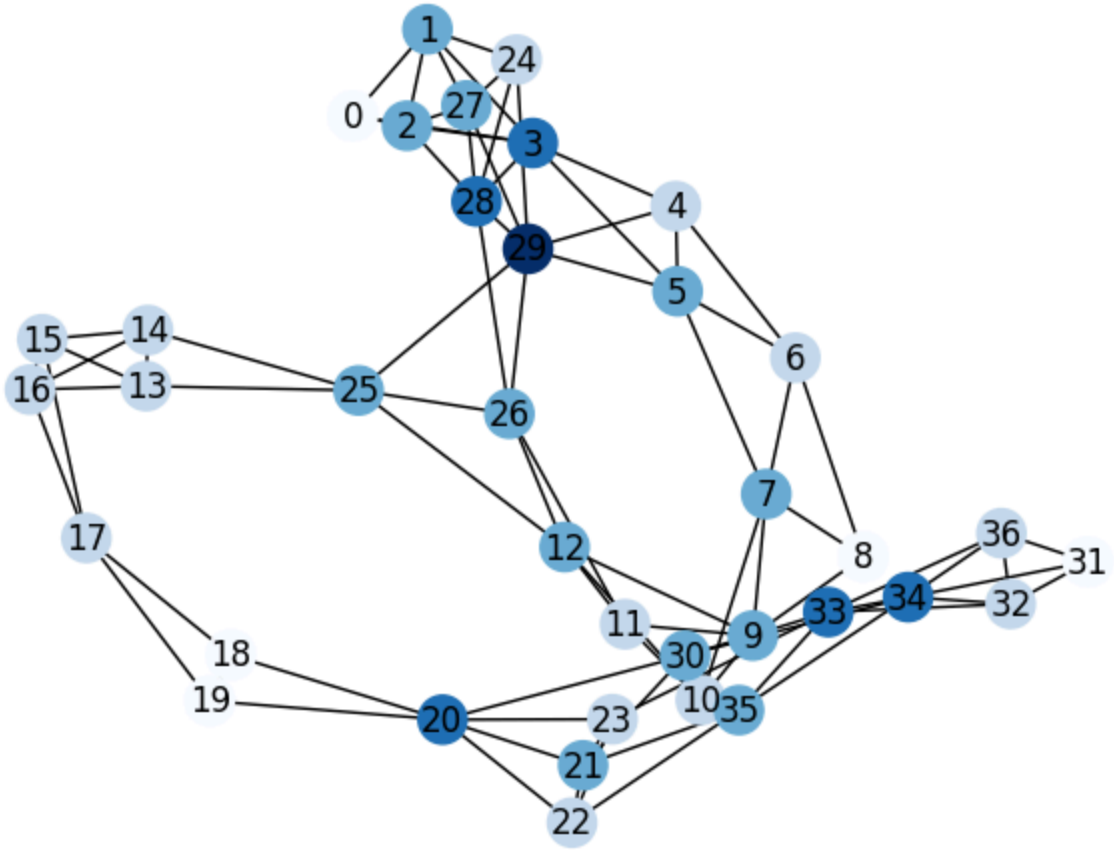
In [1]:
```
import torch
from torch_geometric.datasets import TUDataset
from torch_geometric.utils.convert import to_networkx
import networkx as nx
```

