Student Name : TAN ZHI LIN
Matric : 206730
Link to GitHub : https://github.com/ravenCrown0627/calculate_cgpa
Total Points (20 pts) :

Due date: 7th Nov, 2023 before 9pm

Assignment 1
ECC4802
Calculating CGPA

Analysis of Problem

To record multiple students results, the Java program should read multiple CSV files that contain information about students' results. All the CSV files had been stored at the directory named as *./csv*. The category of the CSV files can be classified as shown at the table below.

Table 1 CSV Files Classification

File name	Description
course_info.csv	This file including the course code taken by student for each semester and their corresponding credit hour. Assume that the first line is 0, the even line number is showing the course code and the odd line number is showing the course credit hour.
<student_name>_result.csv	This file including the result of each course taken by student for each semester. The first row is the name of student followed by each row of the grade value representing the result of courses taken in a semester. The test cases prepared for the program is listed below: - zhilin_result.csv - yewy_result.csv - tabina_result.csv - hasif_result.csv - shisilia_result.csv

Once the CSV files had been parsed, the student's Grade Point Average (GPA) for each semester will be calculated using the formula below as stated by the School of Graduate Studies Universiti Putra Malaysia.

$$\sum Grade\ value\ = \frac{\sum_{i=1}^{N} Grade\ value\ _i \times Credit\ hour_i}{\sum_{i=1}^{N} Credit\ hour_i}$$

$$whereby\ N = \ Total\ number\ of\ course\ taken\ in\ a\ semester$$

Figure 1 Formula to calculate student's GPA for a semester.

To determine a student's Cumulative Grade Point Average (CGPA), it can obtain by dividing the total sum of grade values for all eight semesters by the combined credit hours completed during those semesters as shown in Figure 2.

$$\sum Cumulative\ Grade\ Point\ Average\ (CGPA)\ = \frac{\sum Grade\ value}{\sum Credit\ hour}$$

Figure 2 Formula to calculate student's CGPA throughout 8 semesters.

While the GPA and CGPA for a student had been calculated, their result had been printed on console and saved as a text file in the meantime. The name of text file is in the format of *<student_name>_semester_detail_output.txt* under the directory of *./csv/output*.

The pseudo code for the program had been shown in Figure 3.

```
In function main
    Initialize student_index as 0
    If the file paths provided are type of CSV
        Parse the course code and course credit hour information
        For each of the student result file paths
            If successfully parsed the student result
                Calculate the student GPA
                Calculate the student CGPA
                Display the student's result for each semester and save as a text file
                Display the student's summarized result and save as a text file
            Else assign error code and exit the loop
    Else assign error code
    If the error code is not empty
        Display the error code on console
```

