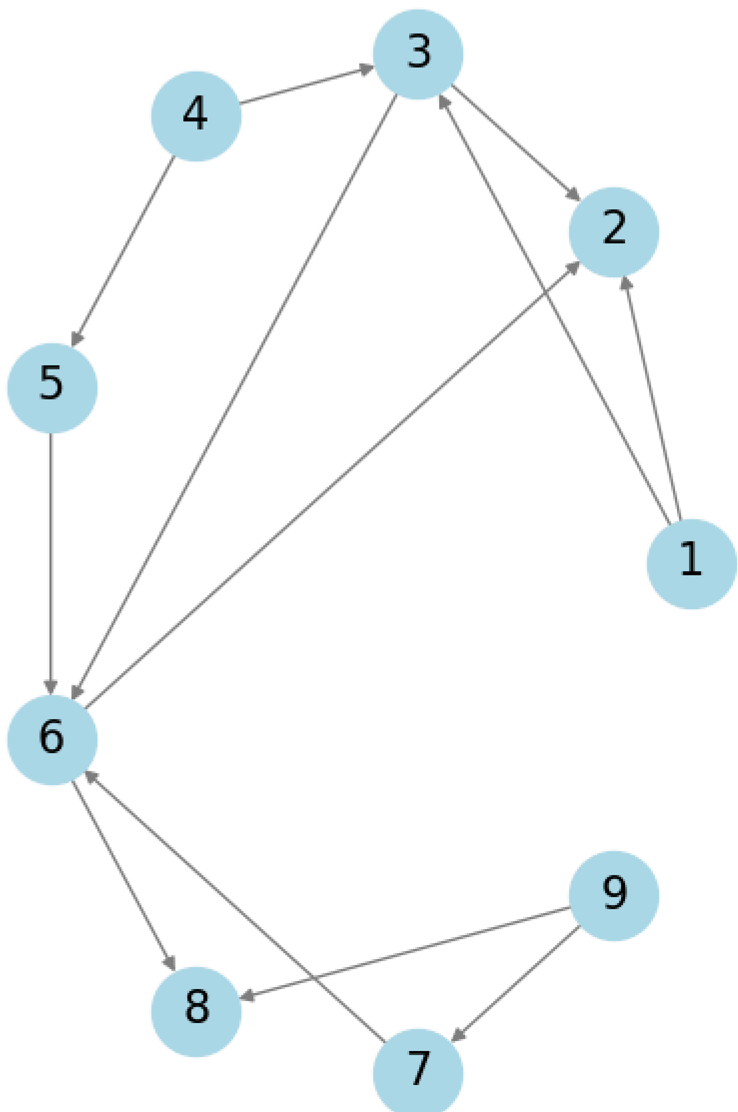
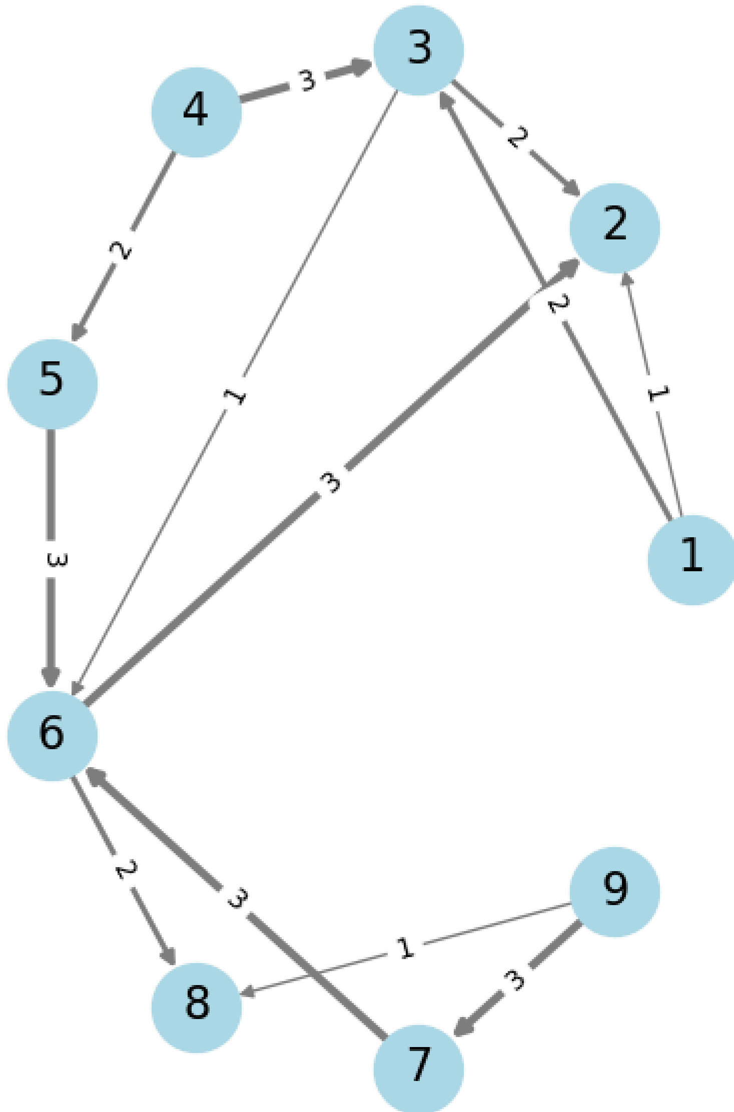
In [3]:
```
import networkx as nx
import matplotlib.pyplot as plt
import copy

# Create an empty graph
G = nx.DiGraph()

# Larger connected component (4 nodes)
G.add_weighted_edges_from([(1, 2, 1), (1, 3, 2), (3, 2, 2), (4, 3, 3), (4, 5, 2)])

# Smaller connected component (3 nodes)
G.add_weighted_edges_from([(5, 6, 3), (6, 8, 2), (7, 6, 3), (9, 7, 3), (9, 8, 1)])

# Connect two components
G.add_weighted_edges_from([(3, 6, 1), (6, 2, 3)])

pos = nx.circular_layout(G)

# Draw the graph without weight
fig, ax = plt.subplots(figsize=(5, 8))
nx.draw_networkx(
    G,
    pos=pos,
    width=1,
    with_labels=True,
    node_color="lightblue",
    edge_color="gray",
    node_size=1000,
    font_size=16,
)
ax.axis("off")  # remove the frame of the generated figure

plt.savefig(
    "/Users/dwu24/Desktop/Doc/CIE500_XF/week4/Example.jpg",
    dpi=600,
    bbox_inches="tight",
)
plt.show()
```

```python
# Draw the graph with weight
weights = {(u, v): d["weight"] for u, v, d in G.edges(data=True)}

weights_list = [d["weight"] for u, v, d in G.edges(data=True)]

fig, ax = plt.subplots(figsize=(5, 8))

nx.draw_networkx(
    G,
    pos=pos,
    width=weights_list,
    with_labels=True,
    node_color="lightblue",
    edge_color="gray",
    node_size=1000,
    font_size=16,
)
nx.draw_networkx_edge_labels(G, pos=pos, edge_labels=weights)
ax.axis("off")  # remove the frame of the generated figure

plt.savefig(
    "/Users/dwu24/Desktop/CIE500Fan/class1/Example_weights.jpg",
    dpi=600,
    bbox_inches="tight",
)
plt.show()


print(
    f"The adjancency matrix of G is \n {nx.adjacency_matrix(G, nodelist=list(range(1,7))).toarray()}"
)

