Laporan
Tugas Kecil 2 IF2211 Strategi Algoritma
Semester II Tahun 2020/2021

Penyusunan Rencana Kuliah dengan Topological Sort (Penerapan Decrease and Conquer)

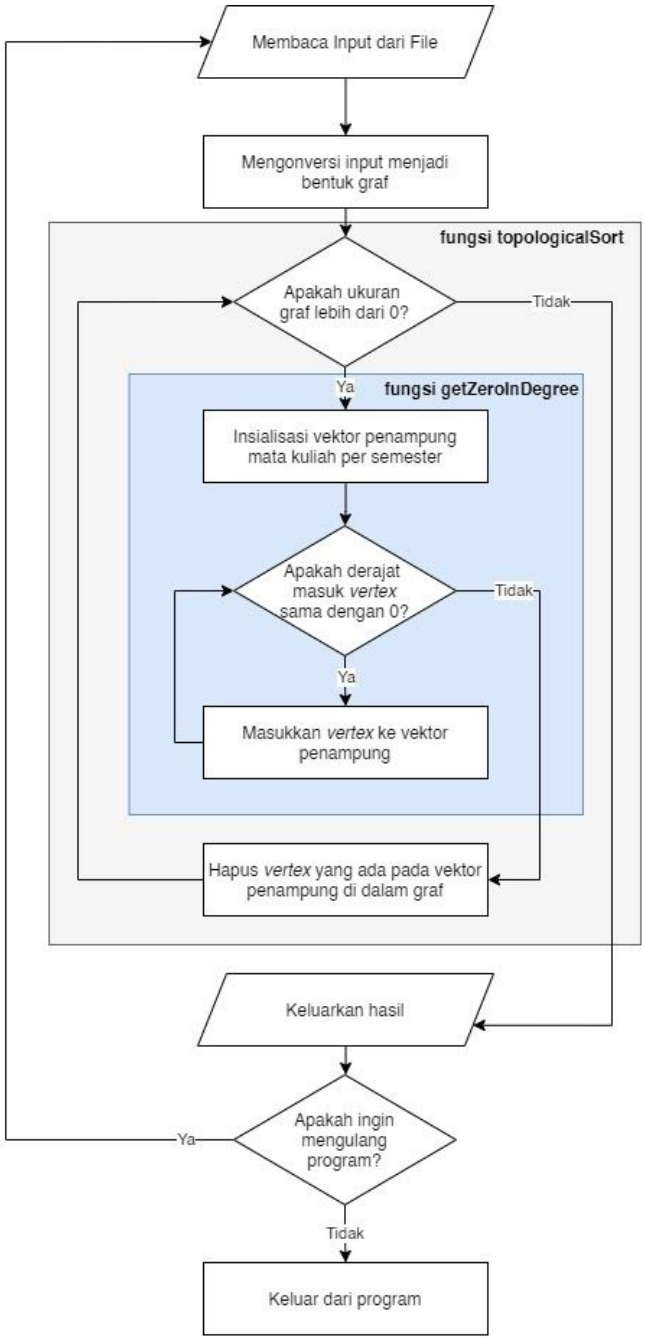
Shaffira Alya Mevia
13519083

Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung
2021

1. Algortima Topological Sort dengan Penerapan Strategi Decrease and Conquer

Algoritma Topological Sort adalah algoritma untuk *sorting* untuk graf berarah dimana pada *edge* uv, *vertex* u terurut lebih dahulu daripada *vertex* v. Pengurutan secara topologis tidak dapat dilakukan apabila bukan graf berarah.

Pada tugas kecil kali ini program bekerja dengan mengimplementasikan strategi *decrease and conquer* dalam penerapannya. Untuk memperjelas cara kerja program, berikut adalah *flowchart* yang merepresentasikan alur program.



Gambar 1.1 *Flowchart* cara kerja Rencanain

Berikut adalah penjelasan langkah-langkah yang digunakan untuk menjalankan program Rencanain.

1) Mengubah input menjadi bentuk graf

   Input awal berupa file dengan eksistensi .txt yang akan dibaca oleh program dan diubah menjadi bentuk graf yang menggunakan representasi *adjacency list*. Pada graf digunakan *abstract data type Vertex* untuk membantu kelas *Graph*.

