PENYUSUNAN RENCANA KULIAH DENGAN *TOPOLOGICAL SORT* (PENERAPAN *DECREASE AND CONQUER*)

Nguli Planner

LAPORAN TUGAS KECIL 2

Diajukan sebagai laporan dari tugas kecil dua mata kuliah IF2211 Strategi Algoritma pada Semester II Tahun Akademik 2020-2021

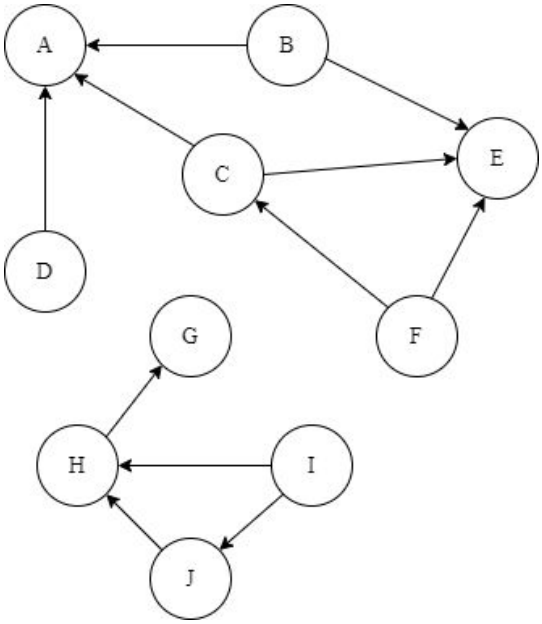
Nama	: Jeremia Axel
NIM	: 13519188
Kelas	: K-04
Bahasa yang digunakan	: Python3

TEKNIK INFORMATIKA
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG

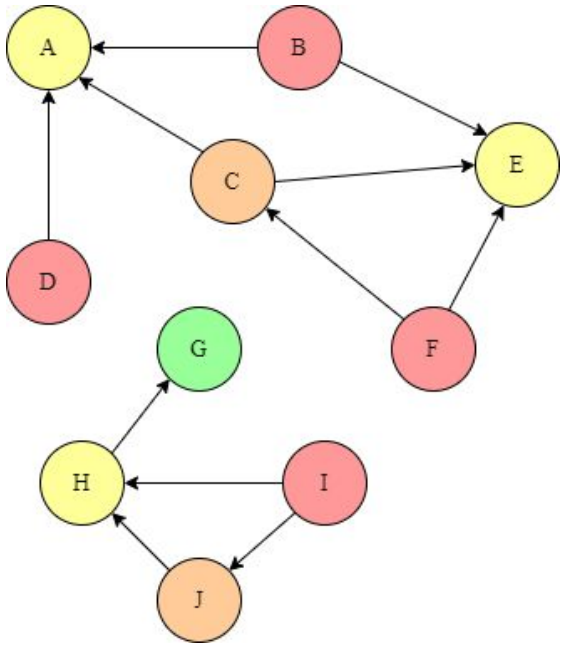
2021

1. Algoritma Topological Sort Dengan Pendekatan *Decrease and Conquer*

Algoritma Topological Sort adalah algoritma untuk melakukan pengurutan (*sorting*) terhadap graf berarah asiklik. Pengurutan dilakukan terhadap simpul-simpul dari graf berarah tersebut sedemikian sehingga setiap sisi berarah dari simpul x ke simpul y, dalam hasil *topological sort*-nya simpul x ada sebelum simpul y.



Gambar Diagram Asiklik



Gambar Urutan Topological Sorting*

***urutan : merah → jingga → kuning → hijau**

Pada gambar di atas, terdapat diagram asiklik tak terhubung yang membuat dua upagraf. Jika melakukan *topological sorting* pada graf ini, dihasilkan urutan :
B, D, F, I, C, J, A, E, H, G

Algoritma Topological Sort bermanfaat untuk membuat penjadwalan dari suatu tugas yang memiliki ketergantungan terhadap tugas lainnya. Dalam tugas kali ini, program dibuat untuk melakukan penyusunan rencana kuliah. Algoritma yang digunakan dalam program:

1. Mencari beberapa mata kuliah di daftar mata kuliah dengan nol prasyarat.
2. Menghapus mata kuliah-mata kuliah tersebut dari daftar prasyarat mata kuliah lain di daftar mata kuliah.
3. Pindahkan mata kuliah-mata kuliah tersebut ke dalam daftar hasil.
4. Ulangi dari proses nomor 1 sampai seluruh mata kuliah di daftar mata kuliah telah habis.
5. Tampilkan hasil dari mata kuliah-mata kuliah dalam daftar hasil terurut berdasarkan semester pengambilan.
6. Jika dalam prosesnya tidak terdapat mata kuliah dengan nol prasyarat, program dihentikan karena tidak memiliki keterurutan dan graf merupakan graf siklik (terdapat perputaran sisi-sisi berarah).

Dalam algoritma ini, setiap kali didapati mata kuliah-mata kuliah dengan nol prasyarat, maka jumlah data di pencarian berikutnya berkurang sebanyak n mata kuliah-mata kuliah yang sebelumnya memiliki nol prasyarat. Hal ini merupakan penerapan *decrease and conquer*.

2. *Source Code* Program Dalam Bahasa Python3

Kode program dalam laporan ini menggunakan bahasa pemrograman python3 seperti berikut:

