Complex Network Analysis

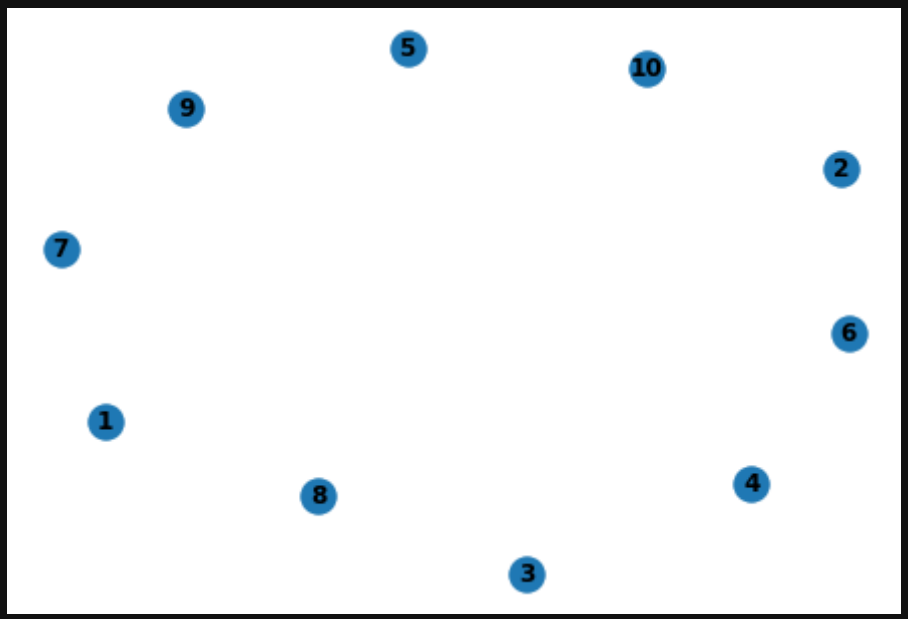
Assignment 1

1. Create a graph G with 10 nodes.

- a. An edge is to be created between nodes i and j if both are even numbered nodes. 2,4; 6,6; 2,8; 4,10; 4,6; etc.
- b. Count the number of self-loops in the graph
- c. Count the number of edges in the graph
- d. Print the adjacency list of the graph G
- e. Add weights to the edges in G. Weight of edge(i,j) is i+j

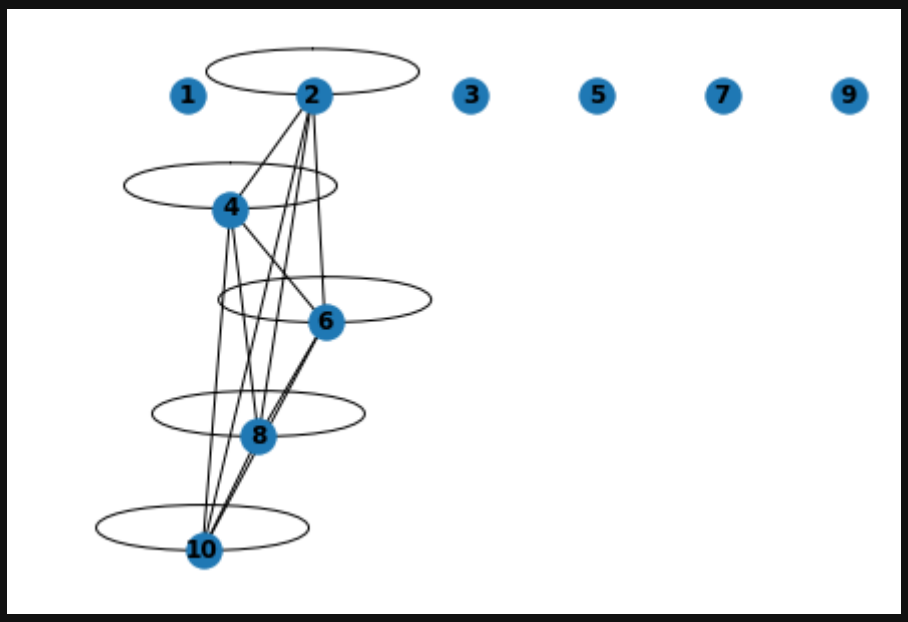
In [1]:
```
import networkx as nx
import matplotlib.pyplot as plt
```