2) Memproses graf dalam pengurutan topologis

   Berikut adalah rincian proses yang dilakukan dalam tahapan pengurutan. Tahapan ini akan berhenti ketika ukuran dari graf sudahlah menjadi nol dan akan mengeluarkan vektor yang merepresentasikan semester dengan isi mata kuliah yang masuk ke dalam semester tersebut.

   - Mencari *vertex* dengan derajat masuk nol

     Pertama dibuat dahulu vektor sementara untuk menyimpan *vertex-vertex* yang memiliki derajat masuk nol. Untuk menghitung derajat masuk, digunakan *method* `countInDegree`. Cara kerja dari *method* tersebut adalah dengan melakukan pengecekan pada graf untuk setiap *vertex* yang ada pada *adjacency list* apa bila sama dengan *vertex* yang sedang dicek maka akan ditambah jumlah derajat masuk untuk *vertex* tersebut.

   - Menghapus *vertex* dengan derajak masuk nol

     Pertama tahapan ini akan menerima vektor dengan isi *vertex-vertex* yang memiliki derajat masuk nol. Kemudian akan dihapus dengan menggunakan *method* `removeVertex`. Cara kerja dari *method* tersebut adalah dengan mengapus semua *edges* yang terhubung dengan *vertex* itu. Setelah itu, karena cara kelas graf disusun, setiap *vertex* akan diupdate nilai id-nya dan juga array graf pada kelas graf. Kemudian, karena perubahan pada id *vertex* tersebut, *vertex-vertex* pada array graf dan juga vektor *vertex*. Terakhir adalah menghapus *vertex* tersebut dari vektor *vertex*.

3) Mengeluarkan hasil pemrosesan

   Hasil dari vektor yang dikeluarkan oleh fungsi sebelumnya dan akan di keluarkan keluaran denga format yang menunjukkan semesternya dengan angka dalam angka romawi.

2. *Source Code* Program

Dalam program Rencanain terdapat beberapa file yang saling berhubungan untuk menjalankan program, berikut adalah *source code* dari program. File juga dapat dilihat pada repository GitHub.

1. File main

Berikut adalah *source code* pada file *main_13519083.cpp*.

```cpp
#include<iostream>
#include<limits> // try catch
#include <conio.h> // getch

#include "Graph_13519083.cpp" // Graph Class
#include "filesProcessing_13519083.cpp" // filesToGraph
#include "topologicalSort_13519083.cpp" // topologicalSort, printTopoResult
using namespace std;

void run();
int handleExit();

int main() {
    int pilihan;
    cout << " ___                                _          " << endl;
    cout << "| _ \\___ _ _  __ __ _ _ _  __ _(_)_ _      " << endl;
    cout << "|   / -_) ' \\/ _/ _` | ' \\/ _` | | ' \\  " << endl;
    cout << "|_|_\\___|_||_\\__\\__,_|_||_\\__,_|_|_||_|" << endl;
    cout << "Selamat datang di Rencanain! Dimana kamu bisa tahu mata kuliah
tiap semesternya" << endl;
    cout << "berdasarkan pilihan mata kuliah kamu dengan prerequisites-nya!" <<
endl;
    cout << endl;

    run();
    cout << endl;
    pilihan = handleExit();
    while (pilihan != 2) {
        if (pilihan == 1) {
            run();
            cout << endl;
            pilihan = handleExit();
        }
    }

    cout << "Terima kasih sudah menggunakan Rencanain! Semoga lulus kuliahnya
:)" << endl;
    cout << "Silahkan tekan key apapun untuk keluar." << endl;
    getch();

    return 0;
}

void run() {
    cout << "Apabila file berada di dalam folder maka bisa menggunakan relative
```

```
path, misalnya '../test/test8.txt'" << endl;
    cout << "Masukkan nama file : ";
    string filename; cin >> filename;

    Graph myGraph = filesToGraph(filename);

    vector<vector<Vertex>> result = topologicalSort(myGraph);

    cout << "Rencana Studi Semestermu adalah..." << endl;
    printTopoResult(result);
}

int handleExit() {
    int pilihan; bool pilihanBenar = false;
    while (!pilihanBenar){
        cout << "Apakah kamu masih ingin menggunakan Rencanain? (Masukkan
angkanya ya!)" << endl;
        cout << "1. ya" << endl;
        cout << "2. keluar" << endl;
        cout << "Pilihan: " << endl;
        cin >> pilihan;
        cout << endl;

        if (cin.fail()) {
            cout << "Wah, mohon masukkan angka saja ya! Mohon coba lagi." <<
endl;
            cin.clear();
            cin.ignore(std::numeric_limits<std::streamsize>::max(), '\n');

        } else if (pilihan < 1 || pilihan > 2) {
            cout << "Ups, sepertinya anda tidak memasukkan pilihan yang tepat.
Mohon coba lagi." << endl;
            cin.clear();
            cin.ignore(std::numeric_limits<std::streamsize>::max(), '\n');

        } else {
            pilihanBenar = true;
        }
    }

    return pilihan;
}
```