main.py

```
# File : main.py
# program untuk melakukan topological sort

import sys

from pathlib import Path
from os.path import join

from myLib import FileParser
from myLib.DAGraph import DAGraph

def get_courses_str(list_of_courses) -> str:
    '''
    Fungsi menerima list dan menghasilkan string dari list tersebut:
    C1, C2, C3, ...
    '''
    if (len(list_of_courses) > 0):
        result = list_of_courses[0]
        for i in range(1, len(list_of_courses)):
            result += ", " + list_of_courses[i]
        return result
    return

def print_hasil(list_of_courses_plan):
    '''
    Prosedur menerima list dan menampikan hasil ke terminal.
    dengan format:
    Semester {angka romawi}{buffer spasi} : <course>, <course>, ...
    Misalkan
    Semester I   : C1
```

```
    Semester II : C2
    dst.
    '''
    roman_num = [FileParser.romanizer(i+1) for i in
range(len(list_of_courses_plan))]
    max = 0
    for rom in roman_num:
        if (len(rom) > max):
            max = len(rom)

    print("\nResult:")
    for i in range(len(list_of_courses_plan)):
        buffer = max-len(roman_num[i])
        print("Semester {num} ".format(num=roman_num[i]) + " "*buffer +
": {courses}.".format(courses=get_courses_str(list_of_courses_plan[i])))

def toposort_dnc(list_of_courses, list_of_courses_plan):
    '''
    prosedur melakukan topological sorting secara rekursif.
    Tiap rekursif, panjang list_of_courses berkurang sebanyak n item yang
memiliki nol preq
    '''
    if (len(list_of_courses) == 0):
        pass
    else :
        targets = []

        # mencari course-course dengan nol prequisite
        for course in list_of_courses:
            if len(course.getPreq()) == 0:
                # memasukan semua course yang preqnya nol ke dalam list
                targets.append(course.getMain())

        # jika terdapat course dengan nol prequisite
        if (len(targets) > 0):
            # memasukan semua course dengan nol preq ke dalam hasil
            list_of_courses_plan.append(targets)

            # menghapus courses dengan nol prequisite dari prequisite
```

```
courses lainnya
            for current_target in targets:
                for course in list_of_courses:
                    if (course.getPreq().has(current_target)):
                        course.remove_this_from_preq(current_target)

                # menghapus course dengan nol prequisite dari list of
courses utama
                for course in list_of_courses:
                    if course.getMain() == current_target:
                        list_of_courses.remove(course)

        # jika tidak terdapat course dengan nol prequisite berarti graf
tidak asiklik
        # dan program tidak dapat dilanjutkan
        else:
            print("No course with zero or taken prequisite found")
            print("Last course searched :
{}".format(list_of_courses_plan[-1][-1]))
            exit(1)

        # rekursif dengan list_of_courses yang sudah berkurang
        toposort_dnc(list_of_courses, list_of_courses_plan)


### MAIN ####
test_dir = join(Path(__file__).resolve().parent.parent, "test")

# input filename
if (len(sys.argv) > 1):
   filename = sys.argv[1]
   print("Test case file : " + filename)
else:
   print("File input otomatis dipindah ke folder test.")
   filename = input("Insert file name : ")

# open file and create list of (list of strings)
course_list = FileParser.file_to_list_of_courses(filename, test_dir)
```

```
# convert list of strings to Graph
# first string in the list of strings becomes the course, the rest
become the prequisites.
list_of_courses = []
for course in course_list:
   list_of_courses.append(DAGraph(course[0], course[1:]))

del course_list

# show the graph
print("Initial list of courses")
for course in list_of_courses:
   print(course, end=".\n")

# start topological sorting
list_of_courses_plan = []

# loop selama list of courses masih ada isinya
toposort_dnc(list_of_courses, list_of_courses_plan)

# menampilkan hasil akhir
print_hasil(list_of_courses_plan)
```

FileParser.py

```
from os.path import join
import os

def remove_excess_whitespace(word):
    i = 0
    while word[i] == " ":
        i += 1
    j = len(word)-1
    while word[j] == " ":
        j -= 1
    for k in range(j-i+1):
        new_word = word[i:j+1]
    return new_word
```

```
def file_to_list_of_courses(filename, test_dir=None) -> list:
    '''
    Takes test directory path (test_dir) and name of the file (filename)
    Return a list of strings.

    if test_dir is not declared, automatically switch to current
directory
    '''
    try:
        if (test_dir is None):
            test_dir = os.getcwd()

        f = open(join(test_dir, filename), 'r')
        lines = f.read().splitlines()
        f.close()
        print("File read success")

        list_of_soals = []
        for line in lines:
            if (line != ""):
                line = line.replace(".", "").split(",")
                for i in range(len(line)):
                    line[i] = remove_excess_whitespace(line[i])

                list_of_soals.append(line)

        return list_of_soals

    except FileNotFoundError:
        print("File not found")
        exit(1)

    except TypeError:
        print("Input error")
        exit(1)

def romanizer(number):
    arabian_to_roman_dict = [(1000, 'M'), (900, 'CM'), (500, 'D'), (400,
'CD'),
```

```
        ( 100, 'C'), ( 90, 'XC'), ( 50, 'L'), ( 40, 'XL'),
        (  10, 'X'), (  9, 'IX'), (  5, 'V'), (  4, 'IV'),
        (   1, 'I')]

    num = number
    result = ''
    i = 0
    while num > 0:
        while arabian_to_roman_dict[i][0] > num:
            i += 1
        result += arabian_to_roman_dict[i][1]
        num -= arabian_to_roman_dict[i][0]

    return result
```

DAGraph.py

```
from myLib.ListOfCourses import ListOfCourses

class DAGraph :
    ### Construct, Representation, and Destruct ###
    def __init__(self, main, preq=None) -> None:
        self.__main = main
        self.__preq = ListOfCourses(preq)

    def __repr__(self):
        name = self.__main
        if (len(self.__preq.getCourses()) > 0):
            name += ", " + str(self.__preq)

        return name

    def __str__(self) -> str:
        name = self.__main
        if (len(self.__preq) > 0):
            name += ", " + str(self.__preq)

        return name

    def destruct(self) -> None:
```

```
        del self

    ### Printing ###
    def print(self) -> None:
        print("{main}".format(main=self.__main), end="")

        # print(self.preq)
        for item in self.__preq:
            print(", {preq}".format(preq=item), end="")

        print(".", end="\n")

    ### Getter and Setter ###
    def getMain(self):
        return self.__main

    def setMain(self, newMain):
        self.__main = newMain

    def getPreq(self):
        return self.__preq

    def setPreq(self, newPreq):
        self.__preq.setCourses(newPreq)

    def add_this_to_preq(self, add_preq) -> None:
        self.__preq.addCourse(add_preq)

    def remove_this_from_preq(self, removed_preq) -> None:
        self.__preq.removeCourse(removed_preq)
```