In [2]:
```
G = nx.Graph()
N = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
G.add_nodes_from(N)
nx.draw(G, with_labels=True, font_weight='bold')
plt.show()
```



In [3]:
```
for i in N:
    if(i%2==0):
        for j in N:
            if(j%2==0):
                G.add_edge(i, j, minlen = 10)
```

In [4]:
```
pos = nx.nx_pydot.pydot_layout(G, prog='dot')
nx.draw(G, with_labels=True, pos = pos, font_weight='bold')
plt.show()
```



In [5]:
```
print("Number of Self-Loops = ", nx.number_of_selfloops(G))
```

```
Number of Self-Loops =  5
```

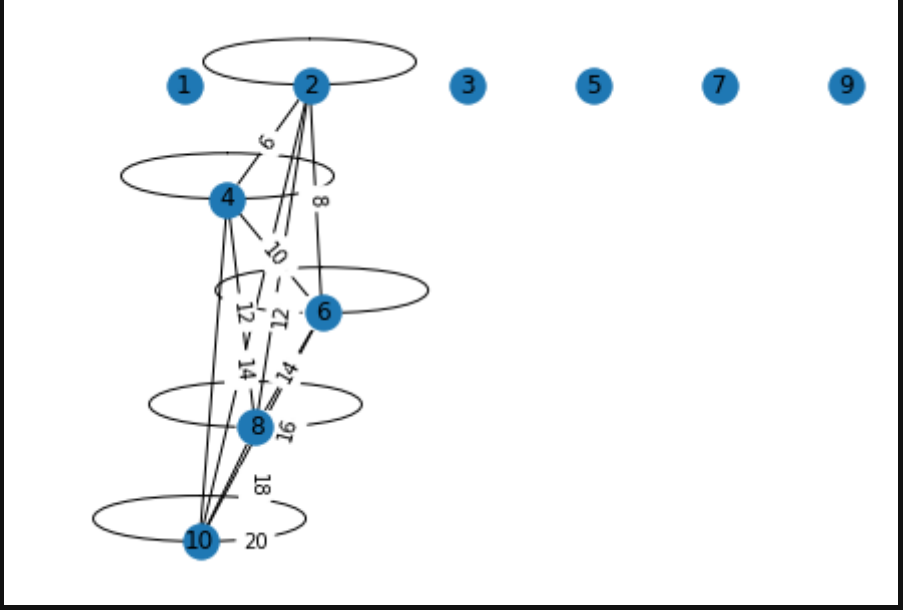
In [6]:
```
print("Number of Edges = ", G.number_of_edges())
```

```
Number of Edges =  15
```

In [7]:
```
print("Adjacency List of the Graph G = ")
for line in nx.generate_adjlist(G):
    print(line)
```

```
Adjacency List of the Graph G =
1
2 2 4 6 8 10
3
4 4 6 8 10
5
6 6 8 10
7
8 8 10
9
10 10
```

In [8]:
```
for i,j in G.edges():
    G[i][j]['weight'] = i+j
nx.draw(G, with_labels=True, pos=pos)
labels = nx.get_edge_attributes(G, 'weight')
pos = nx.nx_pydot.pydot_layout(G, prog='dot')
nx.draw_networkx_edge_labels(G, pos=pos, edge_labels=labels)
plt.show()
```



2.

- a. Create following graph
- b. Add attributes to each node that is name of the people. The edge represents their friendship
- c. For a given 2 nodes I and J in the above graph, check if I and J are present in the graph. If present, then check if I is adjacent to J.

