Lab Experiment No 2

Monesh S
1BM20AI039

NetworkX

Question:

Create a network of 20 nodes using networkX library. and add random edges between the nodes. make sure the graph is connected and visualize the graph using matplotlib. calculate degree centrality, betweenness centrality and eigen centrality of each node. visualize the degrees using an histogram.

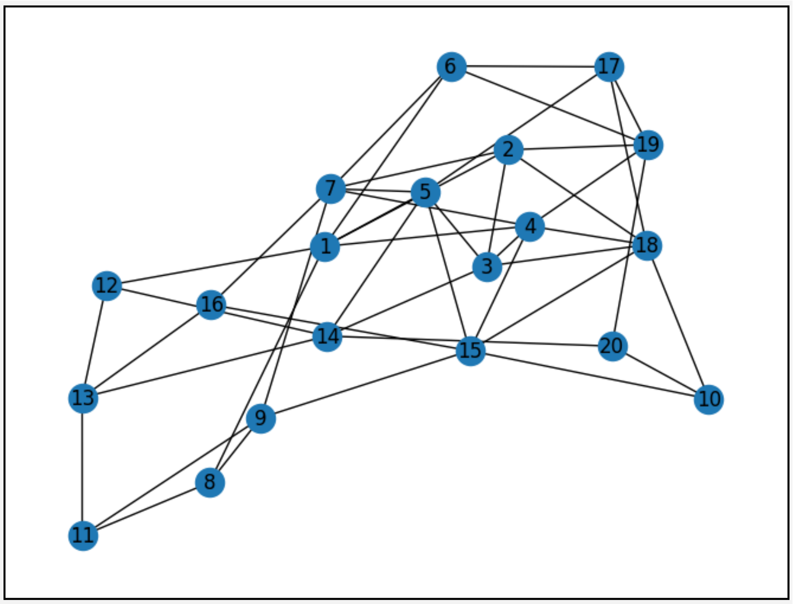
Code:

```
import networkx as nx
import matplotlib.pyplot as plt
import matplotlib.colors as mcolors
import collections
%matplotlib inline
G = nx.Graph()
G.add_edge(2, 3)
G.add_edge(17,19)
G.add_edge(17,6)
G.add_edge(1, 2)
G.add_edge(3, 4)
G.add_edge(1, 4)
G.add_edge(1, 5)
G.add_edge(3, 5)
G.add_edge(4, 7)
G.add_edge(14, 5)
G.add_edge(1,6)
G.add_edge(6, 7)
G.add_edge(10, 15)
G.add_edge(19, 20)
G.add_edge(18, 17)
G.add_edge(13, 14)
G.add_edge(7, 9)
G.add_edge(8, 11)
G.add_edge(12, 13)
G.add_edge(12, 14)
G.add_edge(15, 16)
G.add_edge(2, 19)
```

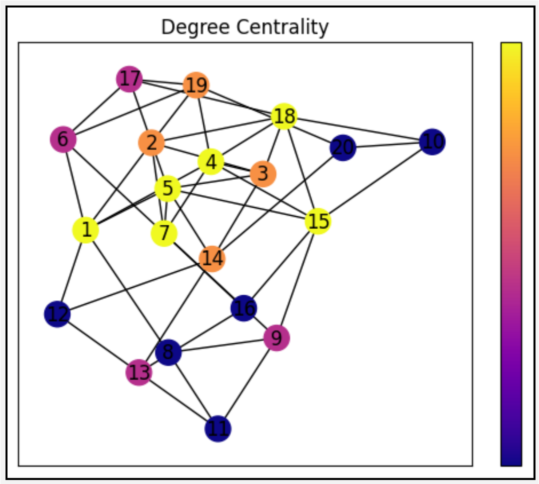
```
G.add_edge(4, 15)
G.add_edge(14, 5)
G.add_edge(7, 16)
G.add_edge(5, 17)
G.add_edge(4, 15)
G.add_edge(8, 9)
G.add_edge(14, 3)
G.add_edge(3, 18)
G.add_edge(1, 12)
G.add_edge(4, 19)
G.add_edge(10,20)
G.add_edge(18,10)
G.add_edge(15,18)
G.add_edge(11,13)
G.add_edge(1,8)
G.add_edge(14,20)
G.add_edge(16,13)
G.add_edge(9, 11)
G.add_edge(4, 18)
G.add_edge(2, 18)
G.add_edge(2, 7)
G.add_edge(5, 15)
G.add_edge(5, 7)
G.add_edge(7, 2)
G.add_edge(9, 15)
G.add_edge(6, 19)
nx.draw(G, with_labels = True)

def draw(G, pos, measures, measure_name):
    nodes = nx.draw_networkx_nodes(G, pos, node_size=250,
cmap=plt.cm.plasma,node_color=list(measures.values()),nodelist=measures.ke
ys())
    nodes.set_norm(mcolors.SymLogNorm(linthresh=0.01, linscale=1,
base=10))
    labels = nx.draw_networkx_labels(G, pos)
    edges = nx.draw_networkx_edges(G, pos)
    plt.title(measure_name)
    plt.colorbar(nodes)
    plt.axis()
    plt.show()
pos = nx.spring_layout(G, seed=675)
```

```
draw(G, pos, nx.degree_centrality(G), 'Degree Centrality')
draw(G, pos, nx.betweenness_centrality(G), 'Betweenness Centrality')
draw(G, pos, nx.eigenvector_centrality(G), 'Eigenvector Centrality')
degree_sequence = sorted([d for n, d in G.degree()], reverse=True)
degreeCount = collections.Counter(degree_sequence)
deg, cnt = zip(*degreeCount.items())
fig, ax = plt.subplots()
plt.bar(deg, cnt, width=0.80, color='b')
plt.title("Degree Histogram")
plt.ylabel("Count")
plt.xlabel("Degree")
ax.set_xticks([d + 0.4 for d in deg])
ax.set_xticklabels(deg)
plt.show()
```