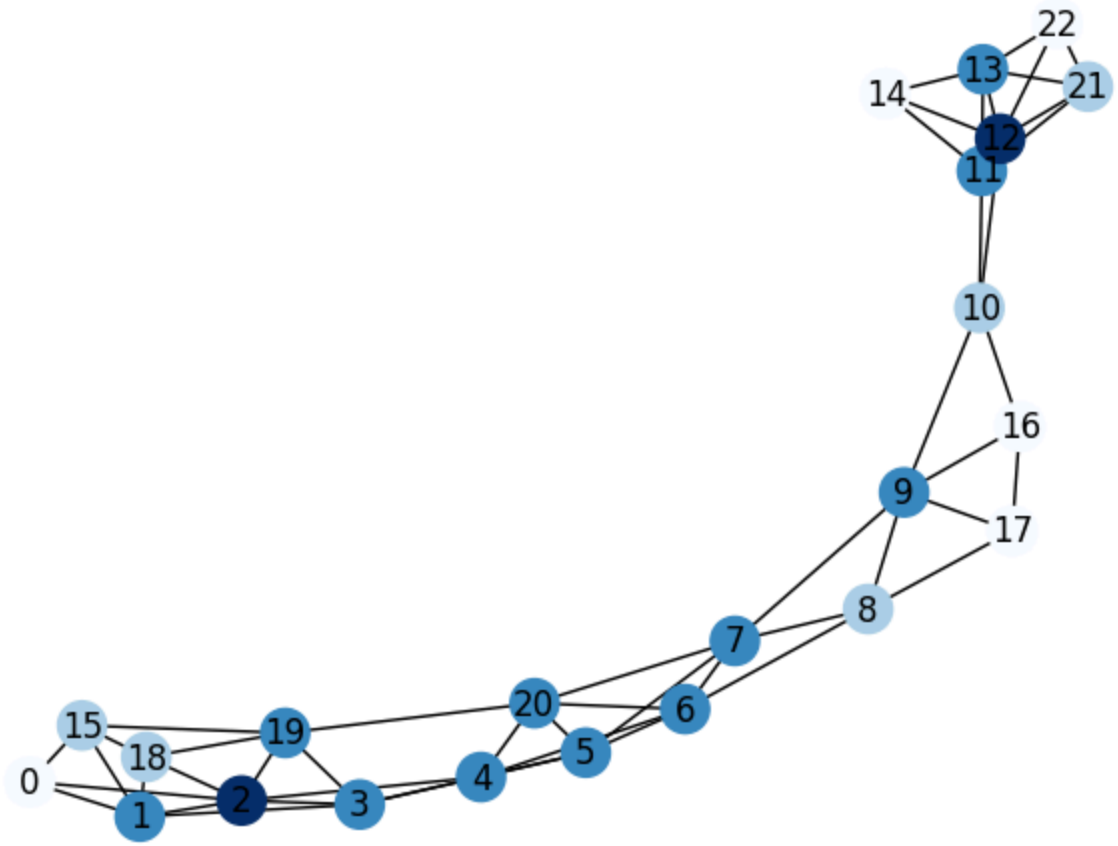
In [2]:
```
dataset = TUDataset(root='/tmp/ENZYMES', name='ENZYMES')
```