ListOfCourses.py

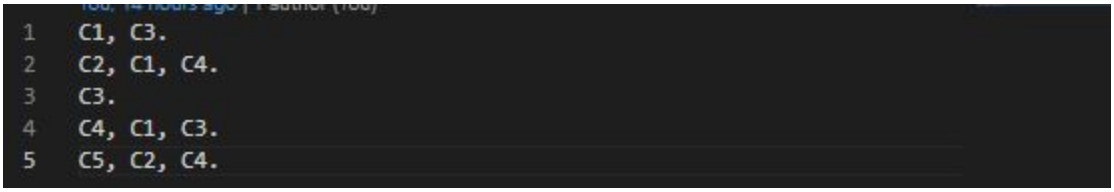
```
class ListOfCourses:
    def __init__(self, list_of_course=None) -> None:
        if (list_of_course is None):
            list_of_course = []
        elif (type(list_of_course) != "list"):
```

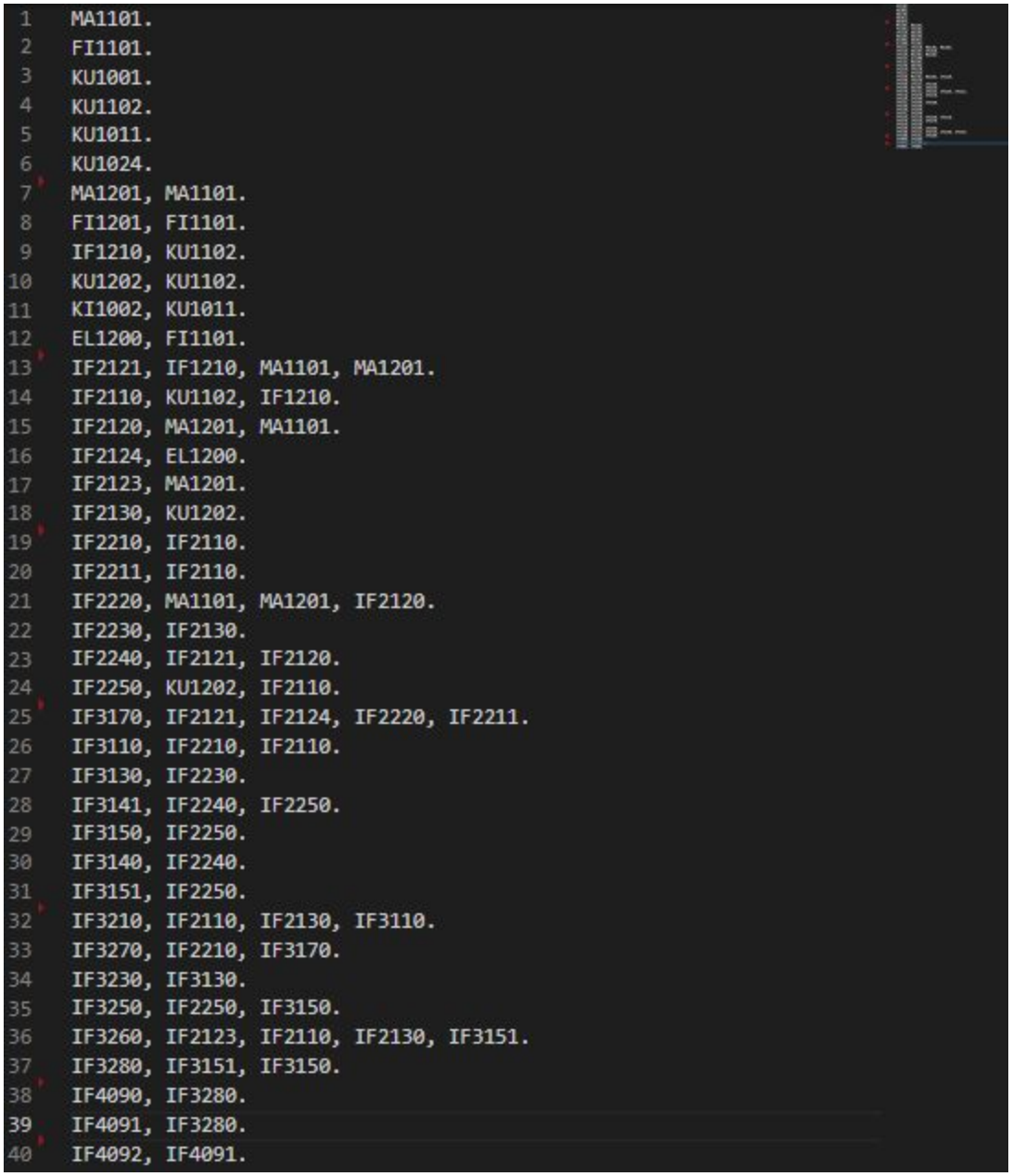
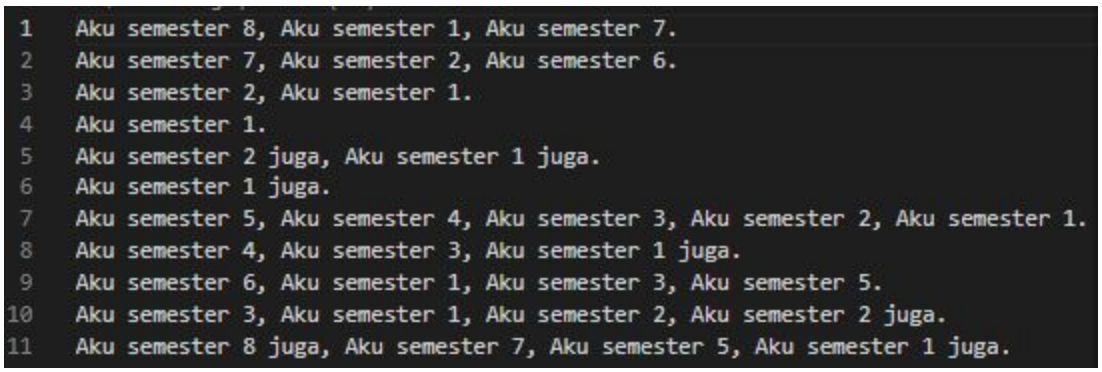
```
            list_of_course = list(list_of_course)
        self.__list_of_course = list_of_course

    def __repr__(self):
        name = ""
        for i in range(len(self.__list_of_course)):
            name += self.__list_of_course[i]
            if (i != len(self.__list_of_course)-1):
                name += ", "

        return name

    def __str__(self) -> str:
        name = ""
        for i in range(len(self.__list_of_course)):
            name += self.__list_of_course[i]
            if (i != len(self.__list_of_course)-1):
                name += ", "

        return name

    def __len__(self) -> int:
        return len(self.__list_of_course)

    def getCourses(self):
        return self.__list_of_course

    def setCourses(self, newCourses):
        if (type(newCourses) != "list"):
            newCourses = list(newCourses)

        self.__list_of_course = newCourses

    def addCourse(self, add_course) -> None:
        self.__list_of_course.append(add_course)

    def removeCourse(self, removed_course) -> None:
        self.__list_of_course.remove(removed_course)
```