Figure 3 Pseudo code for CalculateCGPA

Flowchart

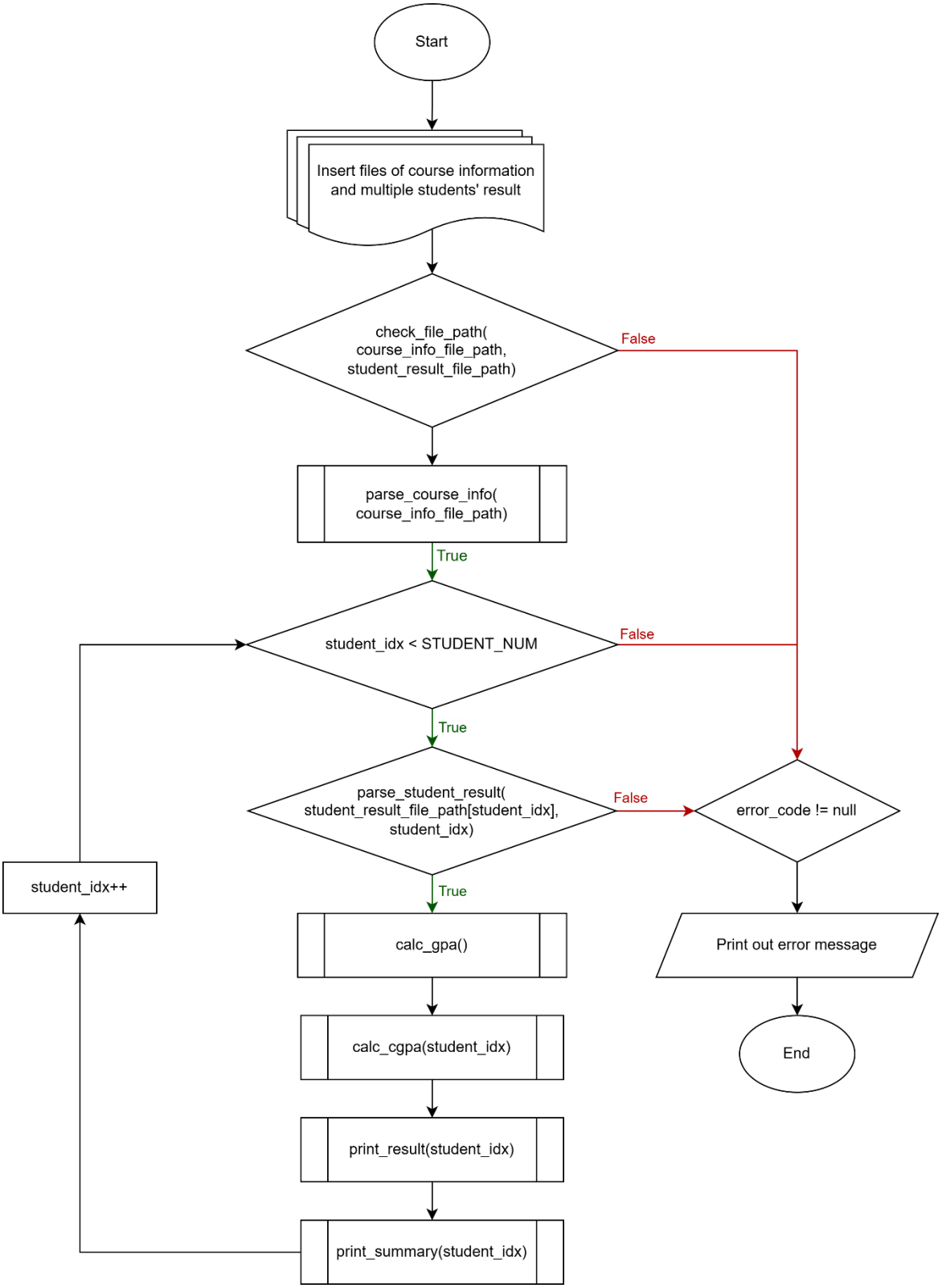

Figure 4 Flowchart for main()

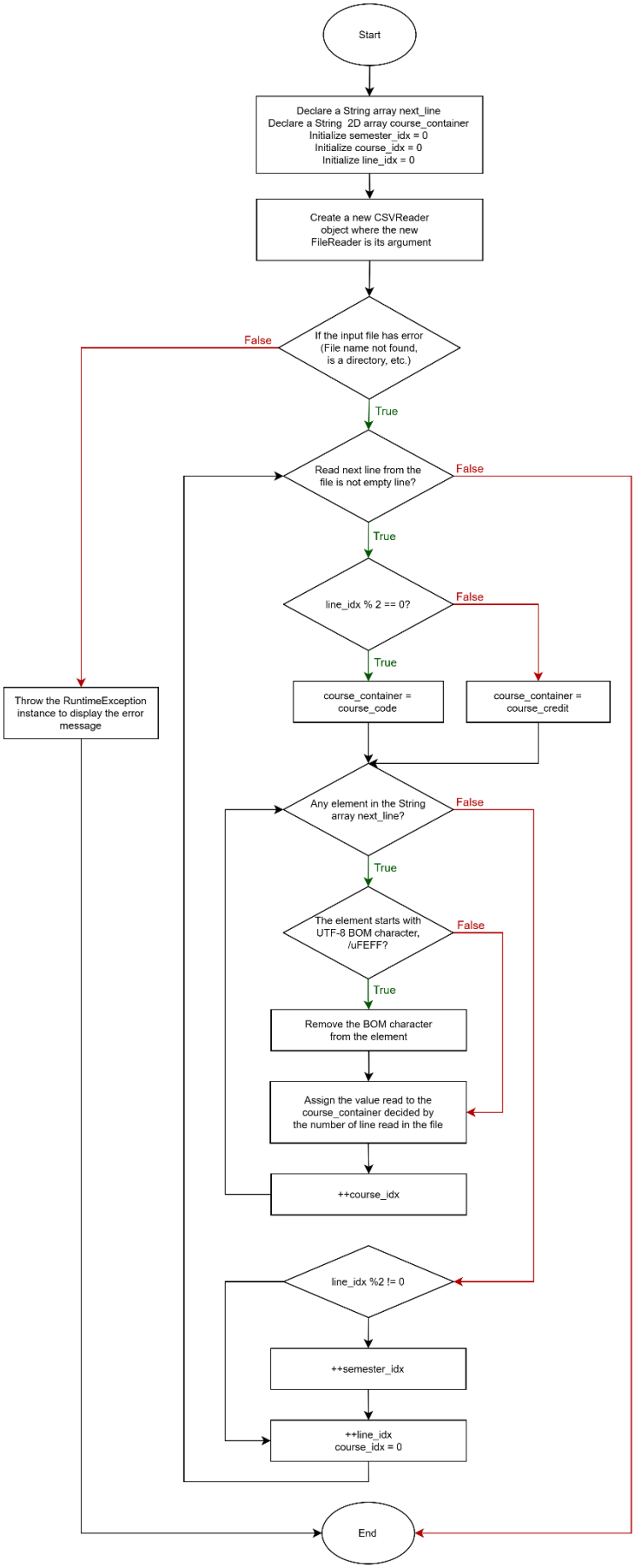

Figure 5 Flowchart for parse_course_info()

