H2

Exercise 1.

▶ see problem

A simple cycle does not have repeating vertecies (aside from the start/finish).

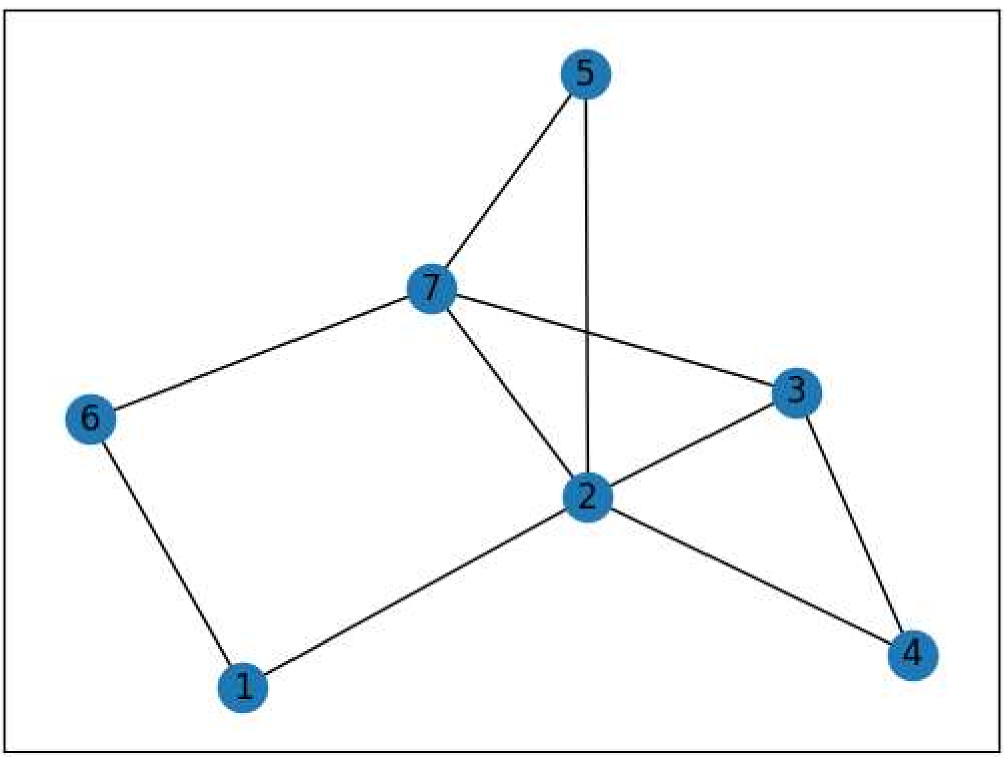
The cycle 243761 is the longest, computed numerically using the networkx package below

In [20]:
```python
import networkx as nx
E = {(1,2),(1,6),(2,5),(2,7),(2,3),(2,4),(4,3),(3,7),(5,7),(6,7)}
G = nx.Graph(E)
nx.draw_networkx(G, with_labels = True)

print([cycle for cycle in nx.simple_cycles(G)])

a = sorted(list(nx.simple_cycles(G)), key = lambda s: -len(s))
print(a[0])
```

```
[[2, 4, 3], [2, 4, 3, 7], [2, 4, 3, 7, 5], [2, 4, 3, 7, 6, 1], [2, 1, 6, 7], [2, 1, 6, 7, 3], [2, 1, 6, 7, 5], [2, 7, 3], [2, 7, 5], [2, 3, 7, 5]]
[2, 4, 3, 7, 6, 1]
```



This result makes logical sense. To make a cycle that includes all points, either the node 7 or 2 must be hit twice since 61, 34 and 5 are all connected only by 7 and 2.

Exercise 2.

▶ see problem

A quick test is to check if the Kraft-McMillan number is less than 1.

$$K = 0 + 0 + \frac{1}{9} + \frac{4}{27} = \frac{7}{27} = 0.259$$

Since $K < 1$, the PF code can be extended without losing the PF property.

An even quicker test would be to give an example of such a code. 111 can be added since 11 and 1 are not codewords

Exercise 3.

▶ see problem

Using length one, only $2^1 = 2$ values can be used, but $2^3 = 8$ can be represented with length 3 codewords.

One arrangement is

$$C = (000, 001, 010, 011, 100, 101)$$

The sum of length is $6 * 3 = 18$, can this be less? Let's look at the tree.

