IF2251 Strategi Algoritma

Tugas Kecil II

Penyusunan Rencana Kuliah dengan Topological Sort Menggunakan Penerapan Decrease and Conquer

Oleh:

Nathaniel Jason 13519108

I.Algoritma

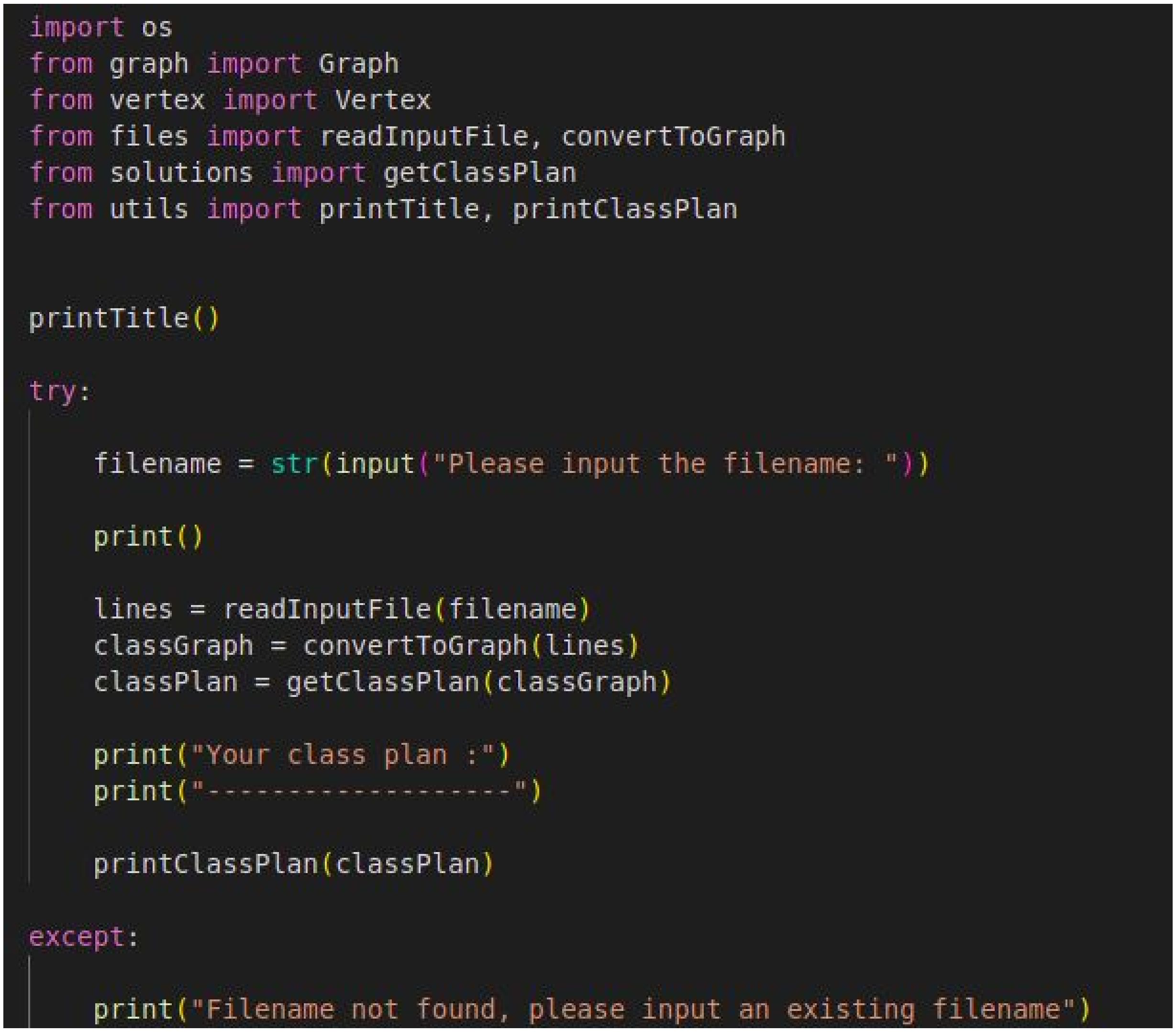
Algoritma yang digunakan dalam menyelesaikan permasalahan penyusunan rencana kuliah adalah algoritma topological sort. Algoritma topological sort secara singkat dapat diterapkan dengan cara memilih simpul simpul yang tidak memiliki derajat masuk pada setiap iterasinya. Algoritma topological sort juga sangatlah erat dengan konsep decrease and conquer. Proses decrease pada algoritma topological sort dapat kita lihat pada fase pemilihan simpul pada setiap iterasinya. Graph kemudian akan meng-decrease sejumlah simpul. Dapat dilihat bahwa pada setiap iterasinya, kita mengurangi input yang kita dapat, dengan tujuan untuk mencapai suatu solusi yang optimal.

Algoritma yang diterapkan untuk menyelesaikan persoalan penyusunan rencana kuliah adalah sebagai berikut.

1. Carilah semua simpul pada graph yang tidak memiliki derajat masuk sama sekali atau yang derajat masuknya sama dengan 0.
2. Semua simpul yang dipilih merupakan pilihan kelas yang optimal untuk semester ini.
3. Hapus semua simpul yang dipilih dari graph. Penghapusan juga harus mempertimbangkan simpul lain yang terhubung dengan simpul yang dihapus. Jika ada simpul yang berhubungan dengan simpul yang dihapus, maka kita juga harus mengkondisikan simpul tersebut sehingga jumlah derajat masuknya berkurang 1, dan simpul yang dihapus itu juga harus dihapus dari larik penyimpanan simpul yang masuk pada simpul tersebut.
4. Ulangi langkah 1 - 3 sampai graph kosong.