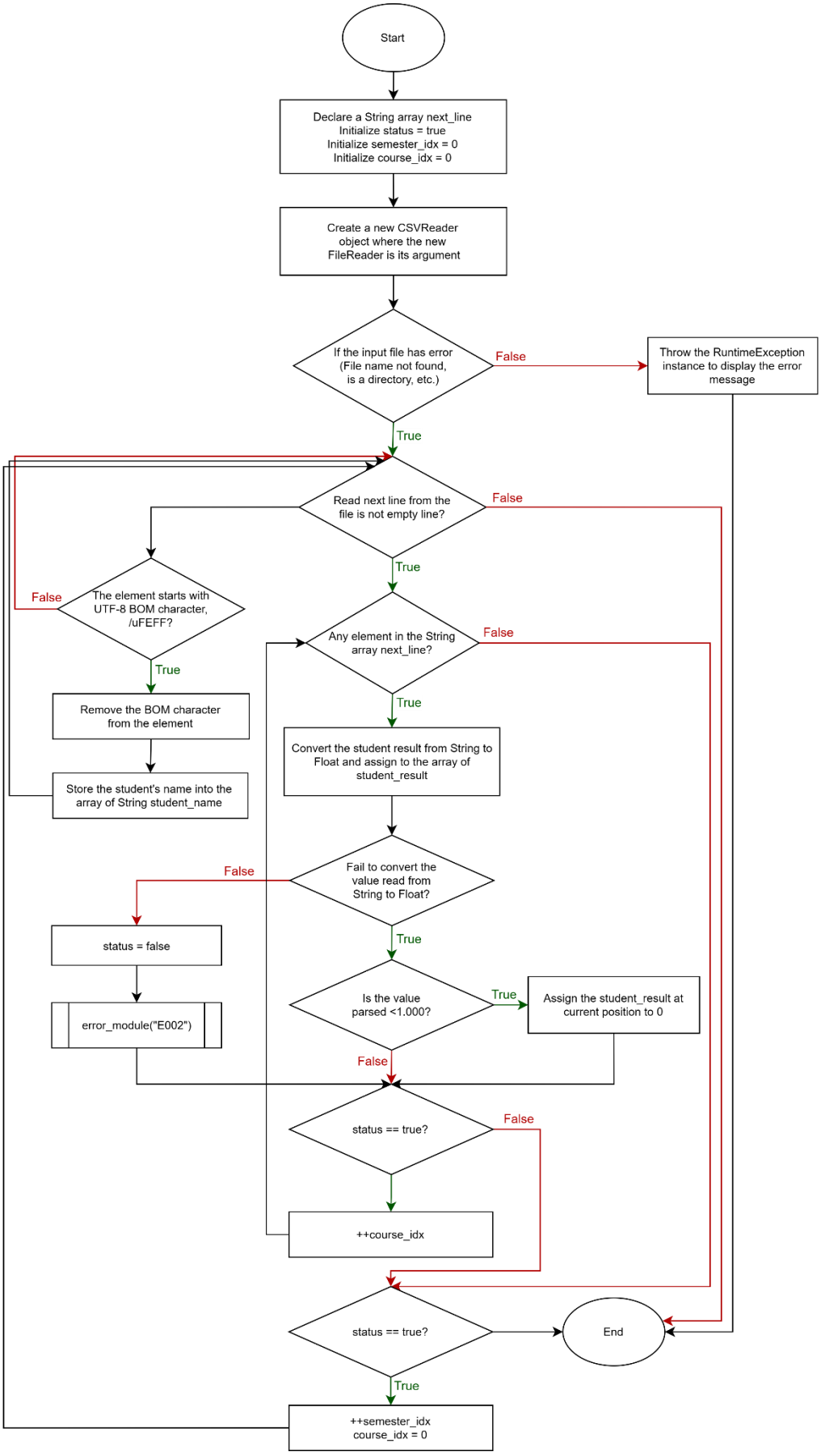

Figure 6 Flowchart for parse_student_result()