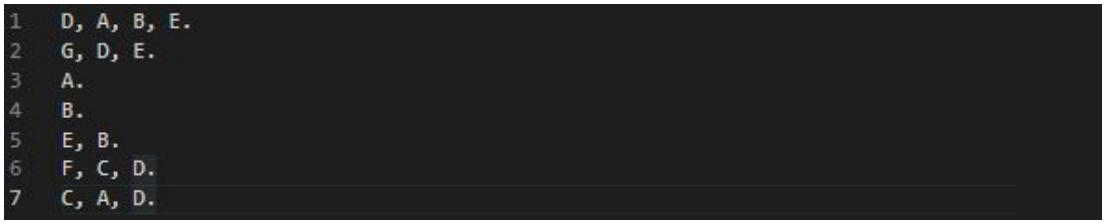
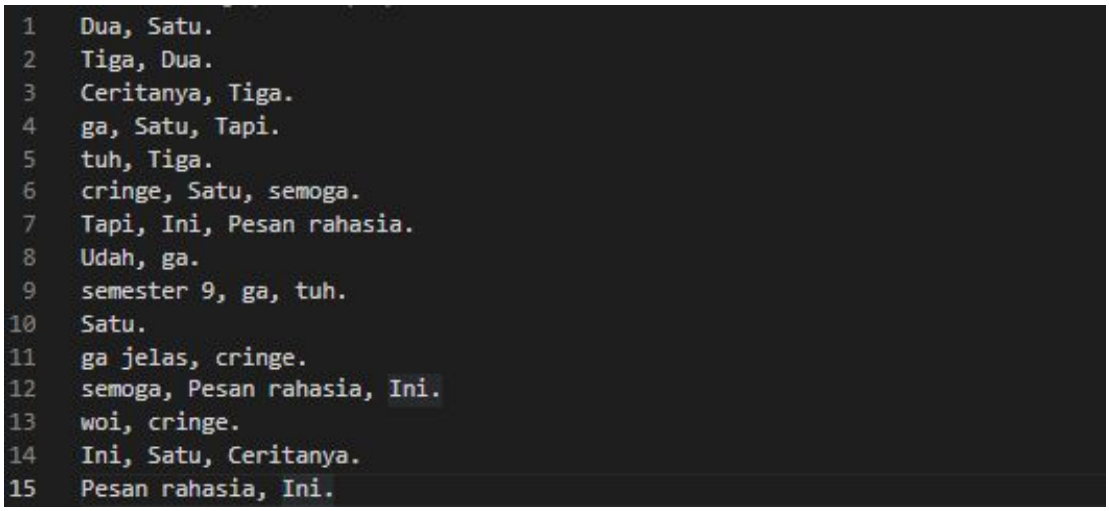
```
    def has(self, search) -> bool:
        return search in self.__list_of_course
```

3. Tangkapan Layar Input/Output

Berikut hasil tangkapan layar dari delapan contoh input/output,

Input :	
1	
2	

3	 MA1101. FI1101. KU1001. KU1102. KU1011. KU1024. MA1201, MA1101. FI1201, FI1101. IF1210, KU1102. KU1202, KU1102. KI1002, KU1011. EL1200, FI1101. IF2121, IF1210, MA1101, MA1201. IF2110, KU1102, IF1210. IF2120, MA1201, MA1101. IF2124, EL1200. IF2123, MA1201. IF2130, KU1202. IF2210, IF2110. IF2211, IF2110. IF2220, MA1101, MA1201, IF2120. IF2230, IF2130. IF2240, IF2121, IF2120. IF2250, KU1202, IF2110. IF3170, IF2121, IF2124, IF2220, IF2211. IF3110, IF2210, IF2110. IF3130, IF2230. IF3141, IF2240, IF2250. IF3150, IF2250. IF3140, IF2240. IF3151, IF2250. IF3210, IF2110, IF2130, IF3110. IF3270, IF2210, IF3170. IF3230, IF3130. IF3250, IF2250, IF3150. IF3260, IF2123, IF2110, IF2130, IF3151. IF3280, IF3151, IF3150. IF4090, IF3280. IF4091, IF3280. IF4092, IF4091.
4	 Aku semester 8, Aku semester 1, Aku semester 7. Aku semester 7, Aku semester 2, Aku semester 6. Aku semester 2, Aku semester 1. Aku semester 1. Aku semester 2 juga, Aku semester 1 juga. Aku semester 1 juga. Aku semester 5, Aku semester 4, Aku semester 3, Aku semester 2, Aku semester 1. Aku semester 4, Aku semester 3, Aku semester 1 juga. Aku semester 6, Aku semester 1, Aku semester 3, Aku semester 5. Aku semester 3, Aku semester 1, Aku semester 2, Aku semester 2 juga. Aku semester 8 juga, Aku semester 7, Aku semester 5, Aku semester 1 juga.

5	 3, 1, 2. 5, 3. 7, 4. 2, 1. 11, 3, 5, 7, 9, 8, 10. 8, 6. 10, 8, 7. 1. 4, 2. 9, 6. 6, 4.
6	 D, A, B, E. G, D, E. A. B. E, B. F, C, D. C, A, D.
7	 Dua, Satu. Tiga, Dua. Ceritanya, Tiga. ga, Satu, Tapi. tuh, Tiga. cringe, Satu, semoga. Tapi, Ini, Pesan rahasia. Udah, ga. semester 9, ga, tuh. Satu. ga jelas, cringe. semoga, Pesan rahasia, Ini. woi, cringe. Ini, Satu, Ceritanya. Pesan rahasia, Ini.