In [65]:
```
import networkx as nx
from other_code import make_edges_from_code, hierarchy_pos
```

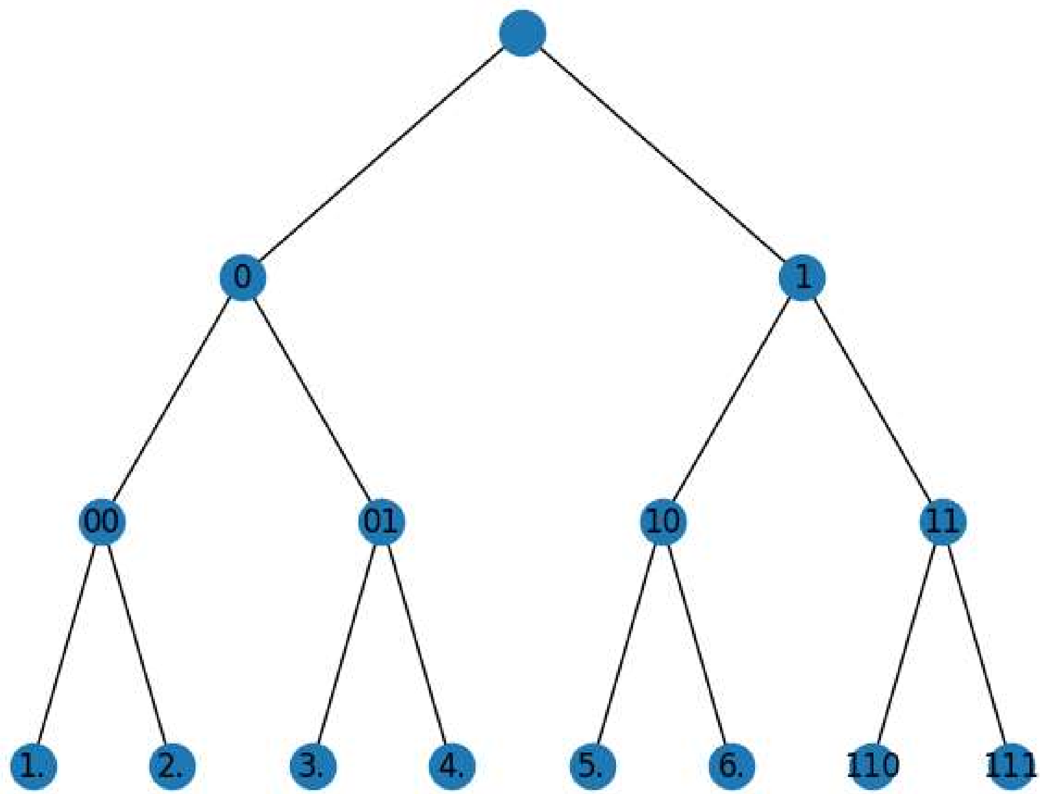
In [62]:
```
num_layers = 3

code = {
    '000':'1.',
    '001':'2.',
    '010':'3.',
    '011':'4.',
    '100':'5.',
    '101':'6.'
    }

connections, layers = make_edges_from_code(code, num_layers=3)
```

In [63]:
```
G=nx.Graph()
G.add_edges_from(connections)
pos = hierarchy_pos(G,'')

nx.draw(G, pos=pos, with_labels=True)
```