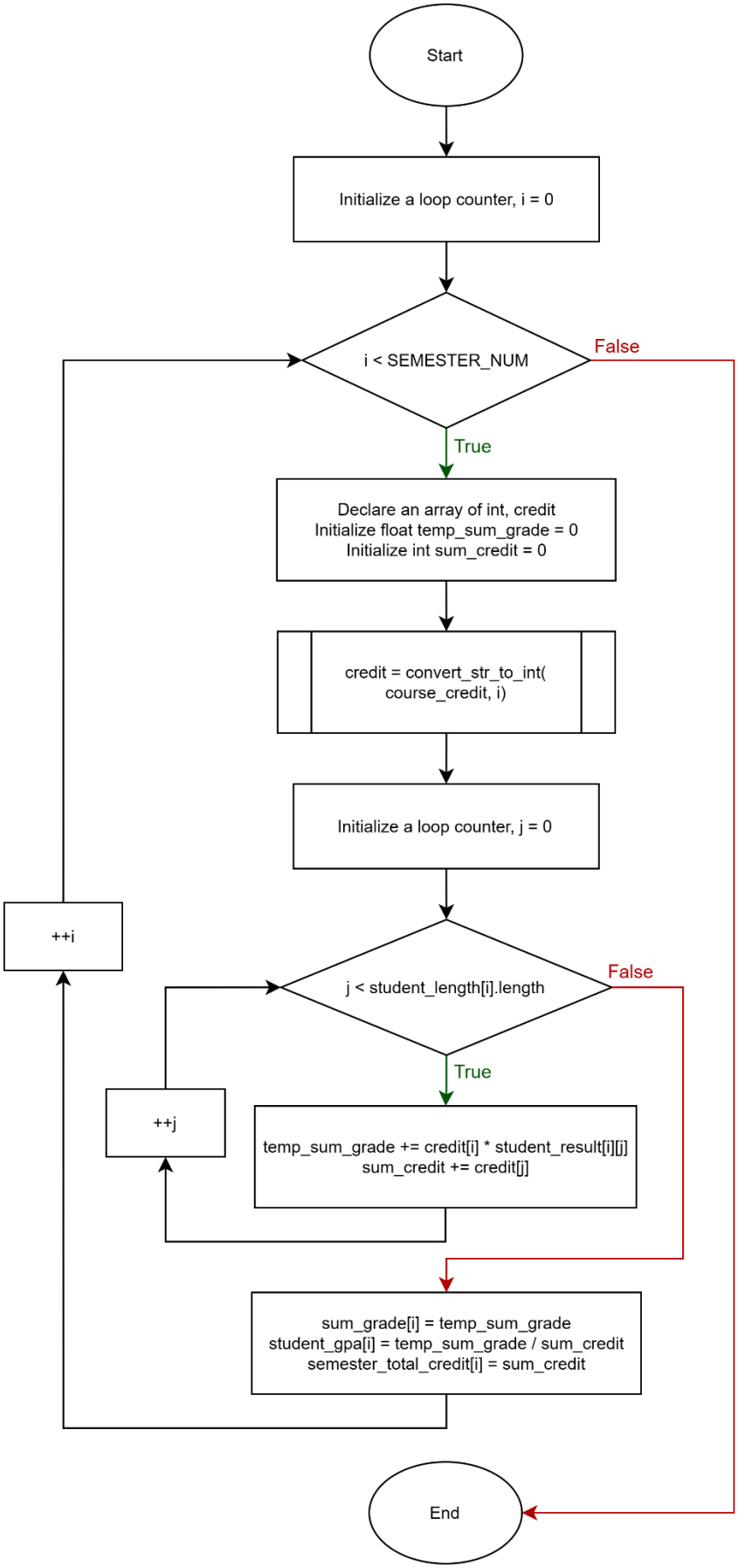

Figure 7 Flowchart for calc_gpa()