2. **File graph**
   Berikut adalah file realisasi dan *header* dari kelas graf.

File graph_13519083.h

```
#ifndef GRAPH_13519083_H
#define GRAPH_13519083_H

#include<vector> // Vector
#include<string> // String
using namespace std;

/*
    < Vertex Abstract Data Type >
    Implemented to make array processing easier
*/
struct Vertex {
    int id;
    string content;

    bool operator==(const Vertex& rhs) {
        if ((this->id == rhs.id) && (this->content.compare(rhs.content) == 0))
{
            return true;
        } else {
            return false;
        }
    }
};

int getVertexId(vector<Vertex> v, string content) {
    int temp;
    for(auto& v : v) {
        if (v.content.compare(content) == 0) {
            temp = v.id;
            break;
        }
    }
    return temp;
}

Vertex findVertexById(vector<Vertex> v, int id) {
    Vertex temp;
    for(auto& v : v) {
        if (v.id == id) {
            temp = v;
            break;
        }
    }
```

```
    return temp;
}

bool isContentSame(Vertex v1, Vertex v2) {
    if (v1.content.compare(v2.content) == 0) {
        return true;
    } else {
        return false;
    }
}

/*
    < Graph Class >
    Implemented using the adjacency list representation
    and vector for the realization
*/
class Graph {
    public:
        Graph(int vertrices);
        Graph(const Graph& other);
        ~Graph();

        int getVertices();
        void setVertices(int vertices);
        vector<Vertex>* getGraph();
        vector<Vertex> getVertex();

        void addEdge(Vertex source, Vertex destination);
        void removeEdge(Vertex source, Vertex destination);
        void addVertex(Vertex vertex);
        void removeVertex(Vertex vertex);
        Vertex findVertex(int id);
        int countInDegree(Vertex vertex);
        void printGraph();
        void printVertex();
    private:
        int nVertices;
        vector<Vertex>* graph;
        vector<Vertex> vertex;
};

#endif // !GRAPH_13519083_H
```