Input : Piper and William

Output : yes

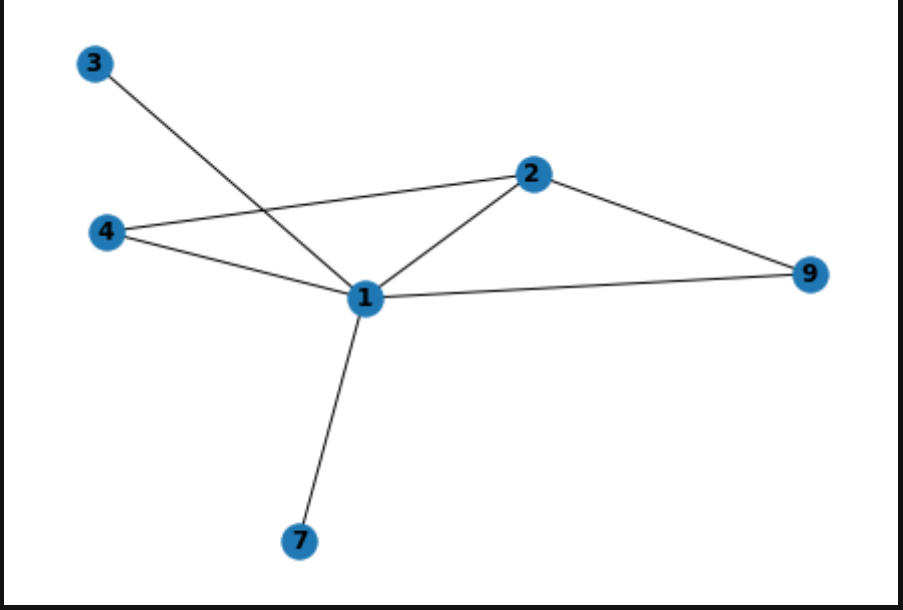
Input : robin and William

Output : No

- d. Find the node/nodes with maximum number of edges
- e. Add the attributes to each edge
- f. Remove node 9 and corresponding edges

In [9]:

```
H = nx.Graph()
H.add_node(1, name = "Manav")
H.add_node(2, name = "Karan")
H.add_node(3, name = "Suparna")
H.add_node(4, name = "Trupti")
H.add_node(7, name = "Urvashi")
H.add_node(9, name = "Vidhi")
H.add_edges_from([(1,2),(1,3),(1,4),(1,7),(1,9),(2,4),(2,9)])
node_labels = nx.get_node_attributes(H,'name')
#pos = nx.nx_pydot.pydot_layout(H, prog='dot')
nx.draw(H, with_labels=True, font_weight='bold')
plt.show()
```



In [10]:

```
for i in H.nodes():
    for j in H.nodes():
        print("Input: ", H.nodes[i]['name'], " and ", H.nodes[j]['name'])
        if(H.has_edge(i,j)):
            print("Output: ", 'yes')
        else:
            print("Output: ", 'no')
```

```
Input:  Manav  and  Manav
Output:  no
Input:  Manav  and  Karan
Output:  yes
Input:  Manav  and  Suparna
Output:  yes
Input:  Manav  and  Trupti
Output:  yes
Input:  Manav  and  Urvashi
Output:  yes
Input:  Manav  and  Vidhi
Output:  yes
Input:  Karan  and  Manav
Output:  yes
Input:  Karan  and  Karan
Output:  no
Input:  Karan  and  Suparna
Output:  no
Input:  Karan  and  Trupti
Output:  yes
Input:  Karan  and  Urvashi
Output:  no
Input:  Karan  and  Vidhi
Output:  yes
Input:  Suparna  and  Manav
Output:  yes
Input:  Suparna  and  Karan
Output:  no
Input:  Suparna  and  Suparna
Output:  no
Input:  Suparna  and  Trupti
Output:  no
Input:  Suparna  and  Urvashi
Output:  no
Input:  Suparna  and  Vidhi
Output:  no
Input:  Trupti  and  Manav
Output:  yes
Input:  Trupti  and  Karan
```

```
Output:  yes
Input:  Trupti  and  Suparna
Output:  no
Input:  Trupti  and  Trupti
Output:  no
Input:  Trupti  and  Urvashi
Output:  no
Input:  Trupti  and  Vidhi
Output:  no
Input:  Urvashi  and  Manav
Output:  yes
Input:  Urvashi  and  Karan
Output:  no
Input:  Urvashi  and  Suparna
Output:  no
Input:  Urvashi  and  Trupti
Output:  no
Input:  Urvashi  and  Urvashi
Output:  no
Input:  Urvashi  and  Vidhi
Output:  no
Input:  Vidhi  and  Manav
Output:  yes
Input:  Vidhi  and  Karan
Output:  yes
Input:  Vidhi  and  Suparna
Output:  no
Input:  Vidhi  and  Trupti
Output:  no
Input:  Vidhi  and  Urvashi
Output:  no
Input:  Vidhi  and  Vidhi
```