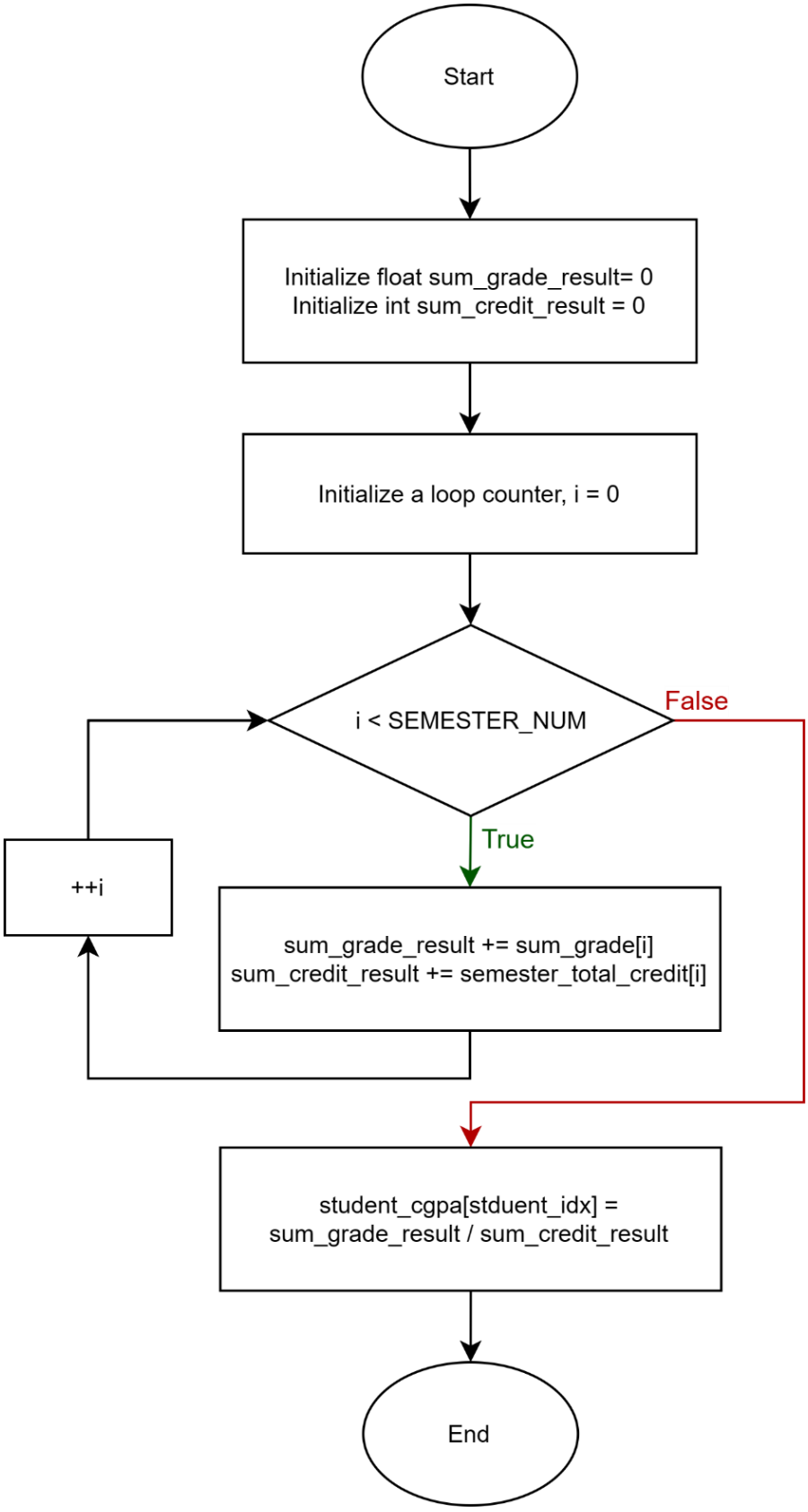

Figure 8 Flowchart for calc_cgpa()