We can map 5 and 6 to the codewords 10 and 11 to make the result shorter

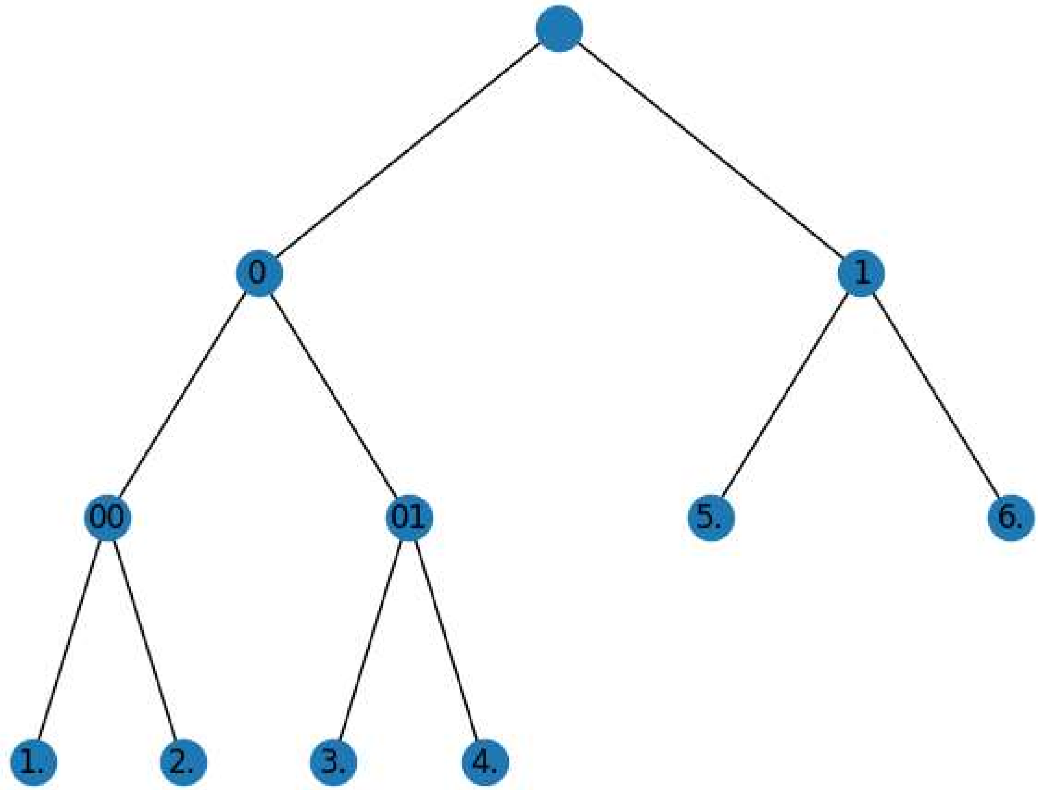
In [66]:
```
num_layers = 3

code = {
    '000':'1.',
    '001':'2.',
    '010':'3.',
    '011':'4.',
    '10':'5.',
    '11':'6.'
    }

connections, layers = make_edges_from_code(code, num_layers=3)

G=nx.Graph()
G.add_edges_from(connections)
pos = hierarchy_pos(G,'')

nx.draw(G, pos=pos, with_labels=True)
```



This is prefix free and has length

$$2 * 2 + 4 * 3 = 16$$

Exercise 4.

▶ see problem

Using theorem 2.3.5, we need only calculate the Kraft-Mcmillan Number

$$K_a = 0 + \frac{1}{3} + \frac{3}{9} + \frac{10}{27} = 28/27 > 1$$
$$K_b = 0 + 0 + \frac{1}{9} + \frac{3}{27} + \frac{39}{81} = \frac{57}{81} < 1$$

Thus parameter a does not have a prefix-free code, while parameter b does

Exercise 5.

▶ see problem

b-nary code: $|T| = b$
alphabet of size m: $|S^n| = m$

Since $C : S^n \mapsto T^*$ exists and $K = 1$,

$$K = \sum_{i}^{M} \frac{n_i}{b^i} = 1$$

Where M is the longest codeword in C, which exists since the alphabet being encoded has a finite size and C is 1-to-1. Then

$$m = \sum_{i}^{M} n_i$$

and

$$b^M \sum_{i}^{M} \frac{n_i}{b^i} = b^M$$
$$\sum_{i}^{M} n_i b^{M-i} = b^M$$

Since $b = 1 \pmod{b-1}$ and $b^k = 1 \pmod{b-1}$ for positive integers of k, this equates to

$$\sum_{i}^{M} n_i \pmod{b-1} = 1 \pmod{b-1}$$
$$m \pmod{b-1} = 1 \pmod{b-1}$$
$$m - 1 = 0 \pmod{b-1}$$

Thus $m-1$ divides by $b-1$ with a remainder of zero

Exercise 6.

▶ see problem

6. The parameter for c is

$$(0, 0, 1, 2, 4)$$

(a)

$$Q_1(x) = x^2 + 2x^3 + 4x^4$$
$$Q_2(x) = x^4 + 4x^5 + 12x^6 + 16x^7 + 16x^8$$
$$Q_3(x) = x^6 + 6x^7 + 24x^8 + 56x^9 + 96x^10 + 96x^11 + 64x^12$$

(b) The coefficients of x^7 represent how many codewords of CC and CCC (aka C^3) have length 7.

The list of messages in S^* that map to codewords of length 16 and 6 are:

In [72]:

```python
code = {
    'a':'00',
    'b':'010',
    'c':'011',
    'd':'1000',
    'e':'1001',
    'f':'1101',
    'g':'1111'
}

q2 = set()
q3 = set()

for c1 in code.keys():
    for c2 in code.keys():
        if len(code[c1]+code[c2]) == 7:
            q2.add(c1+c2)

        for c3 in code.keys():
            if len(code[c1]+code[c2]+code[c3]) == 7:
                q3.add(c1+c2+c3)

print(q2)
print(q3)
```

```
{'gb', 'eb', 'db', 'gc', 'ec', 'fc', 'bd', 'be', 'fb', 'bg', 'cg', 'bf', 'ce', 'cf',
'cd', 'dc'}
{'aca', 'aab', 'aac', 'aba', 'caa', 'baa'}
```