Problem 5 Given an directed graph with V vertices and E edges, check whether it contains any cycle or not. (using DFS)

```
from collections import defaultdict


class Graph:
    def __init__(self, vertices):
        self.vertices = vertices
        self.graph = defaultdict(list)

    def add_edge(self, u, v):
        self.graph[u].append(v)

    def is_cyclic_util(self, v, visited, recursion_stack):
        visited[v] = True
        recursion_stack[v] = True

        for neighbor in self.graph[v]:
            if not visited[neighbor]:
                if self.is_cyclic_util(neighbor, visited, recursion_stack):
                    return True
            elif recursion_stack[neighbor]:
                return True

        recursion_stack[v] = False
        return False

    def contains_cycle(self):
        visited = [False] * self.vertices
        recursion_stack = [False] * self.vertices

        for v in range(self.vertices):
```

```
            if not visited[v]:
                if self.is_cyclic_util(v, visited, recursion_stack):
                    return True

        return False

V = 4
E = 6

graph = Graph(V)
graph.add_edge(0, 1)
graph.add_edge(0, 2)
graph.add_edge(1, 2)
graph.add_edge(2, 0)
graph.add_edge(2, 3)
graph.add_edge(3, 3)

if graph.contains_cycle():
    print("The graph contains at least one cycle.")
else:
    print("The graph does not contain any cycle.")
```