# get the topological sort order of the graph.

sorted_order = list(nx.topological_sort(G))

print(f"the sorted order is {sorted_order}")

print(
    f"the lenght of sorted order is {len(sorted_order)}\n the total number of nodes is {len(G.nodes())}"
)
```

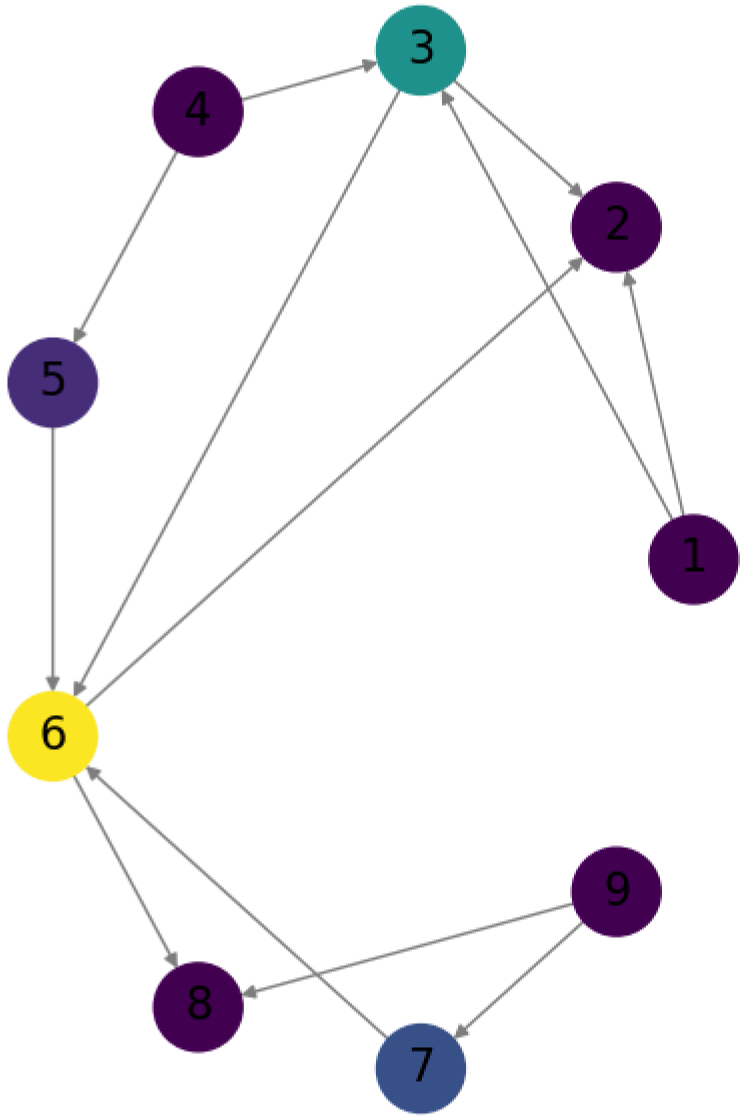
```
# the following codes calculate the betwenness centrality of all nodes of graph G.

node_central = nx.betweenness_centrality(G)

fig, ax = plt.subplots(figsize=(5, 8))

nx.draw_networkx(
    G,
    pos=pos,
    with_labels=True,
    node_color=list(node_central.values()),
    edge_color="gray",
    node_size=1000,
    font_size=16,
)
ax.axis("off")  # remove the frame of the generated figure
plt.savefig(
    "/Users/dwu24/Desktop/CIE500Fan/class1/node_betweenness.jpg",
    dpi=600,
    bbox_inches="tight",
)
plt.show()


node_clossness = nx.closeness_centrality(G)

fig, ax = plt.subplots(figsize=(5, 8))

nx.draw_networkx(
    G,
    pos=pos,
    with_labels=True,
    node_color=list(node_clossness.values()),
    edge_color="gray",
    node_size=1000,
    font_size=16,
)
ax.axis("off")  # remove the frame of the generated figure
plt.savefig(
    "/Users/dwu24/Desktop/CIE500Fan/class1/node_closeness.jpg",
    dpi=600,
    bbox_inches="tight",
```

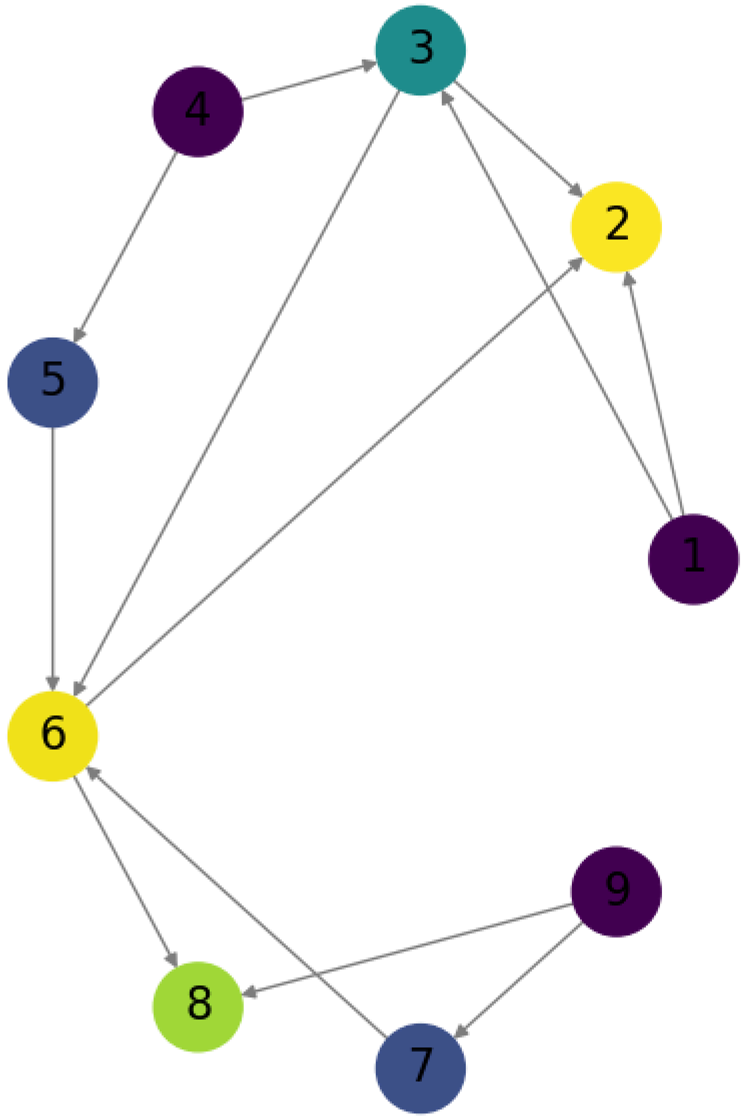
```
)
plt.show()
```




3
4
2
5
1
6
1
9
8
7
3
2
2
2
1
1
3
3
2
3
1
3

```
The adjancency matrix of G is