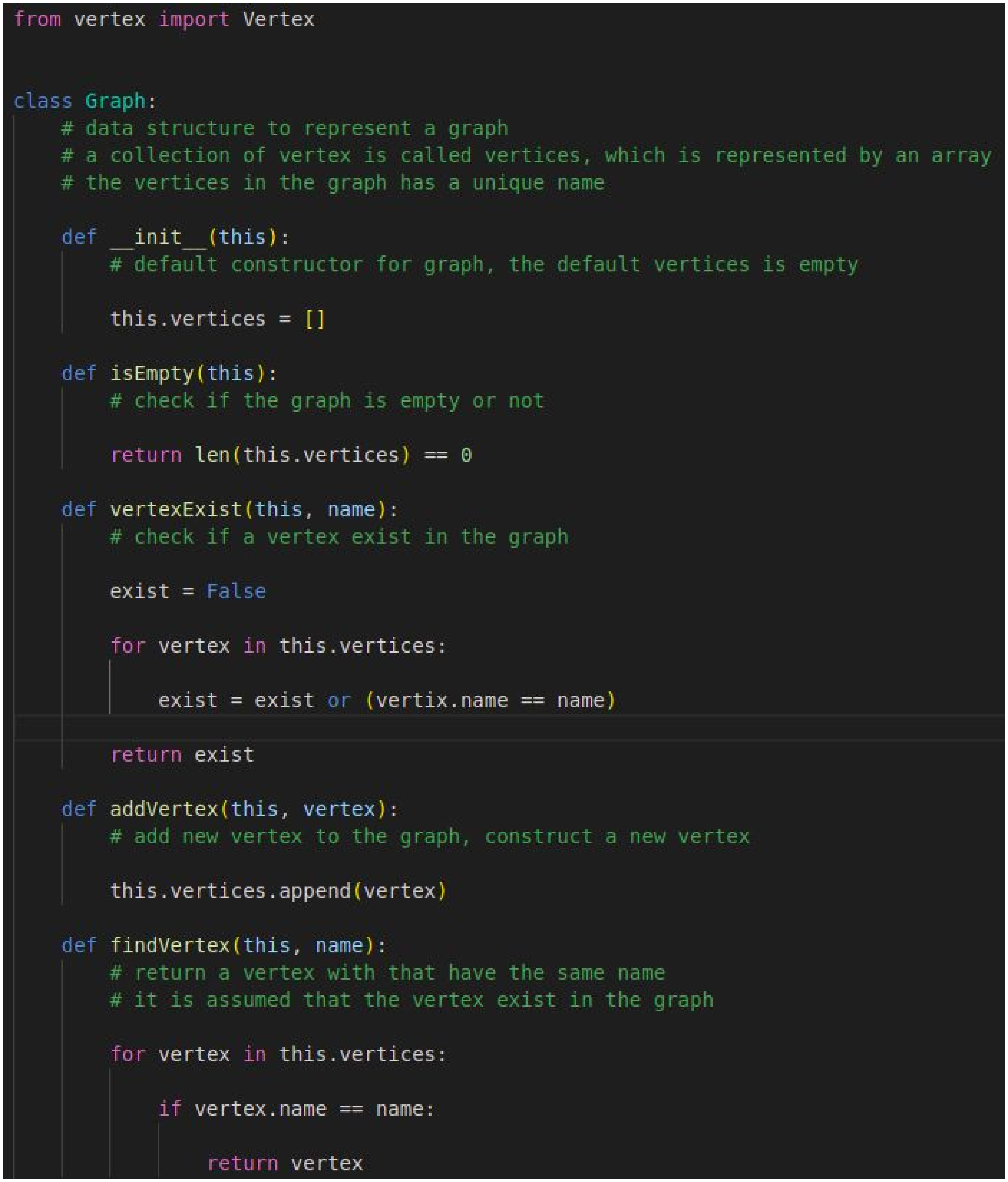
II.Source code

1. Modul_13519108.py

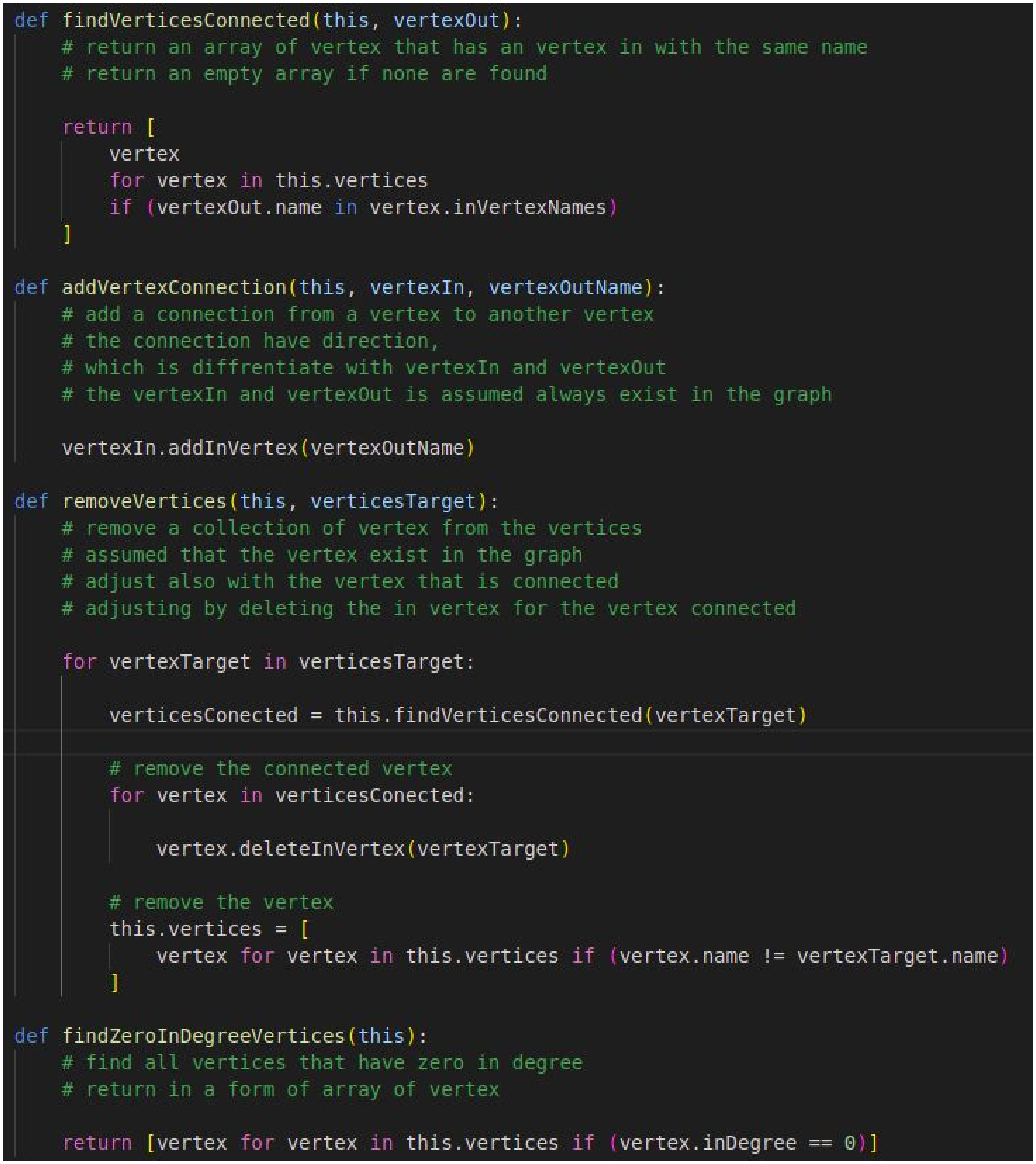


```
import os
from graph import Graph
from vertex import Vertex
from files import readInputFile, convertToGraph
from solutions import getClassPlan
from utils import printTitle, printClassPlan


printTitle()

try:

    filename = str(input("Please input the filename: "))

    print()

    lines = readInputFile(filename)
    classGraph = convertToGraph(lines)
    classPlan = getClassPlan(classGraph)

    print("Your class plan :")
    print("-------------------")

    printClassPlan(classPlan)

except:

    print("Filename not found, please input an existing filename")
```

2. graph.py

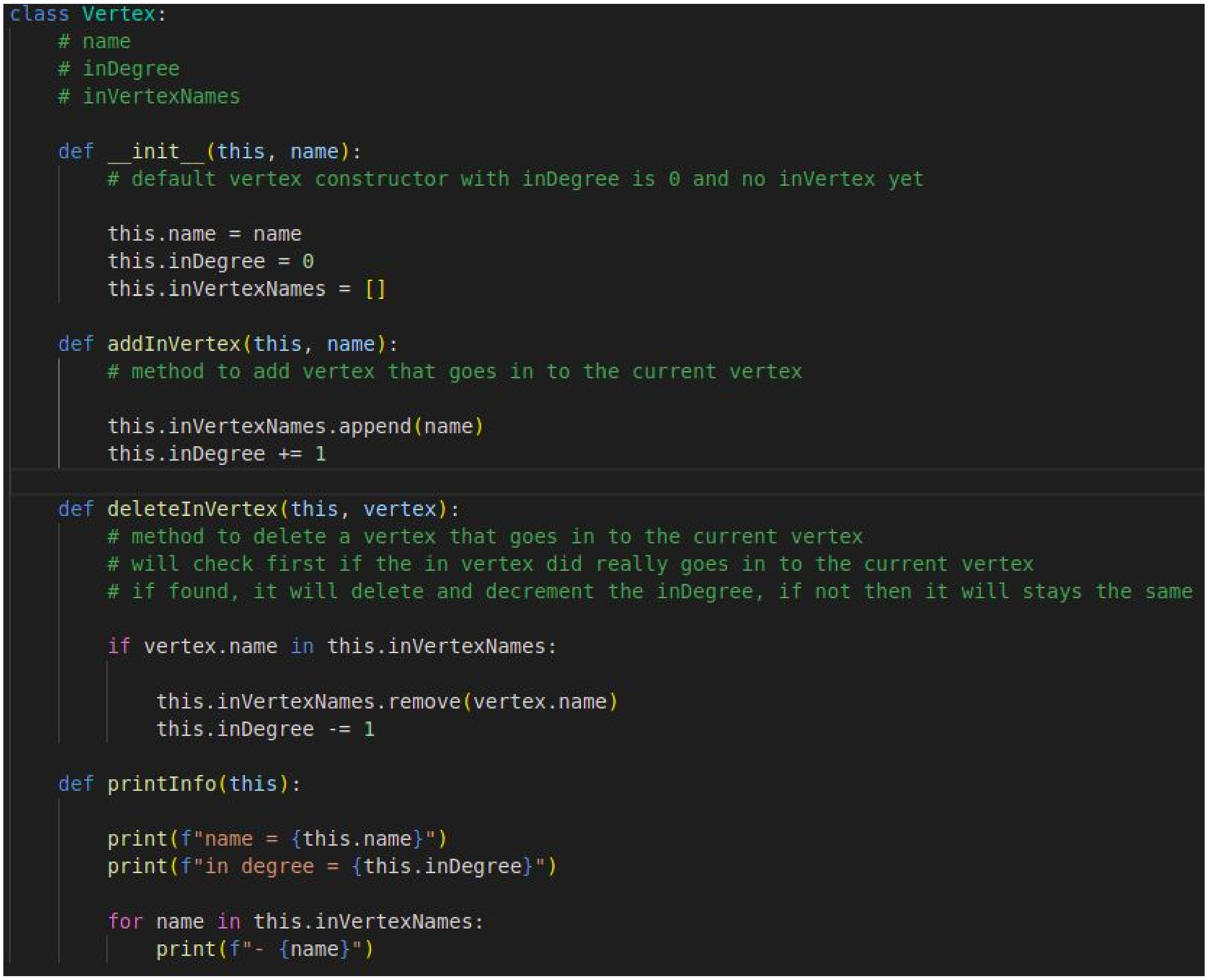


```
from vertex import Vertex


class Graph:
    # data structure to represent a graph
    # a collection of vertex is called vertices, which is represented by an array
    # the vertices in the graph has a unique name

    def __init__(this):
        # default constructor for graph, the default vertices is empty

        this.vertices = []

    def isEmpty(this):
        # check if the graph is empty or not

        return len(this.vertices) == 0

    def vertexExist(this, name):
        # check if a vertex exist in the graph

        exist = False

        for vertex in this.vertices:

            exist = exist or (vertix.name == name)

        return exist

    def addVertex(this, vertex):
        # add new vertex to the graph, construct a new vertex

        this.vertices.append(vertex)

    def findVertex(this, name):
        # return a vertex with that have the same name
        # it is assumed that the vertex exist in the graph

        for vertex in this.vertices:

            if vertex.name == name:

                return vertex
```