File graph_13519083.cpp

```cpp
#include <iostream>
#include<vector> // Vector
#include<string> // String
#include <algorithm> // Find

#include "Graph_13519083.h"
using namespace std;

// Constructor
Graph::Graph(int vertrices) {
    this->nVertices = vertrices;
    this->graph = new vector<Vertex>[vertrices];
}

// Copy Constructor
Graph::Graph(const Graph& other) {
    this->nVertices = other.nVertices;
    this->graph = new vector<Vertex>[this->nVertices];
    for(int i = 0; i < this->nVertices; i++) {
        for(auto& v : other.graph[i]) {
            this->graph[i].push_back(v);
        }
    }

    for(auto& v : other.vertex) {
        this->vertex.push_back(v);
    }
}

// Destructor
Graph::~Graph() {
    delete[] graph;
}

// Getter - Setter
int Graph::getVertices() {
    return this->nVertices;
}

void Graph::setVertices(int vertices) {
    this->nVertices = vertices;
}

vector<Vertex>* Graph::getGraph() {
    return this->graph;
```

```
}

vector<Vertex> Graph::getVertex() {
    return this->vertex;
}

// Methods
void Graph::addEdge(Vertex source, Vertex destination) {
    this->graph[source.id].push_back(destination);
}

void Graph::removeEdge(Vertex source, Vertex destination) {
    for (int i = 0; i < this->graph[getVertexId(this->vertex,
source.content)].size(); i++) {
        if(isContentSame(this->graph[getVertexId(this->vertex,
source.content)][i], destination)) {
            this->graph[getVertexId(this->vertex, source.content)].erase(
                this->graph[getVertexId(this->vertex, source.content)].begin()
+ i
            );
            break;
        }
    }
}

void Graph::addVertex(Vertex vertex) {
    this->vertex.push_back(vertex);
}

void Graph::removeVertex(Vertex vertex) {
    // Remove All Edges
    for (int i = 0; i < this->nVertices; i++) {
        for(int j = 0; j < this->graph[i].size(); j++) {
            Vertex v = this->graph[i][j];
            if (isContentSame(vertex, v)){
                this->removeEdge(this->findVertex(i), vertex);
            } else {
                this->removeEdge(vertex, this->findVertex(i));
            }
        }
    }

    // Update Vertex Id and Graph
    for (int i = getVertexId(this->vertex, vertex.content); i <
this->nVertices; i++) {
```

```
        if (this->vertex[i].id > getVertexId(this->vertex, vertex.content)){
            this->vertex[i].id--;
        }
        if (i+1 != this->nVertices) {
            this->graph[i] = this->graph[i+1];
        }
    };

    // Update edges
    for (int i = 0; i < this->nVertices; i++) {
        for(auto& e : this->graph[i]) {
            for(auto& v : this->vertex) {
                if (isContentSame(v, e)) {
                    e.id = v.id;
                }
            }
        }
    }

    // Remove from Vertex List
    int index;
    for (int i = 0; i < this->nVertices; i++) {
        if (isContentSame(this->vertex[i], vertex)){
            index = i;
        }
    }
    this->vertex.erase(this->vertex.begin() + index);
    this->nVertices--;

}

Vertex Graph::findVertex(int id) {
    Vertex found;
    for(auto v : this->vertex) {
        if (v.id == id) {
            found = v;
            break;
        }
    }
    return found;
}

int Graph::countInDegree(Vertex vertex) {
    int inDegree = 0;
    for (int i = 0; i < this->nVertices; i++) {
```

```
        for(auto v : this->graph[i]) {
            if (isContentSame(v, vertex)) {
                inDegree++;
            }
        }
    }

    return inDegree;
}

void Graph::printGraph() {
    for(int i = 0; i < this->nVertices; i++) {
        cout << "Vertex " << this->findVertex(i).content << " " << i << " :";
        for (auto& v : this->graph[i]) {
            cout << " -> " << v.content << " " << v.id;
        }
        cout << endl;
    }
}

void Graph::printVertex() {
    for(auto& v : this->vertex) {
        cout << "Vertex " << v.content << " with id " << v.id << endl;
    }
}
```

3. **File filesProcessing**

   Berikut adalah file realisasi dan *header* dari file filesProcessing.

File filesProcessing_13519083.h

```
#ifndef FILESPROCESSING_13519083_H
#define FILESPROCESSING_13519083_H

#include <iostream>
#include "Graph_13519083.h"
using namespace std;

Graph filesToGraph(string filename);
vector<Vertex> filesToVertex(string filename);
int countFileLine(string filename);

bool isCommaSpaceDot(char c);
```

```
string removeSpace(string str);

#endif // !FILESPROCESSING_13519083_H
```