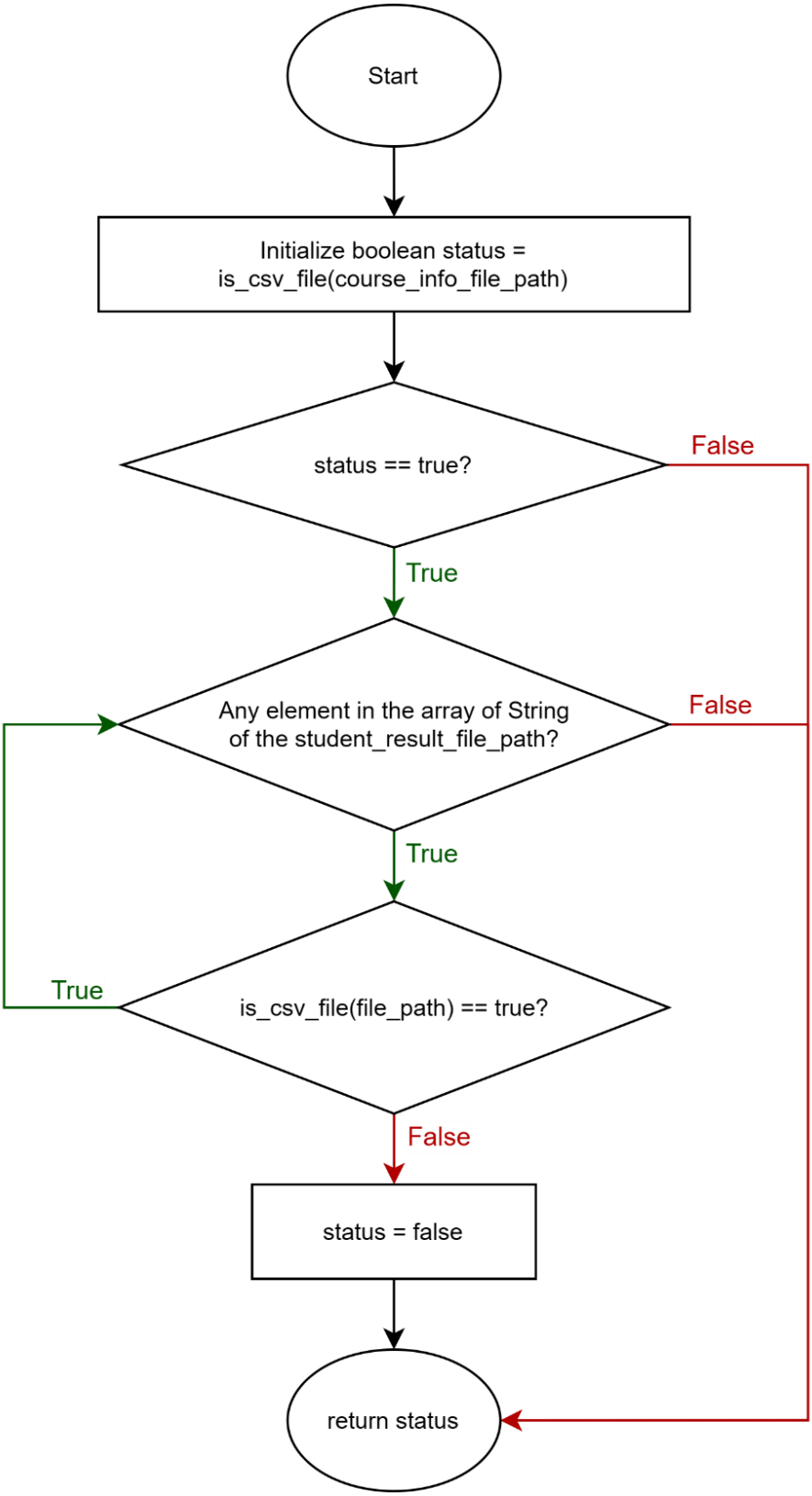

Figure 9 Flowchart for check_file_path()

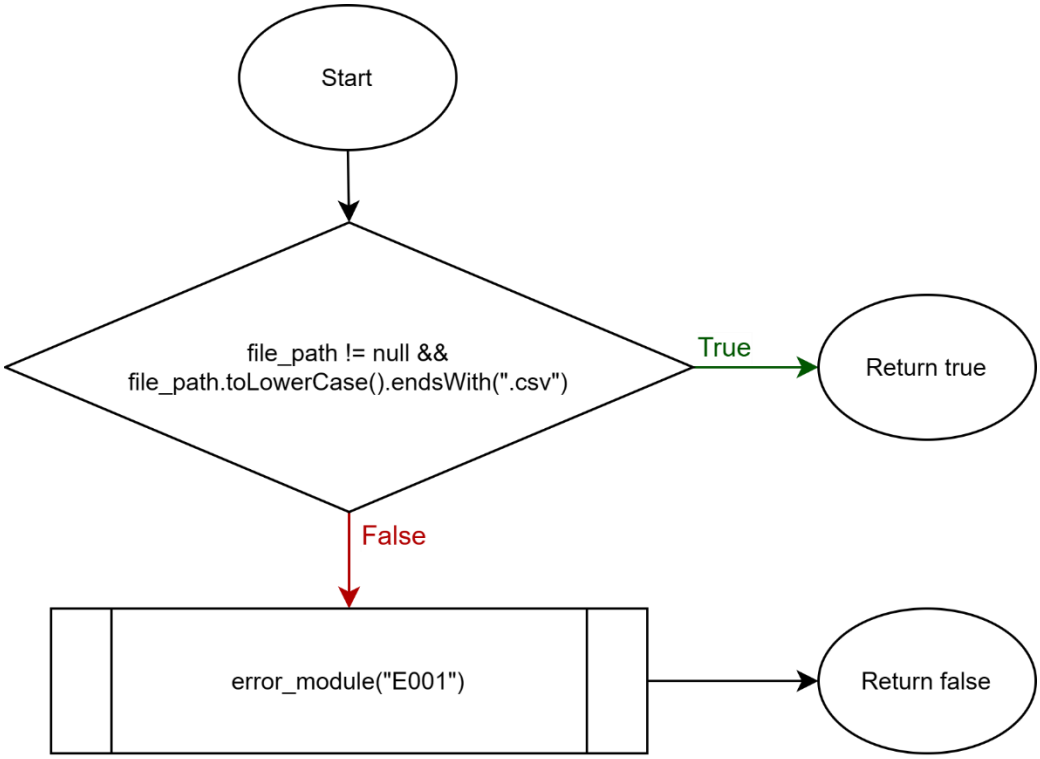

Figure 10 Flowchart for is_csv_file()