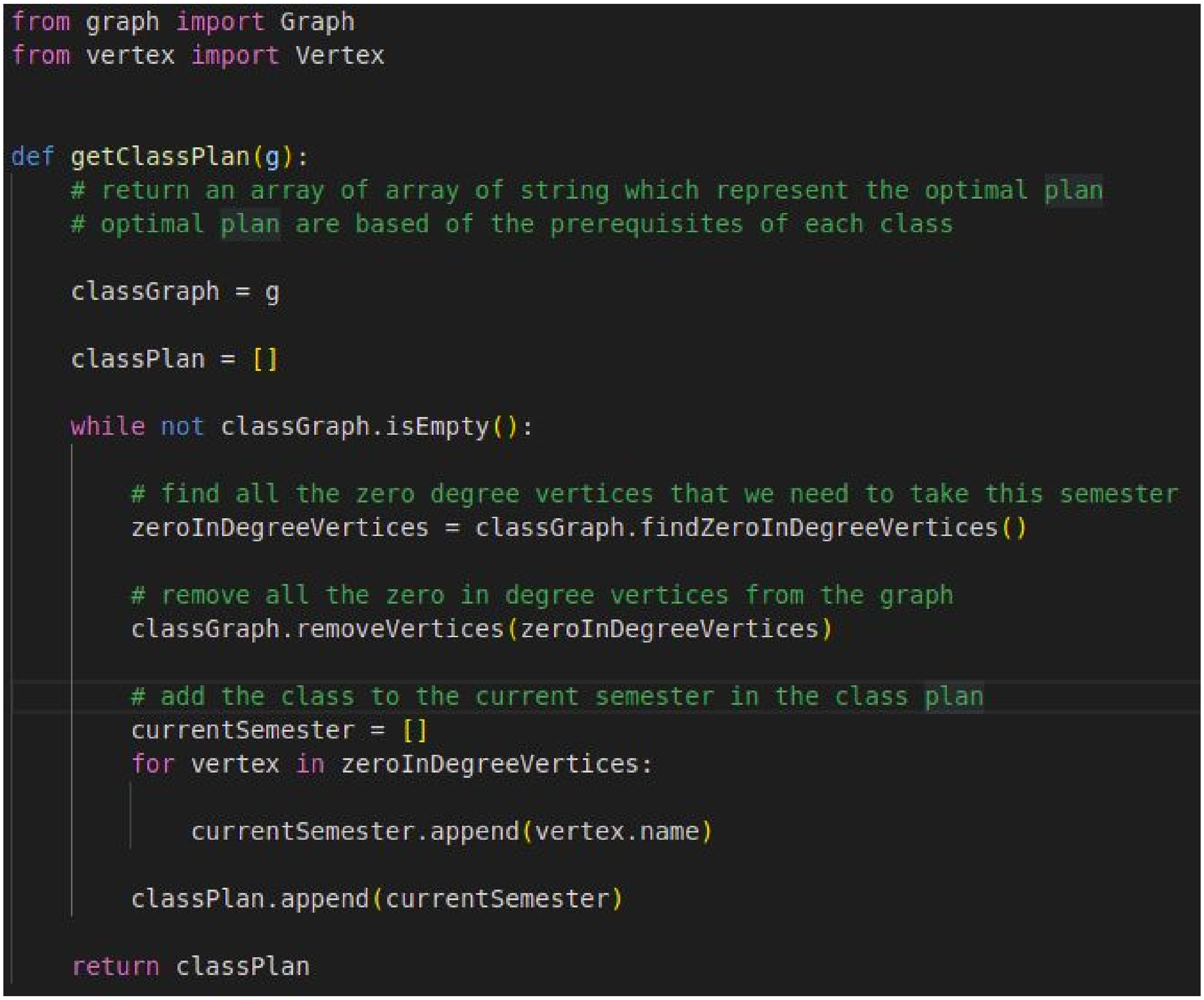
```
def findVerticesConnected(this, vertexOut):
    # return an array of vertex that has an vertex in with the same name
    # return an empty array if none are found

    return [
        vertex
        for vertex in this.vertices
        if (vertexOut.name in vertex.inVertexNames)
    ]

def addVertexConnection(this, vertexIn, vertexOutName):
    # add a connection from a vertex to another vertex
    # the connection have direction,
    # which is diffrentiate with vertexIn and vertexOut
    # the vertexIn and vertexOut is assumed always exist in the graph

    vertexIn.addInVertex(vertexOutName)

def removeVertices(this, verticesTarget):
    # remove a collection of vertex from the vertices
    # assumed that the vertex exist in the graph
    # adjust also with the vertex that is connected
    # adjusting by deleting the in vertex for the vertex connected

    for vertexTarget in verticesTarget:

        verticesConected = this.findVerticesConnected(vertexTarget)

        # remove the connected vertex
        for vertex in verticesConected:

            vertex.deleteInVertex(vertexTarget)

        # remove the vertex
        this.vertices = [
            vertex for vertex in this.vertices if (vertex.name != vertexTarget.name)
        ]

def findZeroInDegreeVertices(this):
    # find all vertices that have zero in degree
    # return in a form of array of vertex

    return [vertex for vertex in this.vertices if (vertex.inDegree == 0)]
```