File filesProcessing_13519083.cpp

```
#include <iostream>
#include<fstream> // ifstream
#include<sstream> // stringstream
#include <vector> // vector
#include<algorithm> // erase, remove_if

#include "Graph_13519083.h"
#include "filesProcessing_13519083.h"
using namespace std;

/*
    Opens a file then converts it to
    graph format
*/
Graph filesToGraph(string filename) {
    // Get Vertex Count and List
    vector<Vertex> vertexList = filesToVertex(filename);
    int vertexCount = countFileLine(filename);

    ifstream file(filename);

    // Make Graph
    Graph fileGraph(vertexCount);

    // Convert to Graph
    int id = 0; string line;
    while(getline(file, line)) {
        // Remove dots and space
        line.erase(remove(line.begin(), line.end(), '.'), line.end());
        line = removeSpace(line);

        // Seperate by commas
        vector<string> vertrices;
        stringstream ss(line);

        while(ss.good()) {
            string substr;
            getline(ss, substr, ',');
            vertrices.push_back(substr);
```

```
        }

        // Get and Add Vertex
        string firstContent = vertrices[0];
        Vertex first = findVertexById(vertexList,
                            getVertexId(vertexList, firstContent)
                        );
        fileGraph.addVertex(first);

        // Add edges
        auto it = vertrices.begin(); ++it;
        while (it != vertrices.end()) {
            Vertex temp = findVertexById(vertexList,
                            getVertexId(vertexList, *it)
                        );
            fileGraph.addEdge(temp, first);
            ++it;
        }

    }

    file.close();

    return fileGraph;
}

/*
    Opens a file then convert to
    unique vertex
*/
vector<Vertex> filesToVertex(string filename) {
    ifstream file(filename);
    vector<string> vertrexContents;

    // Get Vertex
    string line;
    while(getline(file, line)) {
        // Remove dots and space
        line.erase(remove(line.begin(), line.end(), '.'), line.end());
        line = removeSpace(line);

        // Seperate by commas
        vector<string> vertrices;
        stringstream ss(line);
```

```
        while(ss.good()) {
            string substr;
            getline(ss, substr, ',');
            vertrices.push_back(substr);
        }

        // Insert unique values to vertrexContents
        for (auto& v : vertrices) {
            if (find(vertrexContents.begin(), vertrexContents.end(), v) ==
vertrexContents.end()) {
                vertrexContents.push_back(v);
            }
        }
    }

    // Convert to Vertex
    vector<Vertex> result; int id = 0;
    for (auto& content : vertrexContents) {
        Vertex temp; temp.id = id;
        temp.content = content;
        result.push_back(temp);
        id++;
    }

    file.close();

    return result;
}

/*
    Additional Helper Functions
*/
int countFileLine(string filename) {
    ifstream file(filename);

    int lineCount = 0; string line;
    while(getline(file, line)) {
        lineCount++;
    }

    file.close();

    return lineCount;
}
```

```
bool isCommaSpaceDot(char c) {
   return c == ' ' || c == ',' || c == '.';
}

string removeSpace(string str) {
    string result = ""; char last = ' ';
    for (unsigned int i = 0; i < str.length(); i++) {
        if (str[i] != ' ' ||
            (!isCommaSpaceDot(last) &&
            i < str.length()-1 && !isCommaSpaceDot(str[i+1])))
        {
            result += str[i];
            last = str[i];
        }
    }
    return result;
}
```

4. **File topologicalSort**

   Berikut adalah file realisasi dan *header* dari file topologicalSort.

File topologicalSort_13519083.h

```
#ifndef TOPOLOGICALSORT_13519083_H
#define TOPOLOGICALSORT_13519083_H

#include "Graph_13519083.h"

vector<vector<Vertex>> topologicalSort(Graph graph);
vector<Vertex> getZeroInDegree(Graph graph);
void printTopoResult(vector<vector<Vertex>> topoResult);
string convertIntToRoman(int value);

#endif // !TOPOLOGICALSORT_13519083_H
```

File topologicalSort_13519083.cpp

```
#include<iostream>
#include<string> // string
#include<iterator> // size
#include<vector> // vector

#include "topologicalSort_13519083.h"
using namespace std;
```