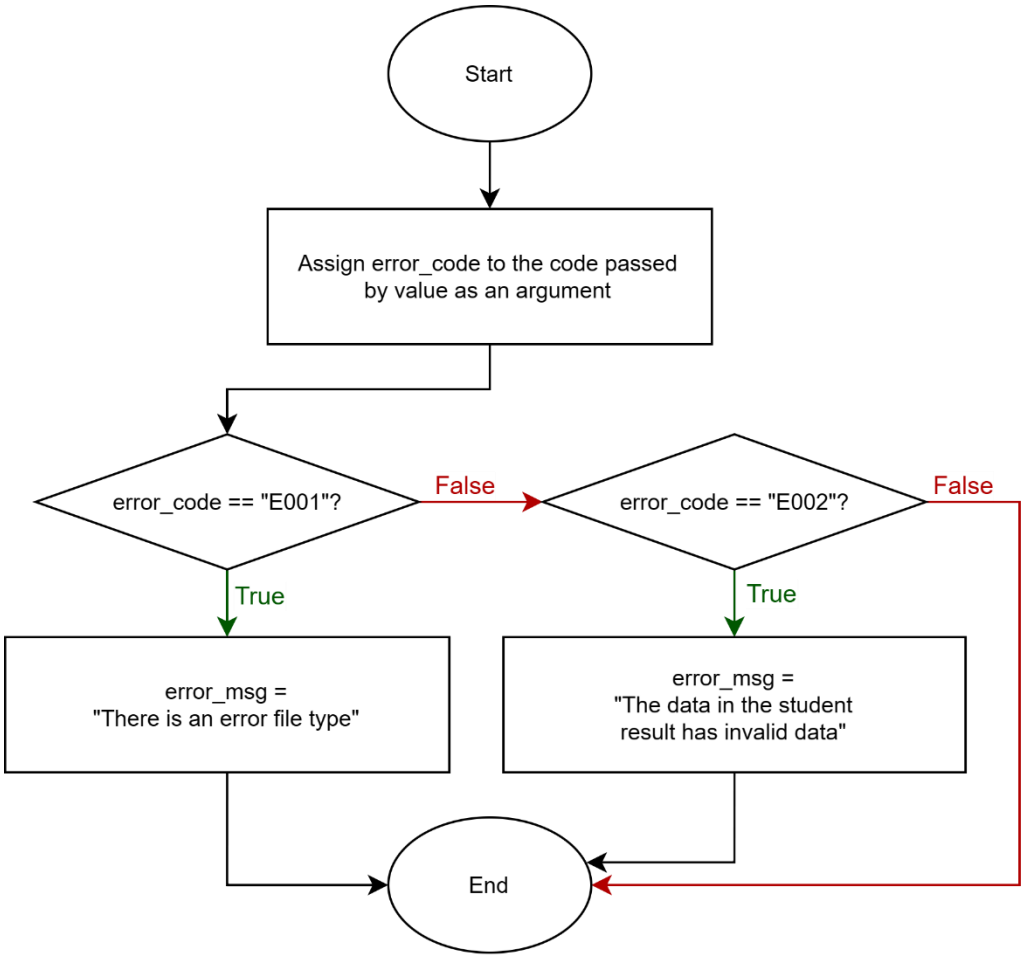

Figure 11 Flowchart for error_module()