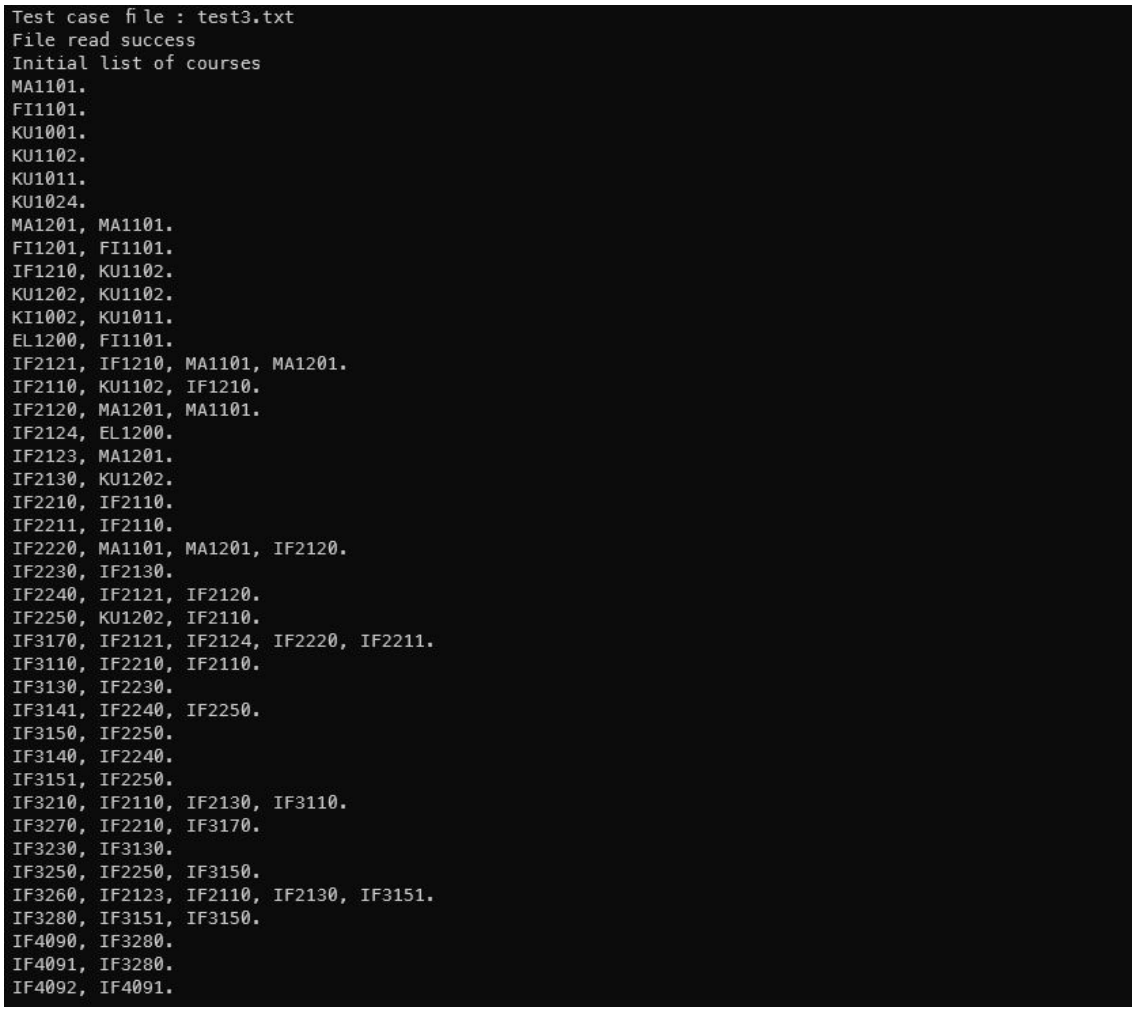
8

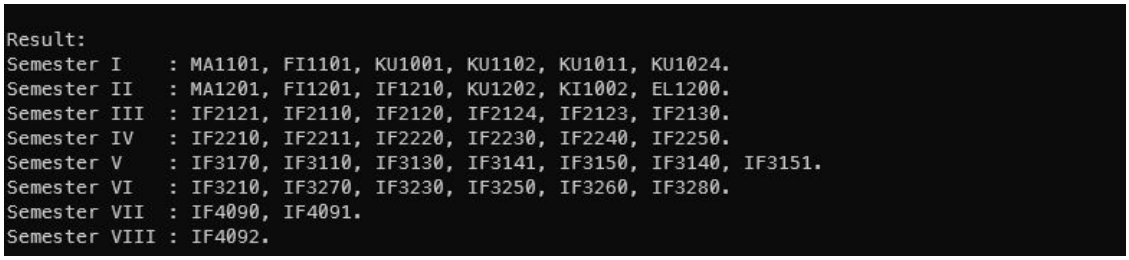
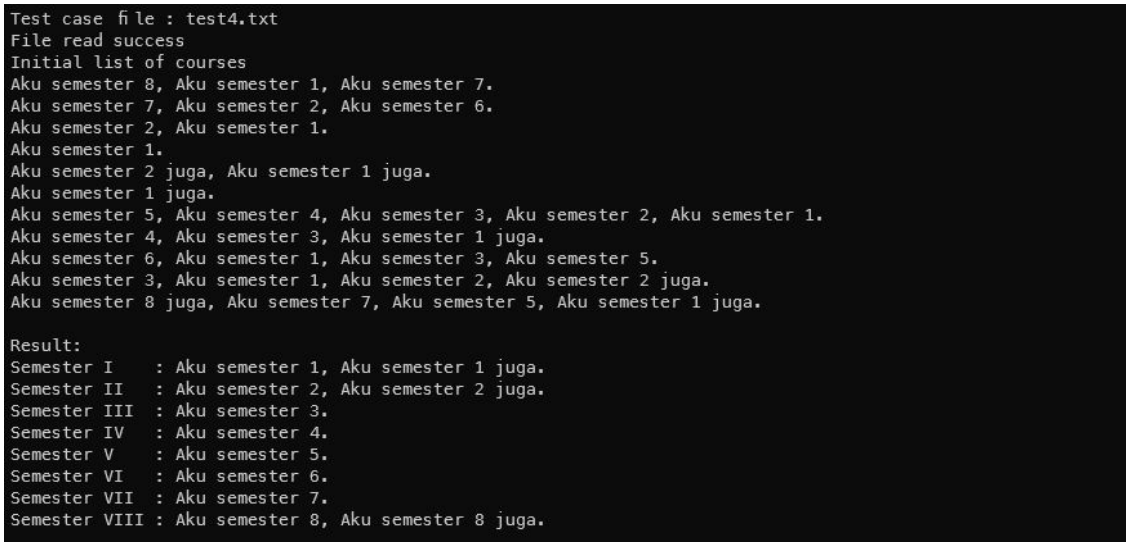
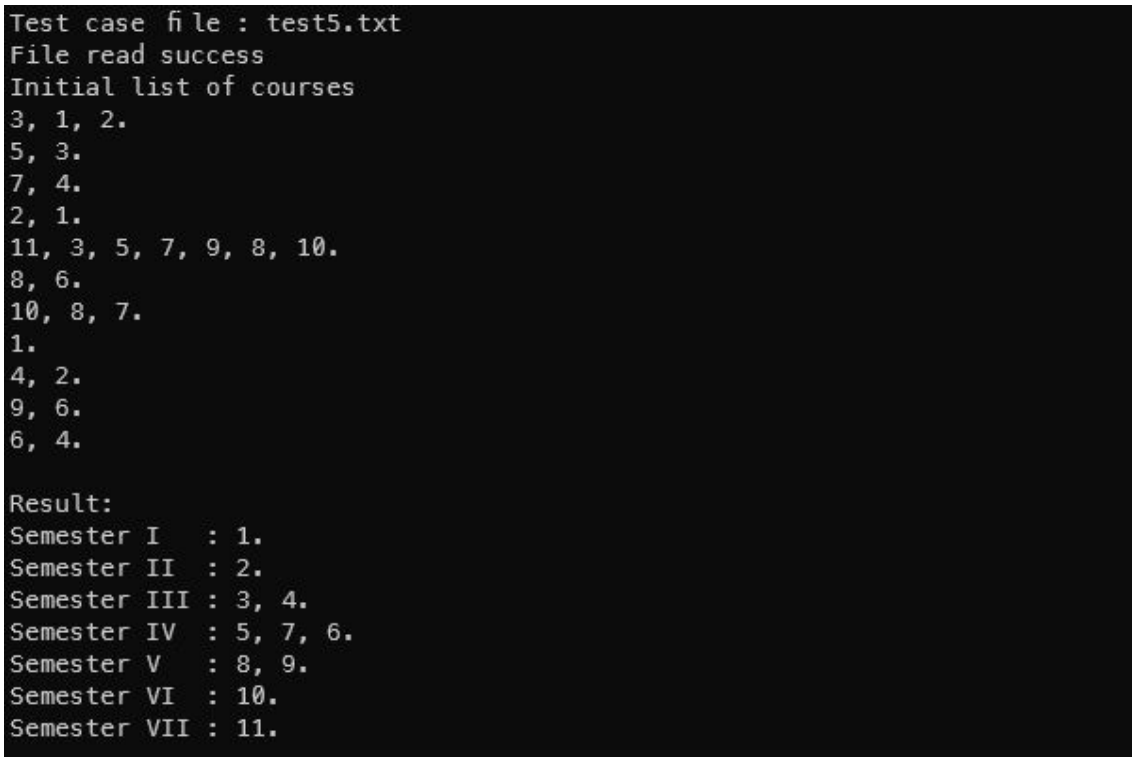
```
Inget, makan, pempek, padat, bikin, capek.
Buat, pempek, padat.
Pempeknya, pempek, AUUUUUUUUU.
Upin, AUUUUUUUUU.
belom, Upin, test, makan, pempek.
AAAAAAAAAA.

Ipin, UAAAAAAAAA.
AUUUUUUUUU.
terlalu, makan.
test, terlalu, makan.

belajar, bikin, Upin, Ipin.
UAAAAAAAAA.
makan, AAAAAAAAAA, AUUUUUUUUU.

buat, Buat, bikin, Ipin.
case, Pempeknya, Ipin.
BasDat, bikin, capek, MIAW.
bikin, padat.
padat, Upin.
pempek, MIAW.
MIAW.
capek, makan, Pempeknya.
```