In [11]:

```
dict = {}
for i in H.nodes():
    dict[i] = len(H.edges(i))
print(dict)
m = max(dict.items(), key = lambda k : k[1])
print("Node", m[0], "has maximum number of Nodes: ", m[1])
```

```
{1: 5, 2: 3, 3: 1, 4: 2, 7: 1, 9: 2}
Node 1 has maximum number of Nodes:  5
```

In [12]:

```
import numpy as np
```

In [13]:

```
new_attr = {}
for i,j in H.edges():
    new_attr[(i,j)] = {"age": np.random.randint(100)}
new_attr
```

Out[13]:

```
{(1, 2): {'age': 40},
 (1, 3): {'age': 52},
 (1, 4): {'age': 48},
 (1, 7): {'age': 61},
 (1, 9): {'age': 95},
 (2, 4): {'age': 74},
 (2, 9): {'age': 36}}
```

In [14]:

```
nx.set_edge_attributes(H, new_attr)
```

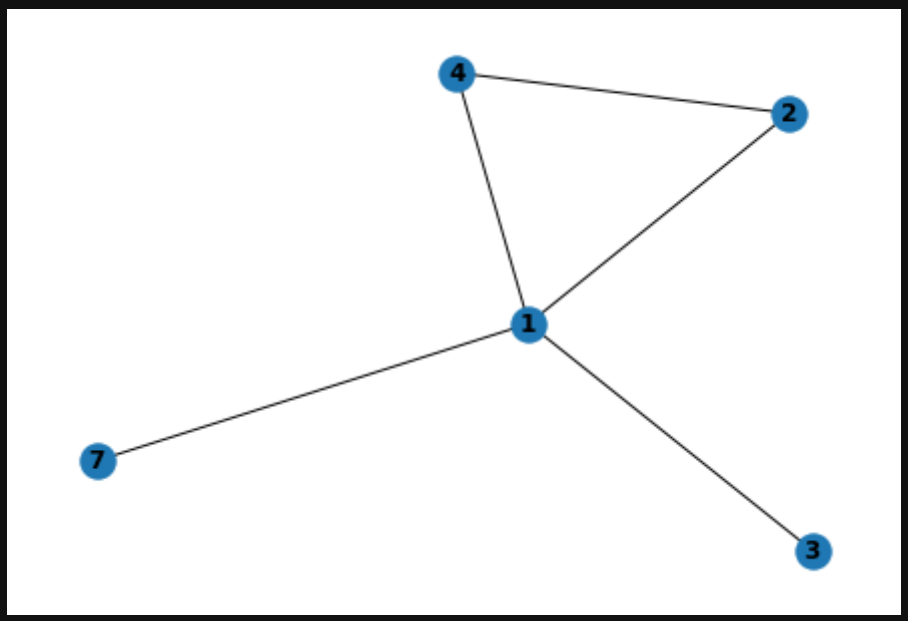
In [17]:
```
for i,j in H.edges():
    print(i, "-", j, H[i][j])
```

```
1 - 2 {'age': 40}
1 - 3 {'age': 52}
1 - 4 {'age': 48}
1 - 7 {'age': 61}
1 - 9 {'age': 95}
2 - 4 {'age': 74}
2 - 9 {'age': 36}
```