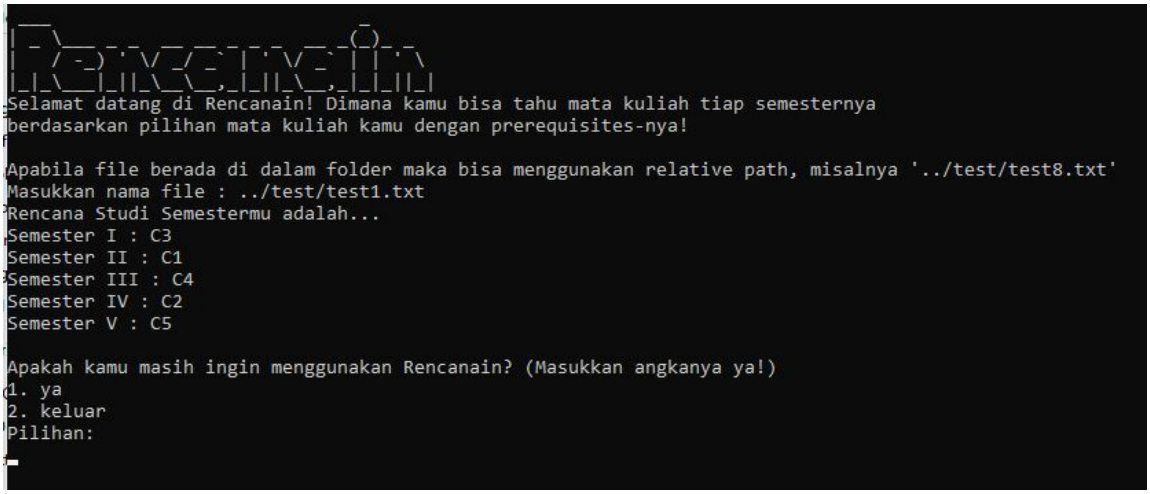
```
/*
    < Topological Sort >
    Will decreas graph size for every vertrices with zero in degree
    The resulted vector is classified as semesters
*/
vector<vector<Vertex>> topologicalSort(Graph myGraph) {
    vector<vector<Vertex>> result;
    while (myGraph.getVertices() > 0) {
        vector<Vertex> temp = getZeroInDegree(myGraph);
        result.push_back(temp);

        for (int i = 0; i < temp.size(); i++){
            myGraph.removeVertex(temp[i]);
        }

    }

    return result;
}

/*
    < Get Zero In Degree >
    Get all vetrices with zero in degree
*/
vector<Vertex> getZeroInDegree(Graph graph) {
    vector<Vertex> result;
    vector<Vertex> vertexVector = graph.getVertex();

    for (auto& v : vertexVector) {
        if (graph.countInDegree(v) == 0) {
            result.push_back(v);
        }
    }

    return result;
}

/*
    < Print Topological Search Result >
    Print result with the semester format
*/
void printTopoResult(vector<vector<Vertex>> topoResult) {
```

```
    for (int semester = 0; semester < topoResult.size(); ++semester) {
        cout << "Semester " << convertIntToRoman(semester+1) << " : ";
        for (int course = 0; course < topoResult[semester].size();
++course) {
            if (course != topoResult[semester].size()-1) {
                cout << topoResult[semester][course].content << ", ";
            } else {
                cout << topoResult[semester][course].content << endl;
            }
        }
    }
}

/*
    < Convert Integer to Roman >
    Convert numeric form to roman numerals
*/
string convertIntToRoman(int value) {
    vector<string> romans = {"M", "CM", "D", "CD", "C", "XC", "L", "XL",
"X", "IX", "V", "IV", "I"};
    vector<int> integer = {1000, 900, 500, 400, 100, 90, 50, 40, 10, 9, 5,
4, 1};

    string result = "";
    for (auto i = 0; i < integer.size(); i++) {
        while (value - integer[i] >= 0) {
            result+= romans[i];
            value-= integer[i];
        }
    }
    return result;
}
```

3. Hasil Program

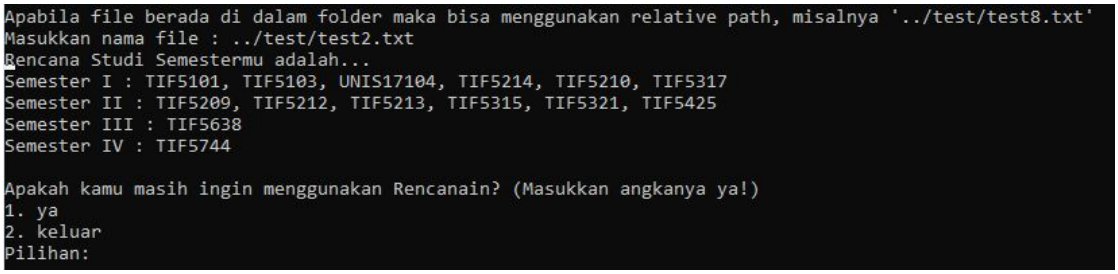
Berikut adalah beberapa hasil test untuk program Rencanain. Untuk file test terdapat pada [repository GitHub] program.