3. vertex.py



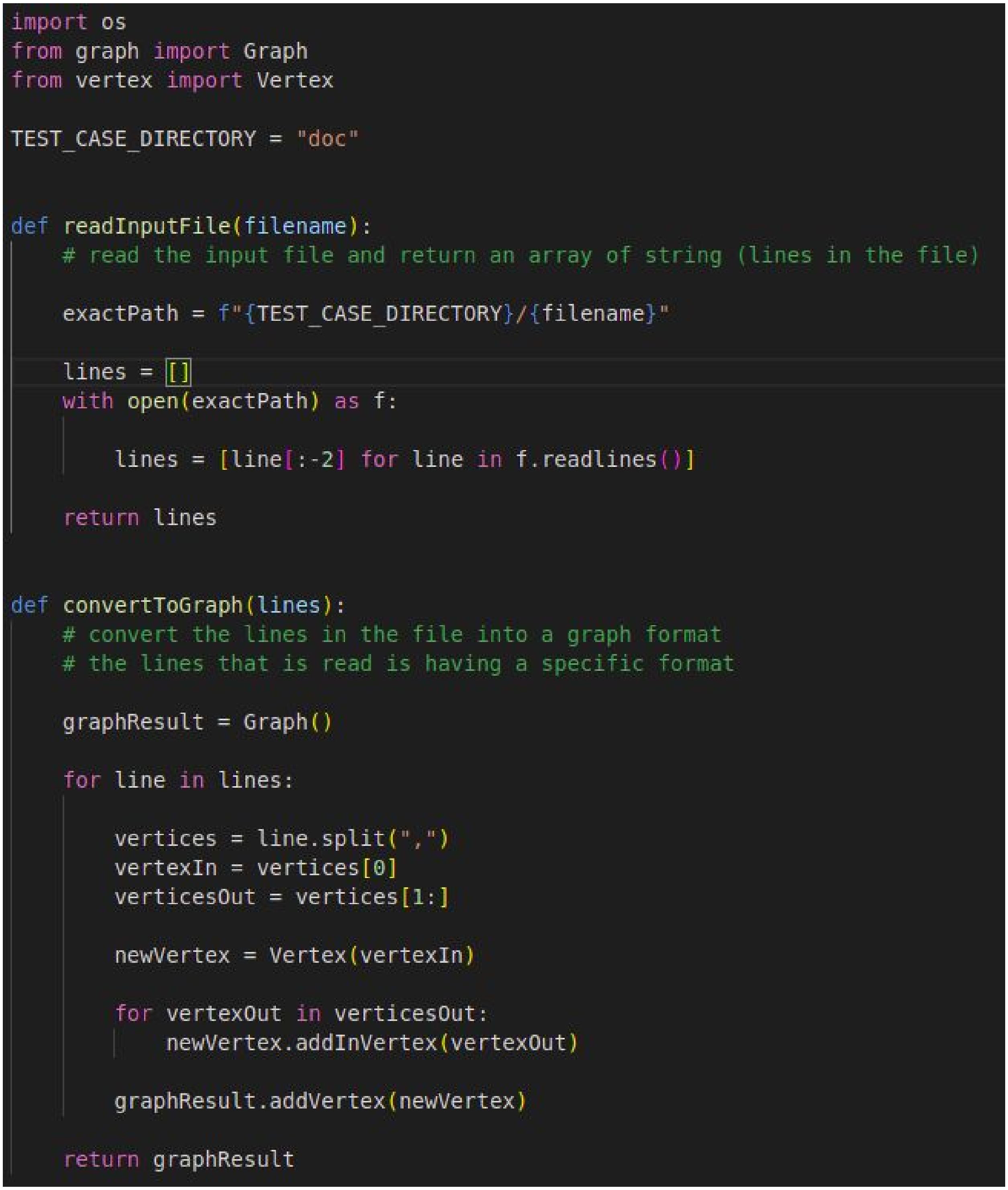
```python
class Vertex:
    # name
    # inDegree
    # inVertexNames

    def __init__(this, name):
        # default vertex constructor with inDegree is 0 and no inVertex yet

        this.name = name
        this.inDegree = 0
        this.inVertexNames = []

    def addInVertex(this, name):
        # method to add vertex that goes in to the current vertex

        this.inVertexNames.append(name)
        this.inDegree += 1

    def deleteInVertex(this, vertex):
        # method to delete a vertex that goes in to the current vertex
        # will check first if the in vertex did really goes in to the current vertex
        # if found, it will delete and decrement the inDegree, if not then it will stays the same

        if vertex.name in this.inVertexNames:

            this.inVertexNames.remove(vertex.name)
            this.inDegree -= 1

    def printInfo(this):

        print(f"name = {this.name}")
        print(f"in degree = {this.inDegree}")

        for name in this.inVertexNames:
            print(f"- {name}")
```