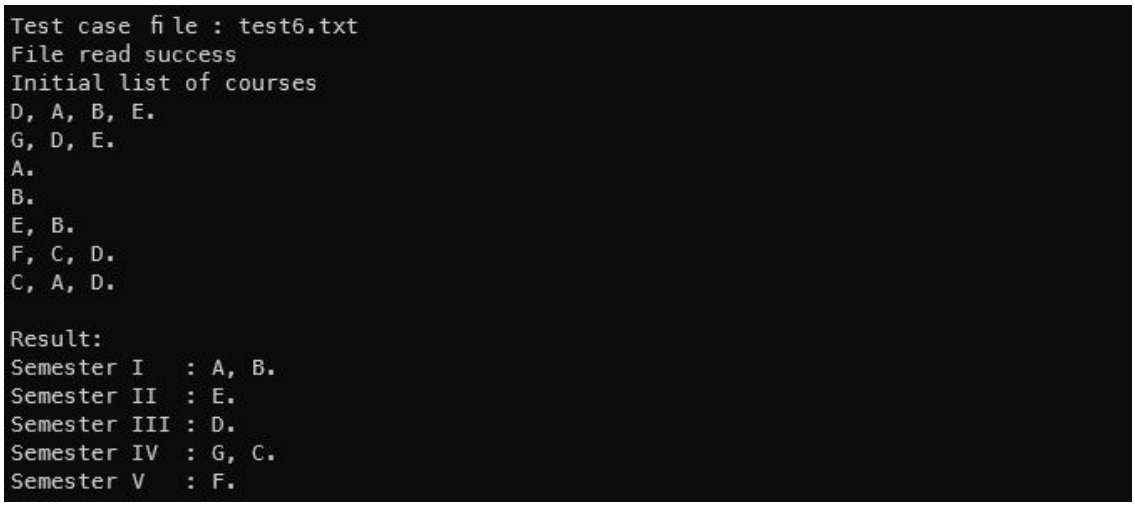
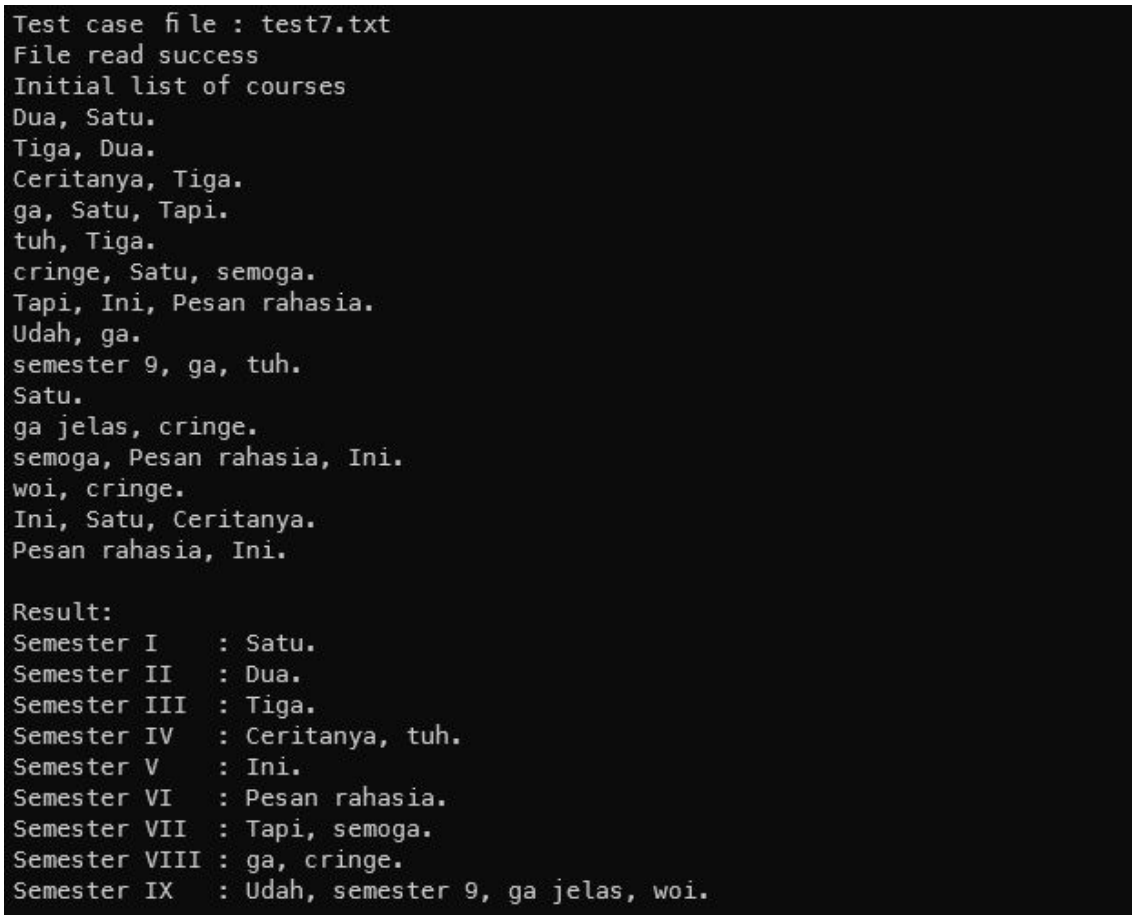
Output:

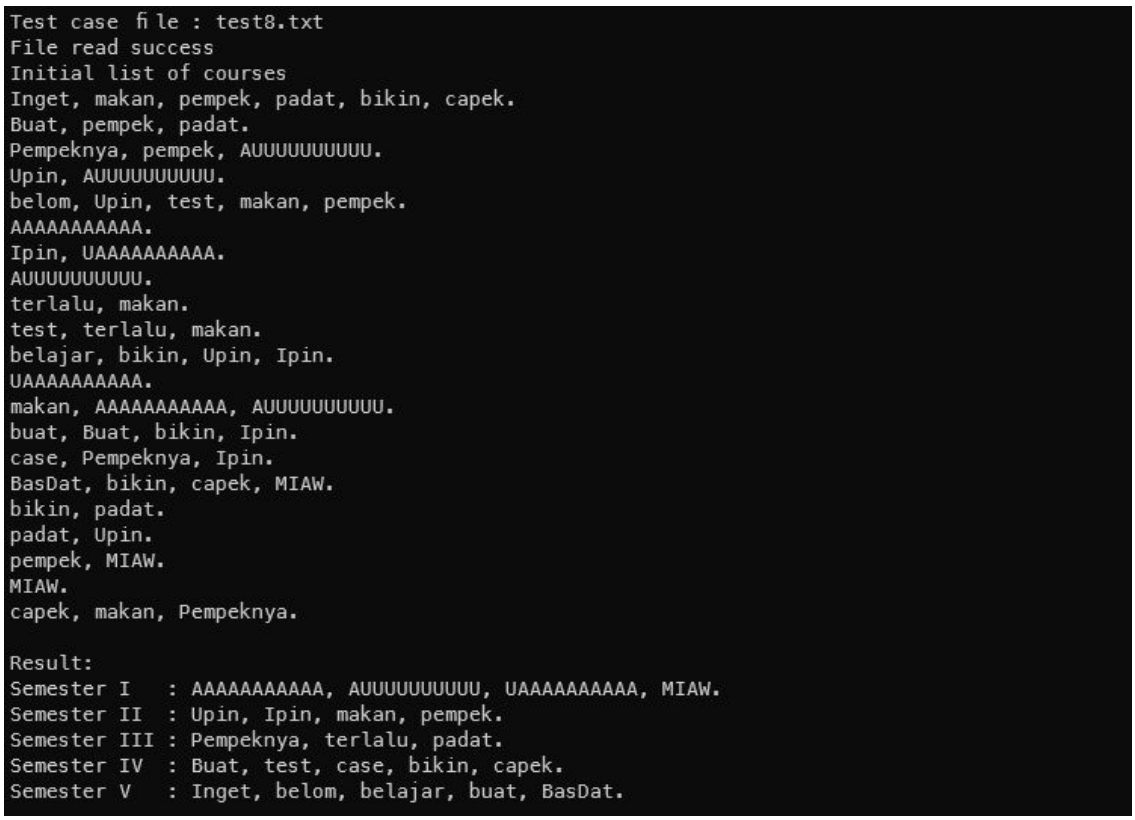
1

```
Test case file : test1.txt
File read success
Initial list of courses
C1, C3.
C2, C1, C4.
C3.
C4, C1, C3.
C5, C2, C4.

Result:
Semester I   : C3.
Semester II  : C1.
Semester III : C4.
Semester IV  : C2.
Semester V   : C5.
```