1. Test Case 1



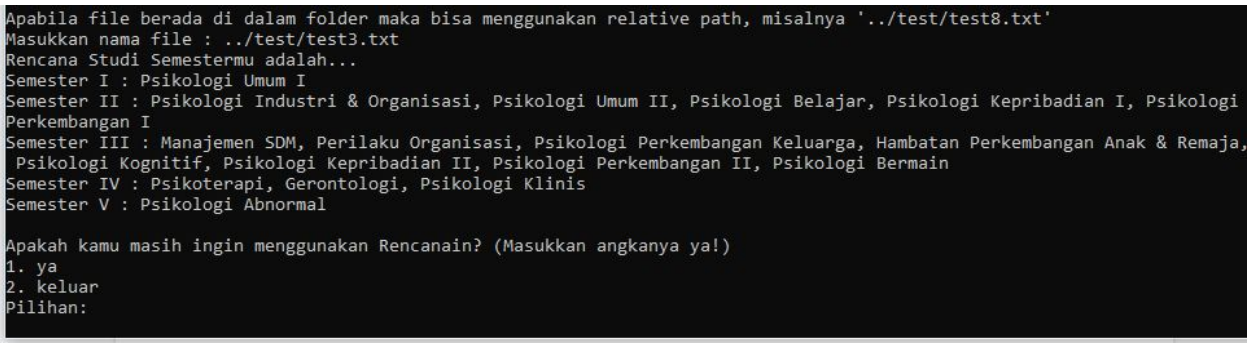
Gambar 3.1 Hasil Keluaran Test Case 1

2. Test Case 2



Gambar 3.2 Hasil Keluaran Test Case 2

3. Test Case 3



Gambar 3.3 Hasil Keluaran Test Case 3

4. Test Case 4

```
Apabila file berada di dalam folder maka bisa menggunakan relative path, misalnya '../test/test8.txt'
Masukkan nama file : ../test/test4.txt
Rencana Studi Semestermu adalah...
Semester I : Fisika Dasar, Statistika Elementer, Kalkulus 1, Bahasa Inggris, Logika, Pemecahan Masalah Mat, Pemrograman
Komputer, Aljabar Matriks, Estetika, Agama Katolik/Fenomonologi Agama, Etika
Semester II : Kalkulus 2, Matematika Diskrit, Komputasi Statistika, Aljabar Linear
Semester III : Kalkulus Vektor, Teori Peluang, Teori Suku Bunga, Metoda Matematika, Persamaan Differensial Biasa
Semester IV : Komputasi Matematika, Optimasi, Statistika Matematika, Fungsi Kompleks, Proses Stokastik
Semester V : Metoda Numerik, Analisis Real, Pemodelan Matematika

Apakah kamu masih ingin menggunakan Rencanain? (Masukkan angkanya ya!)
1. ya
2. keluar
Pilihan:
```

Gambar 3.4 Hasil Keluaran Test Case 4

5. Test Case 5

```
Apabila file berada di dalam folder maka bisa menggunakan relative path, misalnya '../test/test8.txt'
Masukkan nama file : ../test/test5.txt
Rencana Studi Semestermu adalah...
Semester I : Bahasa Inggris, Matematika I, Pendidikan Kewarganegaraan, Pengantar Akuntansi I, Pengantar Bisnis, Penganta
r Ekonomi Mikro, Praktikum Aplikasi Komputer, Sertifikasi I, Studi Islam 1, Bahasa Inggris Ekonomi, Pengantar Ekonomi Ma
kro, Pengantar Manajemen, Ilmu Kealaman Dasar, Sertifikasi Bahasa Inggris I, Sertifikasi II, Studi Islam 2
Semester II : Matematika II, Pengantar Akuntansi II, Statistika I, Teori Ekonomi Mikro, Ekonomi Moneter, Ekonomi Pembang
unan, Teori Ekonomi Makro
Semester III : Ekonomi Sumber Daya Alam dan Lingkungan, Statistika II, Akuntansi Sektor Publik, Ekonomi Internasional, E
konomi Internasional, Ekonomi Publik, Ekonomi Publik, Ekonomi Sumber Daya Manusia, Ekonomi Sumber Daya Manusia
Semester IV : Ekonometrika I

Apakah kamu masih ingin menggunakan Rencanain? (Masukkan angkanya ya!)
1. ya
2. keluar
Pilihan:
```