 [[0 1 2 0 0 0]
 [0 0 0 0 0 0]
 [0 2 0 0 0 1]
 [0 0 3 0 2 0]
 [0 0 0 0 0 3]
 [0 3 0 0 0 0]]
the sorted order is [1, 4, 9, 3, 5, 7, 6, 8, 2]
the lenght of sorted order is 9
 the total number of nodes is 9
```


3
4
2
5
1
6
9
8
7



the network size, diameter, and average shortest path length. the node's degree, or the statistical analysis of the nodes' degree. the network connectivity. the node's betweenness centrality and closeness centrality. the topological order of the nodes.

In [ ]:
```
assume distance between each nodes are same = 1
diam(G) = 3
dist(1,2) = 1
dist(1,3) = 1
dist(1,8) = 3
dist(3,2) = 1
dist(3,2) = 1
dist(3,8) = 2
dist(4,2) = 2
dist(4,8) = 3
dist(5,8) = 2
dist(6,8) = 1
dist(7,2) = 2
dist(9,2) = 3
the shortest path:(1,3,6,8), (4,5,6,8), (9,8,6,2)
weight of path(1,3,6,8) = 5,(4,5,6,8) = 7,(9,8,6,2) = 6
the cheapest path: (1,3,6,8) = 5
average shortest path length = 21/36 = 0.583
Win(1) = 0
Wout(1) = 3
Win(2) = 5
Wout(2) = 0
Win(3) = 5
Wout(3) = 3
Win(4) = 0
Wout(4) = 5
Win(5) = 2
Wout(5) = 3
Win(6) = 7
Wout(6) = 5
Win(7) = 3
Wout(7) = 3
Win(8) = 3
Wout(8) = 0
Win(9) = 0
Wout(9) = 4
This network is a weakly connected.
The normalized betweenness centrality is (1:0.0),(2:0.0),(3:0.0278),(4:0.0),(5:0.0278),(6:0.0833),(7:),(8:(
Closeness centrality: (1:1.6),(2:2.0),(3:2.0),(4:1.6),(5:4.0),(6:8.0),(7:4.0),(8:1.14),(9:2.67)
topologocal order:[1,4,9,3,5,7,8,2]
```