4. solutions.py



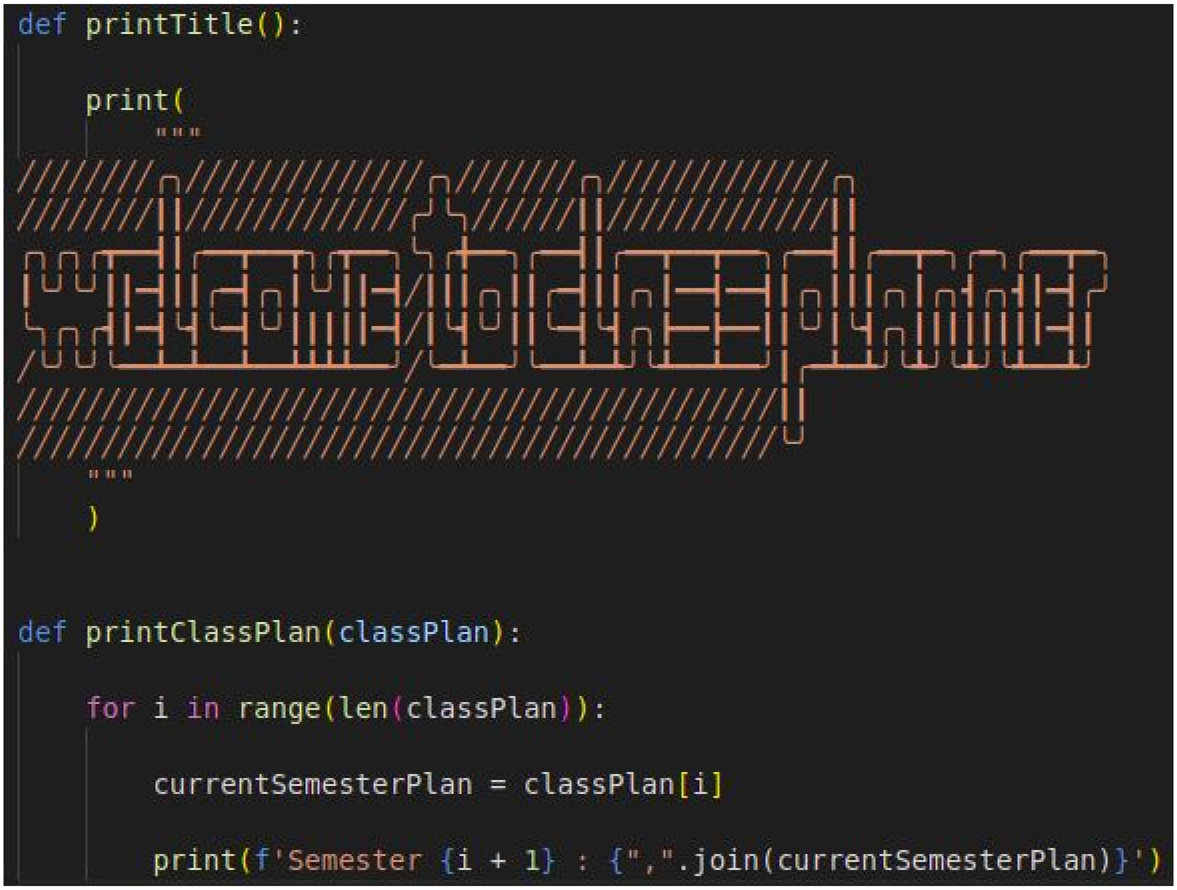
```
from graph import Graph
from vertex import Vertex


def getClassPlan(g):
    # return an array of array of string which represent the optimal plan
    # optimal plan are based of the prerequisites of each class

    classGraph = g

    classPlan = []

    while not classGraph.isEmpty():

        # find all the zero degree vertices that we need to take this semester
        zeroInDegreeVertices = classGraph.findZeroInDegreeVertices()

        # remove all the zero in degree vertices from the graph
        classGraph.removeVertices(zeroInDegreeVertices)

        # add the class to the current semester in the class plan
        currentSemester = []
        for vertex in zeroInDegreeVertices:

            currentSemester.append(vertex.name)

        classPlan.append(currentSemester)

    return classPlan
```

