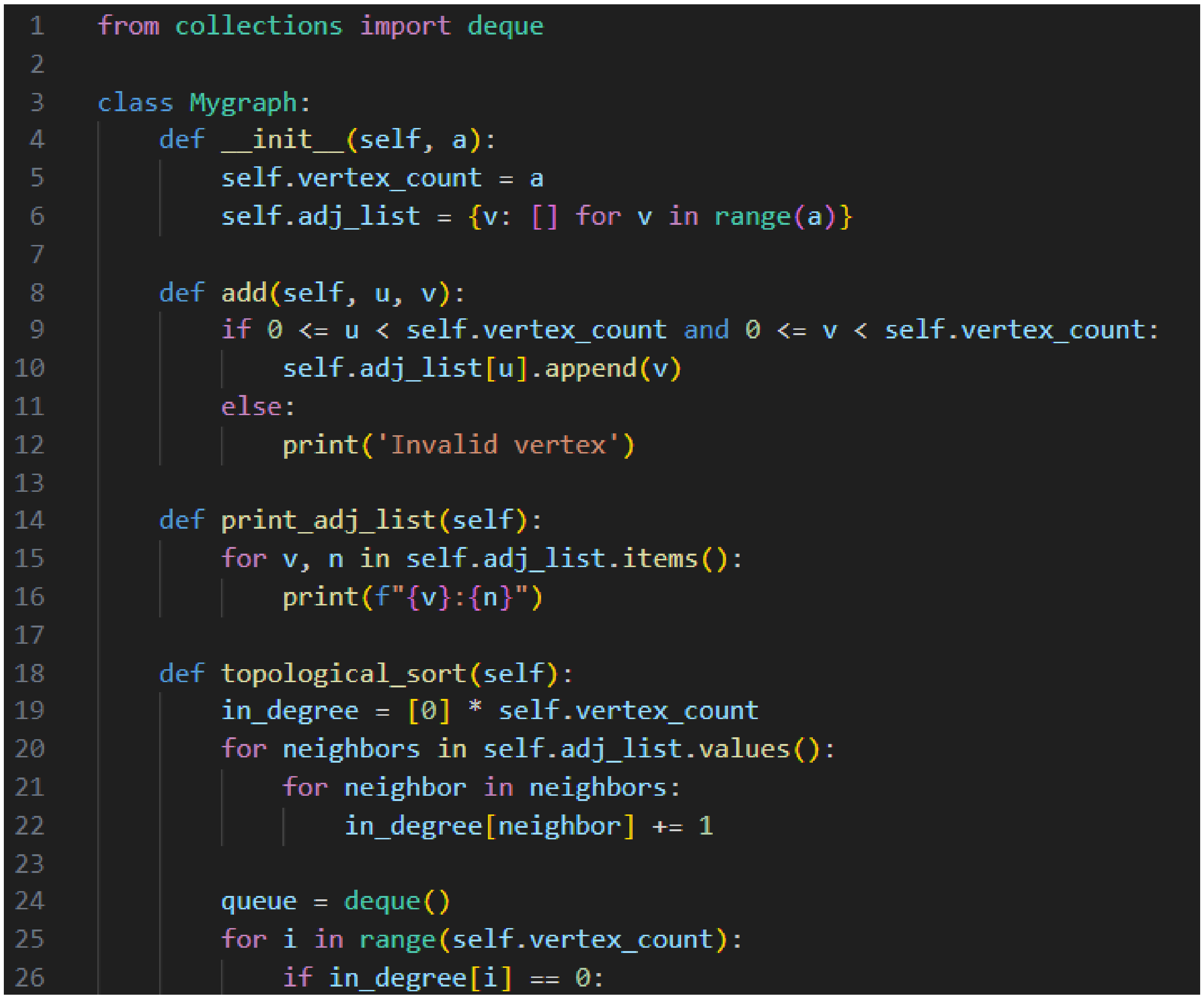
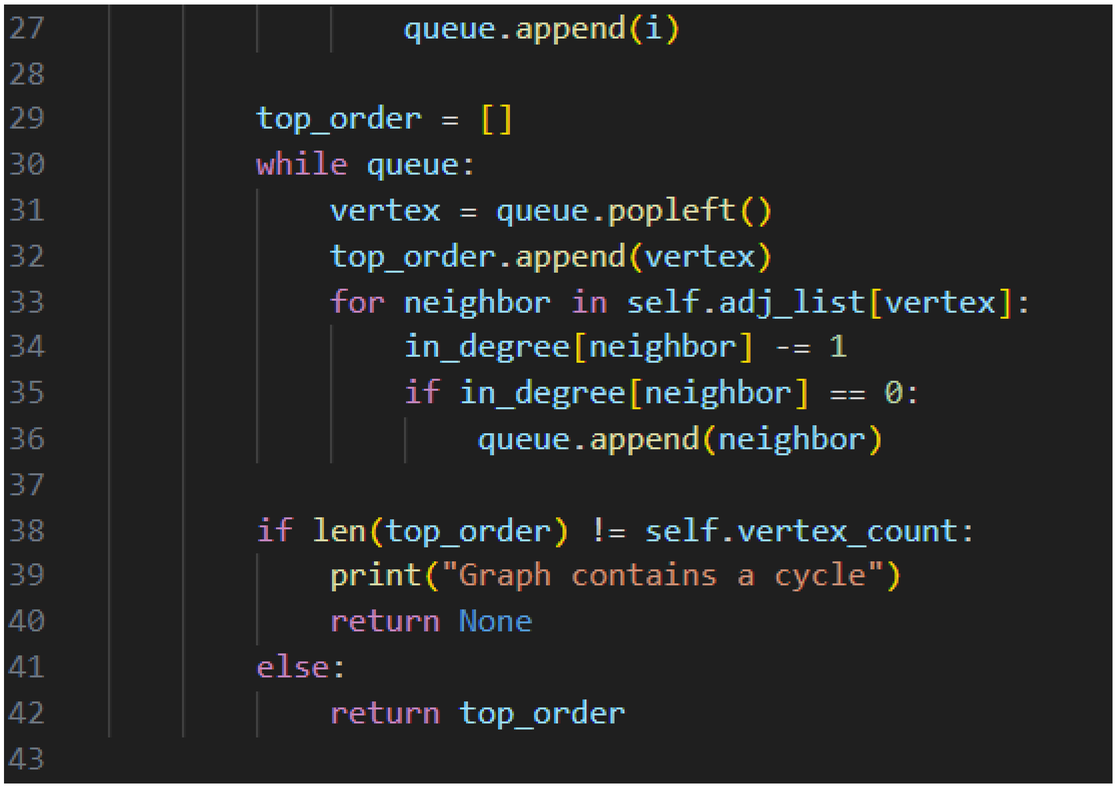
LAB#17