2	 Test case file : test2.txt File read success Initial list of courses A, B, C, D. B. C, F. D. E, B, C, F. F. G, H. H, I, J. I. J, I. Result: Semester I : B, D, F, I. Semester II : C, J. Semester III : A, E, H. Semester IV : G.
3	 Test case file : test3.txt File read success Initial list of courses MA1101. FI1101. KU1001. KU1102. KU1011. KU1024. MA1201, MA1101. FI1201, FI1101. IF1210, KU1102. KU1202, KU1102. KI1002, KU1011. EL1200, FI1101. IF2121, IF1210, MA1101, MA1201. IF2110, KU1102, IF1210. IF2120, MA1201, MA1101. IF2124, EL1200. IF2123, MA1201. IF2130, KU1202. IF2210, IF2110. IF2211, IF2110. IF2220, MA1101, MA1201, IF2120. IF2230, IF2130. IF2240, IF2121, IF2120. IF2250, KU1202, IF2110. IF3170, IF2121, IF2124, IF2220, IF2211. IF3110, IF2210, IF2110. IF3130, IF2230. IF3141, IF2240, IF2250. IF3150, IF2250. IF3140, IF2240. IF3151, IF2250. IF3210, IF2110, IF2130, IF3110. IF3270, IF2210, IF3170. IF3230, IF3130. IF3250, IF2250, IF3150. IF3260, IF2123, IF2110, IF2130, IF3151. IF3280, IF3151, IF3150. IF4090, IF3280. IF4091, IF3280. IF4092, IF4091.

	 Result: Semester I : MA1101, FI1101, KU1001, KU1102, KU1011, KU1024. Semester II : MA1201, FI1201, IF1210, KU1202, KI1002, EL1200. Semester III : IF2121, IF2110, IF2120, IF2124, IF2123, IF2130. Semester IV : IF2210, IF2211, IF2220, IF2230, IF2240, IF2250. Semester V : IF3170, IF3110, IF3130, IF3141, IF3150, IF3140, IF3151. Semester VI : IF3210, IF3270, IF3230, IF3250, IF3260, IF3280. Semester VII : IF4090, IF4091. Semester VIII : IF4092.
4	 Test case file : test4.txt File read success Initial list of courses Aku semester 8, Aku semester 1, Aku semester 7. Aku semester 7, Aku semester 2, Aku semester 6. Aku semester 2, Aku semester 1. Aku semester 1. Aku semester 2 juga, Aku semester 1 juga. Aku semester 1 juga. Aku semester 5, Aku semester 4, Aku semester 3, Aku semester 2, Aku semester 1. Aku semester 4, Aku semester 3, Aku semester 1 juga. Aku semester 6, Aku semester 1, Aku semester 3, Aku semester 5. Aku semester 3, Aku semester 1, Aku semester 2, Aku semester 2 juga. Aku semester 8 juga, Aku semester 7, Aku semester 5, Aku semester 1 juga. Result: Semester I : Aku semester 1, Aku semester 1 juga. Semester II : Aku semester 2, Aku semester 2 juga. Semester III : Aku semester 3. Semester IV : Aku semester 4. Semester V : Aku semester 5. Semester VI : Aku semester 6. Semester VII : Aku semester 7. Semester VIII : Aku semester 8, Aku semester 8 juga.
5	 Test case file : test5.txt File read success Initial list of courses 3, 1, 2. 5, 3. 7, 4. 2, 1. 11, 3, 5, 7, 9, 8, 10. 8, 6. 10, 8, 7. 1. 4, 2. 9, 6. 6, 4. Result: Semester I : 1. Semester II : 2. Semester III : 3, 4. Semester IV : 5, 7, 6. Semester V : 8, 9. Semester VI : 10. Semester VII : 11.

6	 Test case file : test6.txt File read success Initial list of courses D, A, B, E. G, D, E. A. B. E, B. F, C, D. C, A, D. Result: Semester I : A, B. Semester II : E. Semester III : D. Semester IV : G, C. Semester V : F.
7	 Test case file : test7.txt File read success Initial list of courses Dua, Satu. Tiga, Dua. Ceritanya, Tiga. ga, Satu, Tapi. tuh, Tiga. cringe, Satu, semoga. Tapi, Ini, Pesan rahasia. Udah, ga. semester 9, ga, tuh. Satu. ga jelas, cringe. semoga, Pesan rahasia, Ini. woi, cringe. Ini, Satu, Ceritanya. Pesan rahasia, Ini. Result: Semester I : Satu. Semester II : Dua. Semester III : Tiga. Semester IV : Ceritanya, tuh. Semester V : Ini. Semester VI : Pesan rahasia. Semester VII : Tapi, semoga. Semester VIII : ga, cringe. Semester IX : Udah, semester 9, ga jelas, woi.

8	

4. Alamat Kode Sumber Program

Kode program yang digunakan dalam laporan ini dapat didapatkan di pranala berikut:
https://github.com/jeremiaaxel/Stima_Tucil2

5. Checklist

No.	Poin	Ya	Tidak
1.	Program berhasil dikompilasi	✓	
2.	Program berhasil *running*	✓	
3.	Program dapat menerima berkas input dan menuliskan *output*	✓	
4.	Luaran sudah benar untuk semua kasus *input*	✓	