For cyclic circuits to work:

- get connection dictionary ✓ working
- convertToGraph working ✓
- propery destroy class object and clear circComplist() ✓

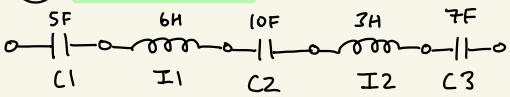
- For multiple cyclic circuits to work:
  - remove (move) duplicates in connection dictionary
  - correct graph conversion

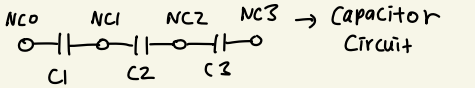
- JS conversion:
  - Connection dictionary ✓ working
  - convertToGraph → first node not added, other nodes are added

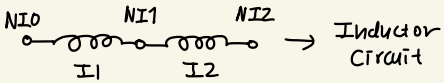
Buggy Function:

Test Cases

① Linear Circuit







- Capacitance Connection Dictionary:

  {'C1_1': [], 'C1_2': ['C2_1'], 'C2_2': ['C3_1'], 'C3_2': []}

  ('C1_1': [] — end; 'C3_2': [] — end)

- Capacitance Graph:

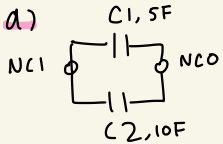
  {'NC1': {'NC0': '5F', 'NC2': '10F'}, 'NC2': {'NC3': '7F'}} → ✗ duplicate connections!

- Inductance Graph:

  {'NI1': {'NI0': '6H', 'NI2': '3H'}} ✓ working!

② Cyclic Circuit

a)



- Capacitance Connection Dictionary: {'C1_1': ['C2_2'], 'C1_2': ['C2_1']} (adjacency dictionary) → would this form work too?

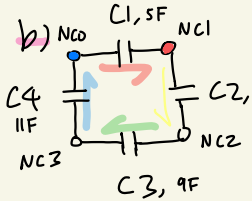
  # of elems = 2 = # of nodes

- Capacitance Graph: {'NC0': {'NC1': '10F'}, 'NC1': {'NC0': '5F'}}

• Capacitance Connection Dictionary:

{'C1_1': ['C4_2'], 'C1_2': ['C2_1'], 'C2_2': ['C3_1'], 'C3_2': ['C4_1']}

b)



• Capacitance Graph: {'NC0': {'NC3': '11F', 'NC1': '5F'},

'NC1': {'NC2': '7F'}, 'NC2': {'NC3': '9F'}}

Algorithm:

① Loop through terminal in conDict:

'C1_1' → 'C4_2'

'C4_2' → 'C4_1'

if 'C1_1': [] false

① Check if cyclic:

'C1_1' → search for 'C1_2' in conDict

↓

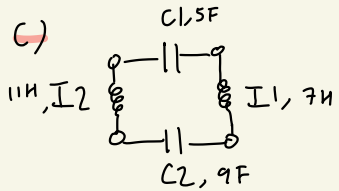
– search for opposing elems

e.g. if C1_2 is key, loop through vals if C1_2 is a val, get key

– if opp. elem == 'C1_1' → true else:

– search if opp. elem is in conDict ....

c)



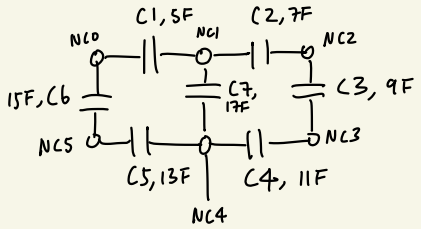
• Capacitance Graph:

{'NC0': {'NC1': '9F'}, 'NC1': {'NC0': '5F'}}

• Inductance Graph:

{'NI0': {'NI1': '11H'}, 'NI1': {'NI0': '7H'}}

(3) Circuit with multiple cycles



- Capacitance Connection Dictionary:

```
{'C1_1': ['C6_2'], 'C1_2': ['C2_1', 'C7_1'], 'C2_2': ['C3_1'],
 'C3_2': ['C4_1'], 'C4_2': ['C5_1', 'C7_2'],
 'C5_2': ['C6_1']}
```