Gambar 3.5 Hasil Keluaran Test Case 5

6. Test Case 6

```
Apabila file berada di dalam folder maka bisa menggunakan relative path, misalnya '../test/test8.txt'
Masukkan nama file : ../test/test6.txt
Rencana Studi Semestermu adalah...
Semester I : Kalkulus Diferensial, Geometri Bidang, Aljabar Matriks, Aljabar Linear
Semester II : Kalkulus Integral, Geometri Ruang, Geometri Analitik Bidang, Program Linear
Semester III : Kalkulus Peubah Banyak, Analisis Vektor, Geometri Analitik Ruang

Apakah kamu masih ingin menggunakan Rencanain? (Masukkan angkanya ya!)
1. ya
2. keluar
Pilihan:
```

Gambar 3.6 Hasil Keluaran Test Case 6

7. Test Case 7

```
Apabila file berada di dalam folder maka bisa menggunakan relative path, misalnya '../test/test8.txt'
Masukkan nama file : ../test/test7.txt
Rencana Studi Semestermu adalah...
Semester I : h
Semester II : g
Semester III : f
Semester IV : e
Semester V : d
Semester VI : c
Semester VII : b
Semester VIII : a

Apakah kamu masih ingin menggunakan Rencanain? (Masukkan angkanya ya!)
1. ya
2. keluar
Pilihan:
```

Gambar 3.7 Hasil Keluaran Test Case 7

8. Test Case 8

```
Apabila file berada di dalam folder maka bisa menggunakan relative path, misalnya '../test/test8.txt'
Masukkan nama file : ../test/test8.txt
Rencana Studi Semestermu adalah...
Semester I : KU206X Agama dan Etika, XXLING Mata Kuliah Wajib Lingkungan, IF4091 Tugas Akhir 1 dan Seminar, IF3280 Socio
-informatika dan Profesionalisme, IF3140 Manajemen Basis Data, IF2211 Strategi Algoritma, IF2230 Sistem Operasi, IF2240
Basis Data, IF2250 Rekayasa Perangkat Lunak, IF2121 Logika Komputasional, IF2110 Algoritma & Struktur Data, IF2120 Matem
atika Diskrit, IF2130 Organisasi dan Arsitektur Komputer, MA1201 Matematika IIA, FI1201 Fisika Dasar IIA, IF1210 Dasar P
emrograman, KU1202 Pengantar Rekayasa dan Desain, KI1002 Kimia Dasar B, MA1101 Matematika IA, FI1101 Fisika Dasar IA, KU
1001 Olah Raga, KU1102 Pengenalan Komputasi, KU1011 Tata Tulis Karya Ilmiah, KU1024 Bahasa Inggris
Semester II : IF4092 Tugas Akhir 2, IF4090 Kerja Praktek, IF3210 Pengembangan Aplikasi pada Platform Khusus, IF3150 Mana
jemen Proyek Perangkat Lunak, IF3151 Interaksi Manusia Komputer, IF2210 Pemrograman Berorientasi Objek, IF2123 Aljabar L
inier dan Geometri, EL1200 Pengantar Analisis Rangkaian
Semester III : IF3250 Proyek Perangkat Lunak, IF3260 Grafika Komputer, IF3130 Jaringan Komputer, IF3141 Sistem Informasi
, IF2220 Probabilitas dan Statistika, IF2124 Teori Bahasa Formal dan Otomata
Semester IV : IF3230 Sistem Paralel dan Terdistribusi, IF3170 Inteligensi Buatan, IF3110 Pengembangan Aplikasi Berbasis
Web
Semester V : IF3270 Pembelajaran Mesin

Apakah kamu masih ingin menggunakan Rencanain? (Masukkan angkanya ya!)
1. ya
2. keluar
Pilihan:
```

Gambar 3.8 Hasil Keluaran Test Case 8

Kemudian berikut juga adalah tabel berisi *checklist* untuk penilaian.

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	✔	