In [18]:
```
H.remove_node(9)
list(H.edges)
```

Out[18]:
```
[(1, 2), (1, 3), (1, 4), (1, 7), (2, 4)]
```

In [19]:
```
nx.draw(H, with_labels=True, font_weight='bold')
plt.show()
```

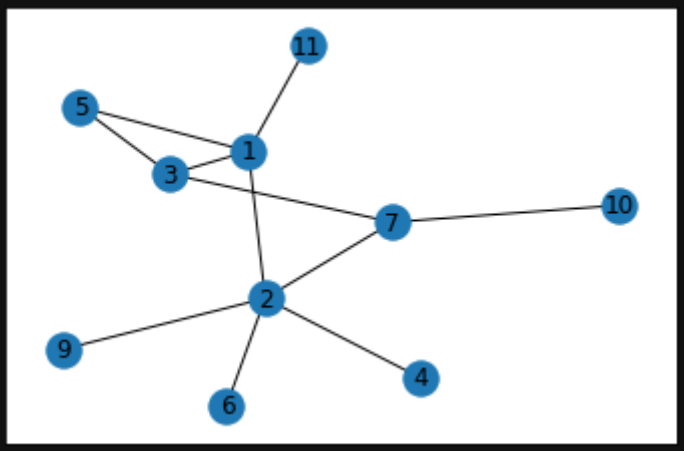


3. Create one adjacency list file. Based on this file, create a graph. Find the nodes with maximum number of edges. Color that node red. Remaining nodes can be colored blue.

In [25]:
```
adj_list = open("graph_adjlst_a1.txt","r")
s = adj_list.read()
print(s)
myG = nx.read_adjlist("graph_adjlst_a1.txt","r")
print(myG)
K = nx.Graph(myG)
print(len(K))
nx.draw_networkx(K)
plt.show()
```

```
1 2 3 5 11
2 6 9 4
3 7 5
7 10 2
```

```
Graph with 10 nodes and 11 edges
```



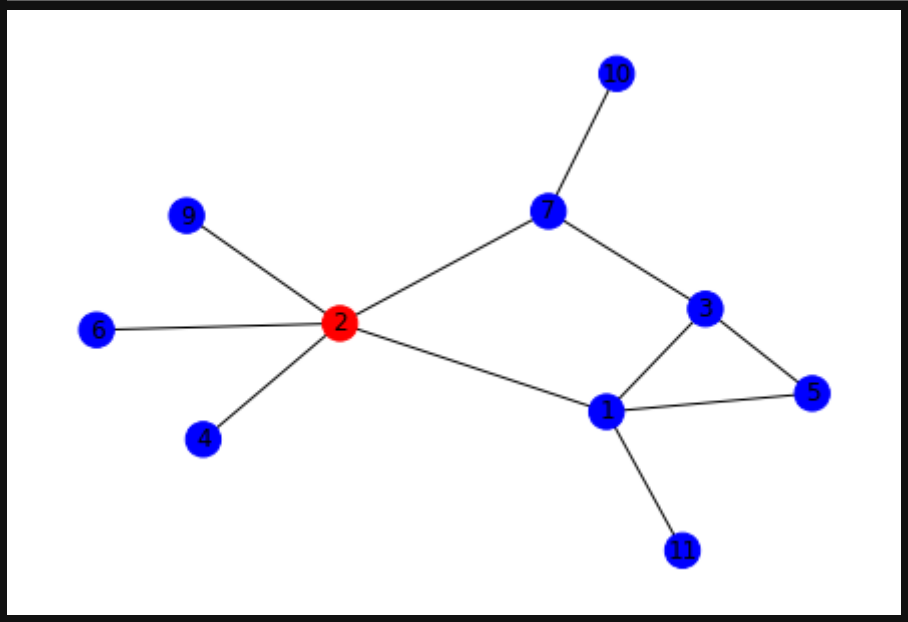
In [26]:

```
dict = {}
for i in K.nodes():
    dict[i] = len(K.edges(i))
print(dict)
m = max(dict.items(), key = lambda k : k[1])
print("Node", m[0], "has maximum number of Nodes: ", m[1])
```

```
{'1': 4, '2': 5, '3': 3, '5': 2, '11': 1, '6': 1, '9': 1, '4': 1, '7': 3, '10': 1}
Node 2 has maximum number of Nodes:  5
```

In [27]:

```
color_map = ['red' if node == m[0] else 'blue' for node in K]
nx.draw(K, node_color=color_map, with_labels=True) # node lables
# nx.draw(G,with_labels=True)
plt.show()
```



4. Create a graph from a numpy matrix with 10 nodes and edges created randomly

In [28]:

```
G_mtrx =np.random.randint(0,2,size=(10,10))
print(G_mtrx)
L=nx.Graph(G_mtrx)
```

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

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

In [29]:

```
nx.draw_networkx(L)
plt.show()
```