5. files.py



```
import os
from graph import Graph
from vertex import Vertex

TEST_CASE_DIRECTORY = "doc"


def readInputFile(filename):
    # read the input file and return an array of string (lines in the file)

    exactPath = f"{TEST_CASE_DIRECTORY}/{filename}"

    lines = []
    with open(exactPath) as f:

        lines = [line[:-2] for line in f.readlines()]

    return lines


def convertToGraph(lines):
    # convert the lines in the file into a graph format
    # the lines that is read is having a specific format

    graphResult = Graph()

    for line in lines:

        vertices = line.split(",")
        vertexIn = vertices[0]
        verticesOut = vertices[1:]

        newVertex = Vertex(vertexIn)

        for vertexOut in verticesOut:
            newVertex.addInVertex(vertexOut)

        graphResult.addVertex(newVertex)

    return graphResult
```

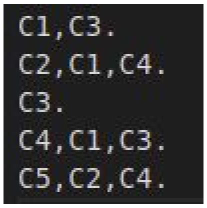
6. utils.py



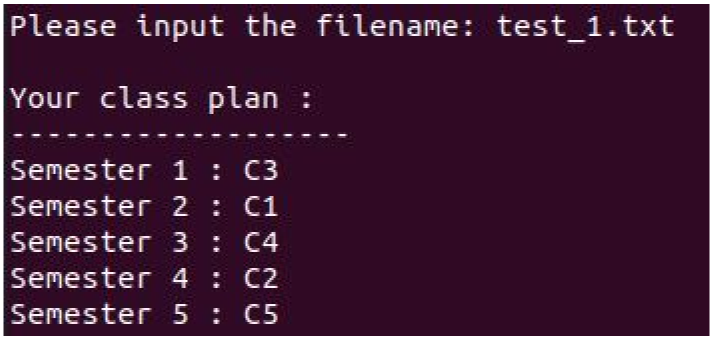
```python
def printTitle():

    print(
        """
    """
    )


def printClassPlan(classPlan):

    for i in range(len(classPlan)):

        currentSemesterPlan = classPlan[i]

        print(f'Semester {i + 1} : {",".join(currentSemesterPlan)}')
```

III.Test case

1.

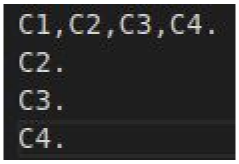


```
C1,C3.
C2,C1,C4.
C3.
C4,C1,C3.
C5,C2,C4.
```

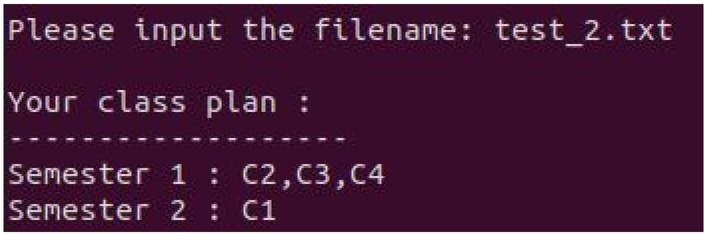


```
Please input the filename: test_1.txt

Your class plan :
-------------------
Semester 1 : C3
Semester 2 : C1
Semester 3 : C4
Semester 4 : C2
Semester 5 : C5
```

2.

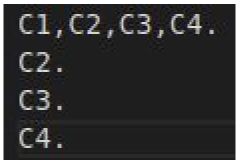


```
C1,C2,C3,C4.
C2.
C3.
C4.
```

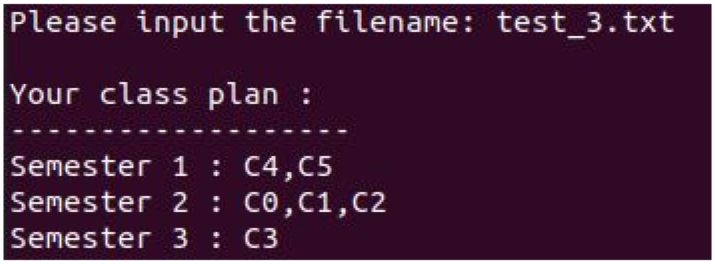


```
Please input the filename: test_2.txt

Your class plan :
-------------------
Semester 1 : C2,C3,C4
Semester 2 : C1
```

3.

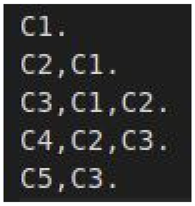


```
C1,C2,C3,C4.
C2.
C3.
C4.
```

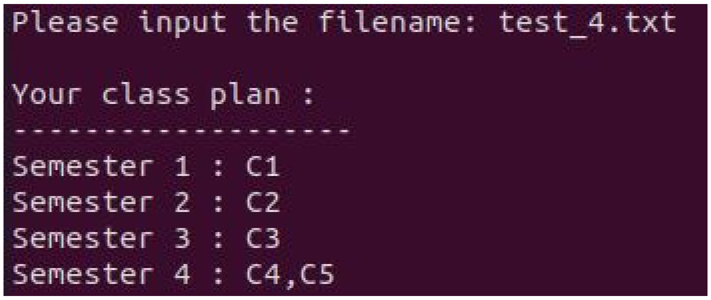


```
Please input the filename: test_3.txt

Your class plan :
-------------------
Semester 1 : C4,C5
Semester 2 : C0,C1,C2
Semester 3 : C3
```

4.