Example#1: Write a program to perform a topological sort on a Directed Acyclic Graph (DAG) using Kahn's algorithm.

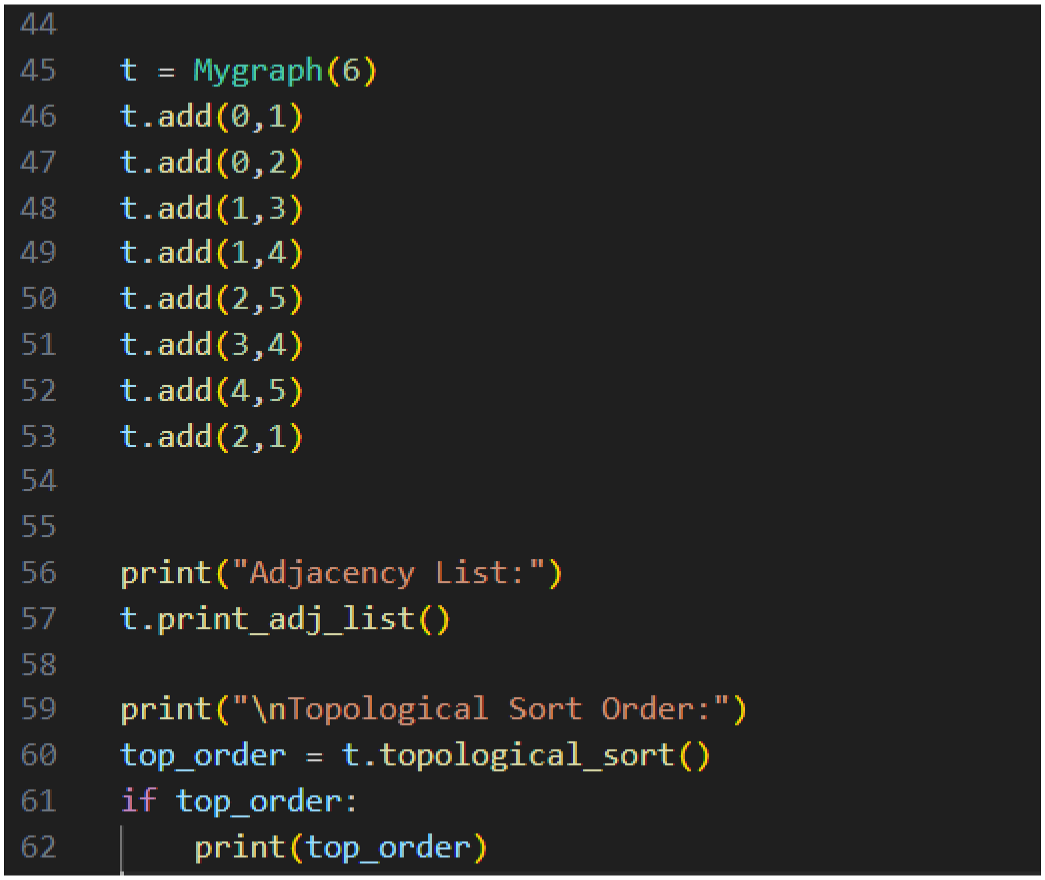
Solution:



```python
from collections import deque

class Mygraph:
    def __init__(self, a):
        self.vertex_count = a
        self.adj_list = {v: [] for v in range(a)}

    def add(self, u, v):
        if 0 <= u < self.vertex_count and 0 <= v < self.vertex_count:
            self.adj_list[u].append(v)
        else:
            print('Invalid vertex')

    def print_adj_list(self):
        for v, n in self.adj_list.items():
            print(f"{v}:{n}")

    def topological_sort(self):
        in_degree = [0] * self.vertex_count
        for neighbors in self.adj_list.values():
            for neighbor in neighbors:
                in_degree[neighbor] += 1

        queue = deque()
        for i in range(self.vertex_count):
            if in_degree[i] == 0:
```