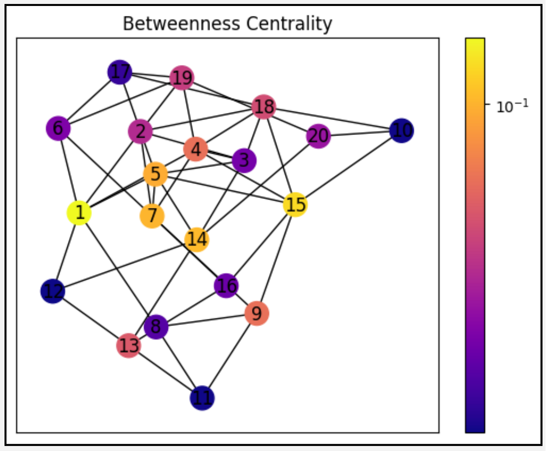
Screenshots and Output:



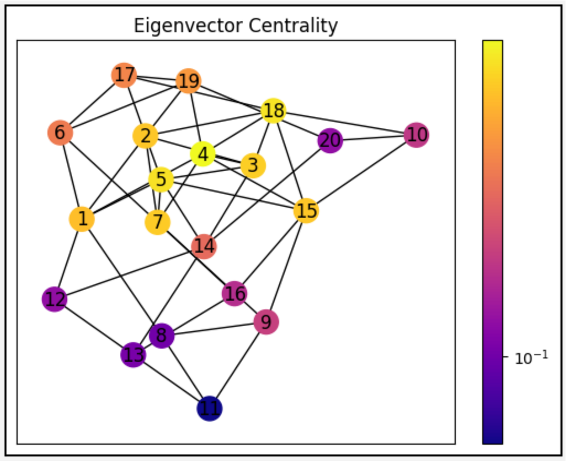
Graph



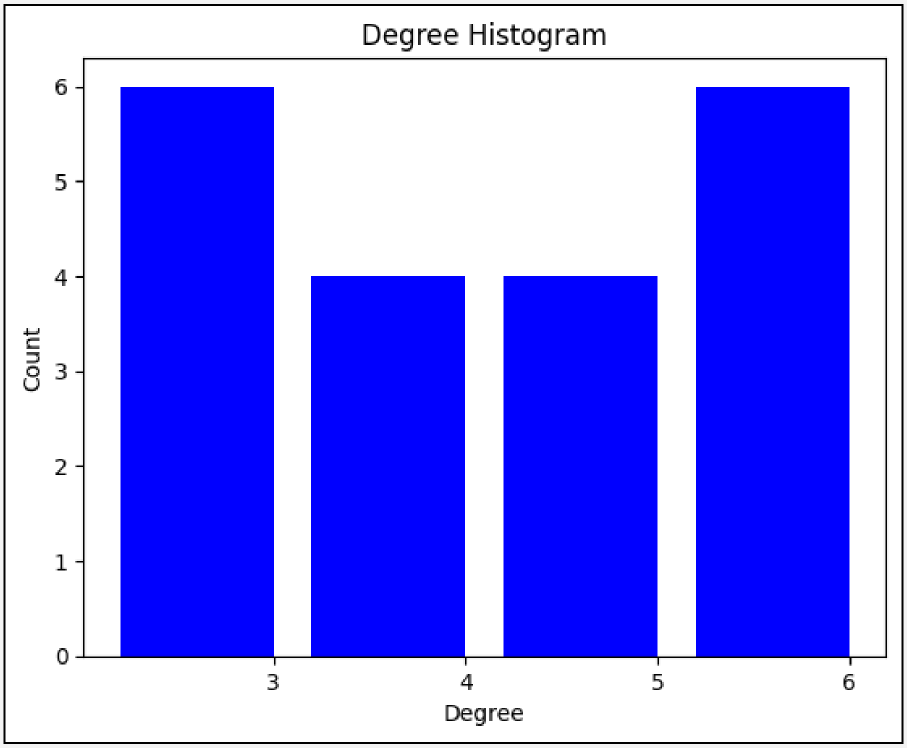
Degree Centrality



Betweenness Centrality



Eigenvector Centrality



Degree Histogram