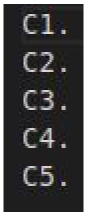


```
C1.
C2,C1.
C3,C1,C2.
C4,C2,C3.
C5,C3.
```

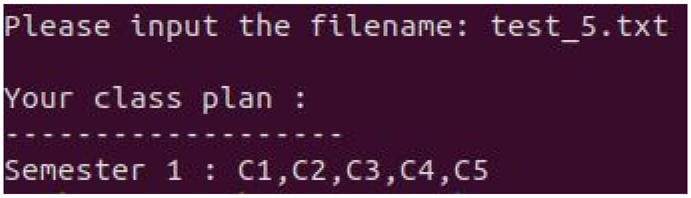


```
Please input the filename: test_4.txt

Your class plan :
-------------------
Semester 1 : C1
Semester 2 : C2
Semester 3 : C3
Semester 4 : C4,C5
```

5.

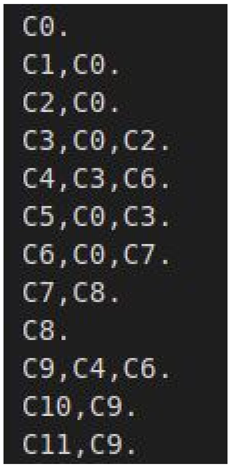


```
C1.
C2.
C3.
C4.
C5.
```

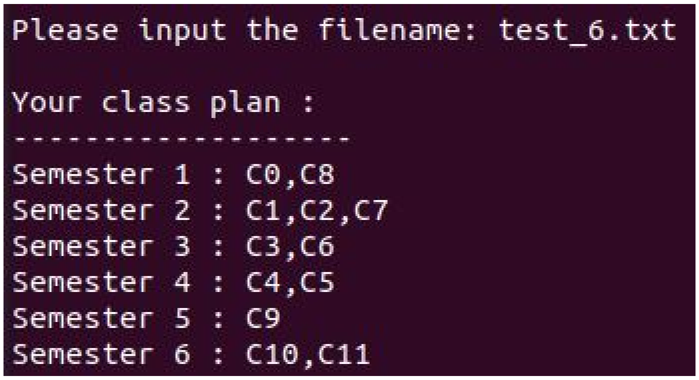


```
Please input the filename: test_5.txt

Your class plan :
--------------------
Semester 1 : C1,C2,C3,C4,C5
```

6.

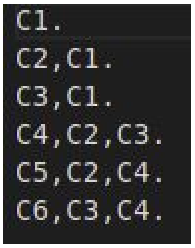


```
C0.
C1,C0.
C2,C0.
C3,C0,C2.
C4,C3,C6.
C5,C0,C3.
C6,C0,C7.
C7,C8.
C8.
C9,C4,C6.
C10,C9.
C11,C9.
```

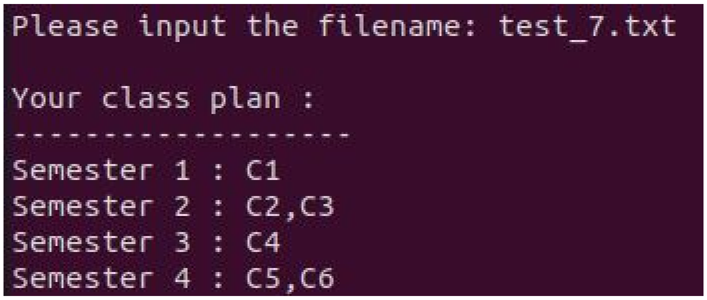


```
Please input the filename: test_6.txt

Your class plan :
--------------------
Semester 1 : C0,C8
Semester 2 : C1,C2,C7
Semester 3 : C3,C6
Semester 4 : C4,C5
Semester 5 : C9
Semester 6 : C10,C11
```

7.

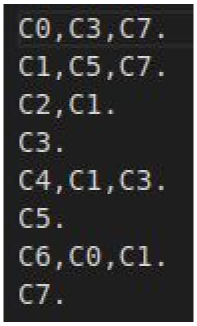


```
C1.
C2,C1.
C3,C1.
C4,C2,C3.
C5,C2,C4.
C6,C3,C4.
```

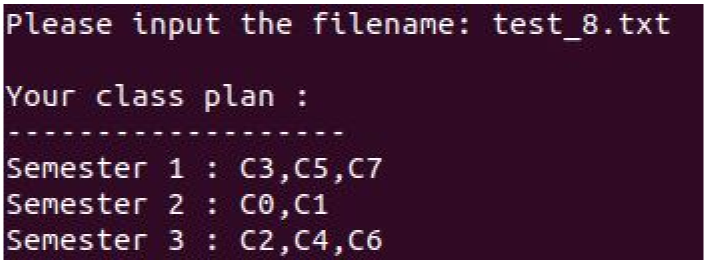


```
Please input the filename: test_7.txt

Your class plan :
--------------------
Semester 1 : C1
Semester 2 : C2,C3
Semester 3 : C4
Semester 4 : C5,C6
```

8.



```
C0,C3,C7.
C1,C5,C7.
C2,C1.
C3.
C4,C1,C3.
C5.
C6,C0,C1.
C7.
```



```
Please input the filename: test_8.txt

Your class plan :
-------------------
Semester 1 : C3,C5,C7
Semester 2 : C0,C1
Semester 3 : C2,C4,C6
```

IV.Alamat source code

https://github.com/nthnieljson/Tucil-2-Stima

V. Checklist

Poin	Ya	Tidak
1. Program berhasil dikompilasi	V	
2. Program berhasil *running*	V	
3. Program dapat menerima berkas input dan menuliskan output	V	
4. Luaran sudah benar untuk semua kasus input	V	