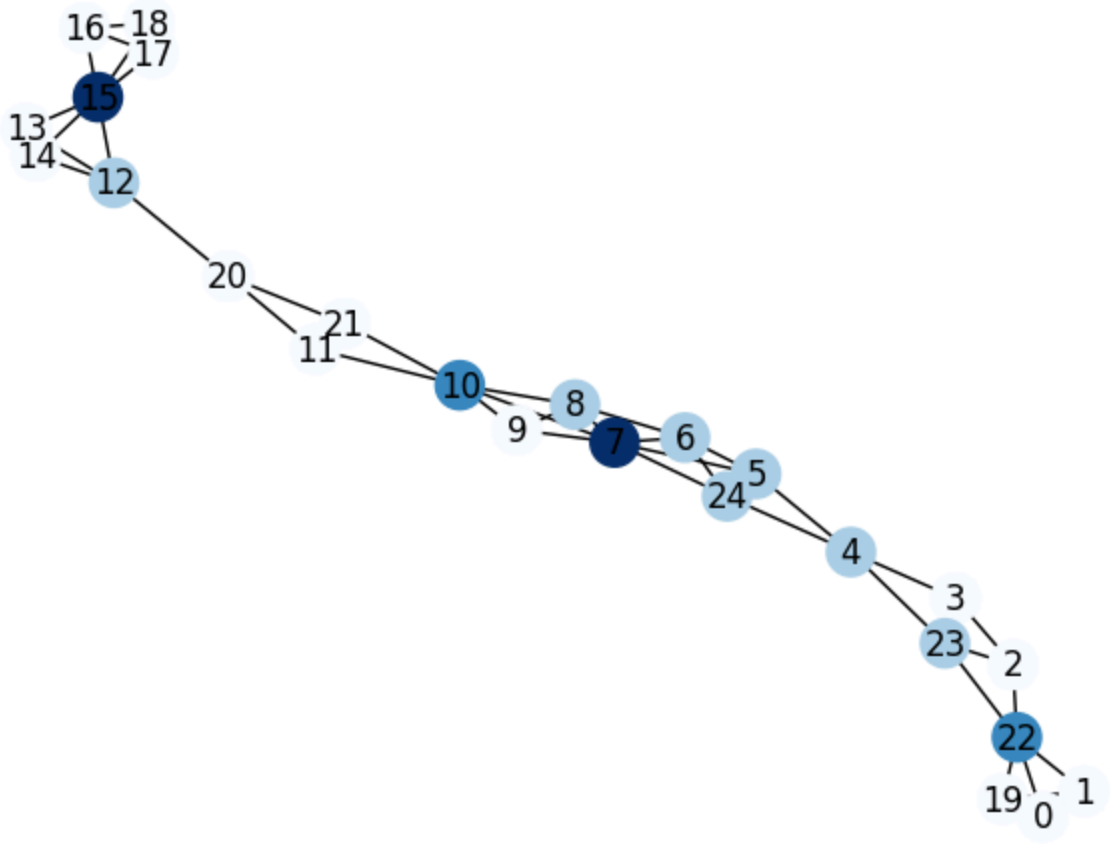
In [3]:
```
import matplotlib.pyplot as plt

def plot_graph_with_degree_color(G):
    degrees = dict(G.degree())
    node_colors = [degrees[node] for node in G.nodes()]
    pos = nx.spring_layout(G)
    G = G.to_undirected()
    nx.draw(G, pos, node_color=node_colors, cmap=plt.cm.Blues, with_labels=True)
    plt.show()

for data in dataset[0:3]:
    G = to_networkx(data)
    plot_graph_with_degree_color(G)
```


1
24
0
2
27
3
28
4
29
5
15
14
6
16
13
25
26
7
17
12
8
36
31
34
32
33
11
9
18
30
19
10
35
20
23
21
22


22
13
21
14
12
11
10
16
9
17
8
7
20
6
15
19
5
18
4
0
2
3
1



In [4]:
```
train_dataset = dataset[:540]
test_dataset = dataset[540:]
```

In [5]:
```
avg_degree = sum(dict(G.degree()).values()) / len(G.nodes())
avg_degree
```

Out[5]: 7.36

In [6]:
```
def calculate_graph_metrics(G):
    avg_degree = sum(dict(G.degree()).values()) / len(G.nodes())
    try:
```

```
        diameter = nx.diameter(G)
    except nx.NetworkXError:
        diameter = float('inf')
    avg_path_length = nx.average_shortest_path_length(G) if nx.is_connected(G) else None
    clustering_coeff = nx.average_clustering(G)

    return {
        'avg_degree': avg_degree,
        'diameter': diameter,
        'avg_path_length': avg_path_length,
        'clustering_coeff': clustering_coeff
    }

for data in dataset[0:3]:
    G = to_networkx(data).to_undirected()
    metrics = calculate_graph_metrics(G)
    print(f"Metrics for graph:\n{metrics}")
```

```
Metrics for graph:
{'avg_degree': 4.54054054054054, 'diameter': 12, 'avg_path_length': 4.978978978978979, 'clustering_coeff':
0.5653796653796653}
Metrics for graph:
{'avg_degree': 4.434782608695652, 'diameter': 8, 'avg_path_length': 3.41897233201581, 'clustering_coeff':
0.5811594202898551}
Metrics for graph:
{'avg_degree': 3.68, 'diameter': 11, 'avg_path_length': 4.626666666666667, 'clustering_coeff': 0.7066666666
666667}
```

Node Classification

Node classification is done by given a graph, predict what specific nodes of the graphs are. For example, in a social network, people being the nodes and edges being connections, classifying a node could be checking if a person is politically left or right leaning.

Link Prediction

Link Prediction is a way to predict edges of a graph. For example, main recommender systems are used to predict new friendships or collaborations based on existing connections.