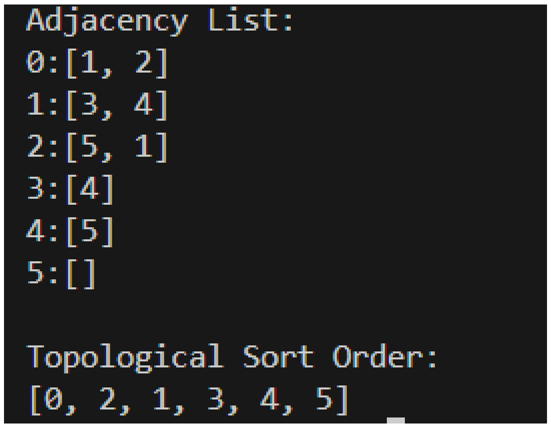


```python
                    queue.append(i)

            top_order = []
            while queue:
                vertex = queue.popleft()
                top_order.append(vertex)
                for neighbor in self.adj_list[vertex]:
                    in_degree[neighbor] -= 1
                    if in_degree[neighbor] == 0:
                        queue.append(neighbor)

            if len(top_order) != self.vertex_count:
                print("Graph contains a cycle")
                return None
            else:
                return top_order

```



```python

t = Mygraph(6)
t.add(0,1)
t.add(0,2)
t.add(1,3)
t.add(1,4)
t.add(2,5)
t.add(3,4)
t.add(4,5)
t.add(2,1)


print("Adjacency List:")
t.print_adj_list()

print("\nTopological Sort Order:")
top_order = t.topological_sort()
if top_order:
    print(top_order)
```

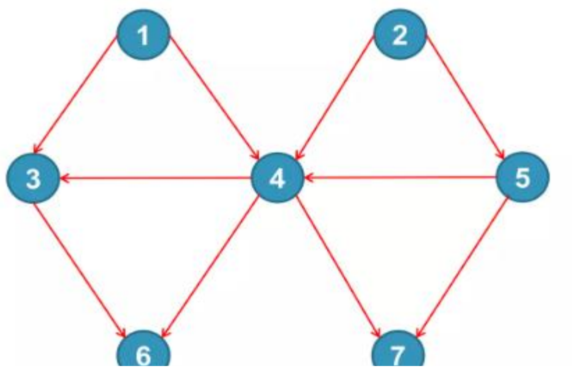
Output:



```
Adjacency List:
0:[1, 2]
1:[3, 4]
2:[5, 1]
3:[4]
4:[5]
5:[]

Topological Sort Order:
[0, 2, 1, 3, 4, 5]
```

Class Assignment

Q.1: Apply the program above to the following graph then explain the code line by line:



Q.2: Explain each line of code while applying the above code to a cyclic graph with at least three nodes