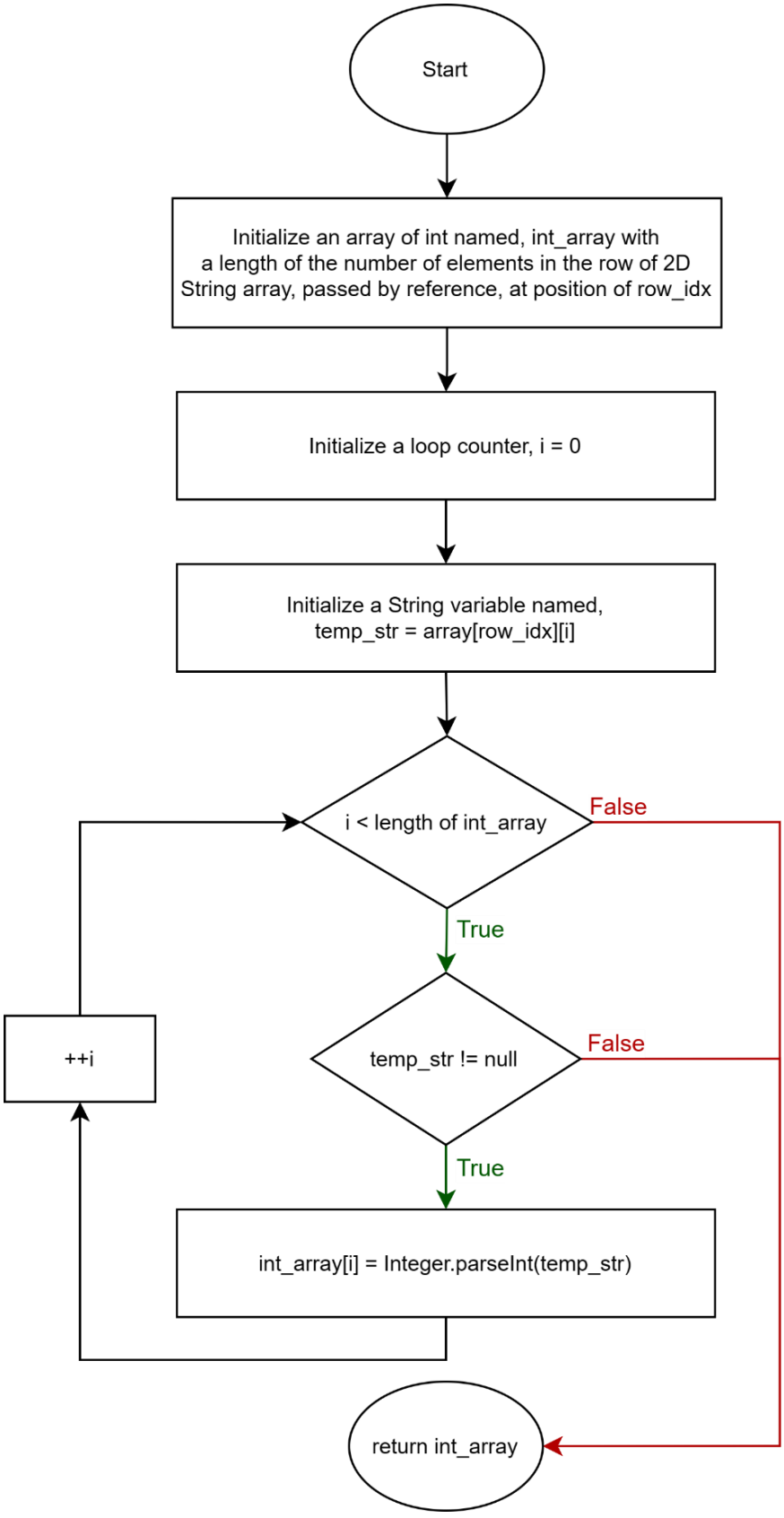

Figure 12 Flowchart for convert_str_to_int()

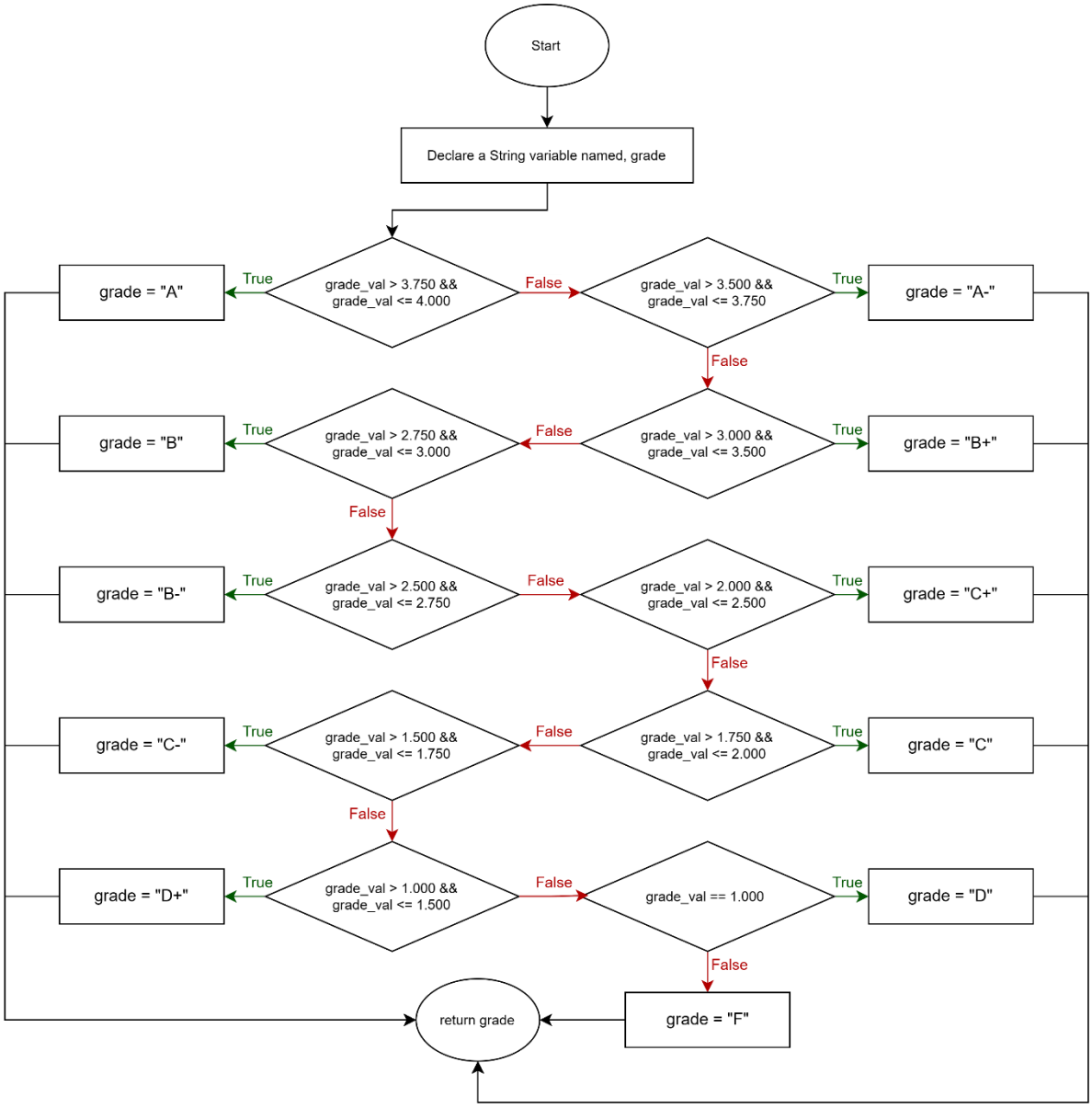

Figure 13 Flowchart for generate_grade()

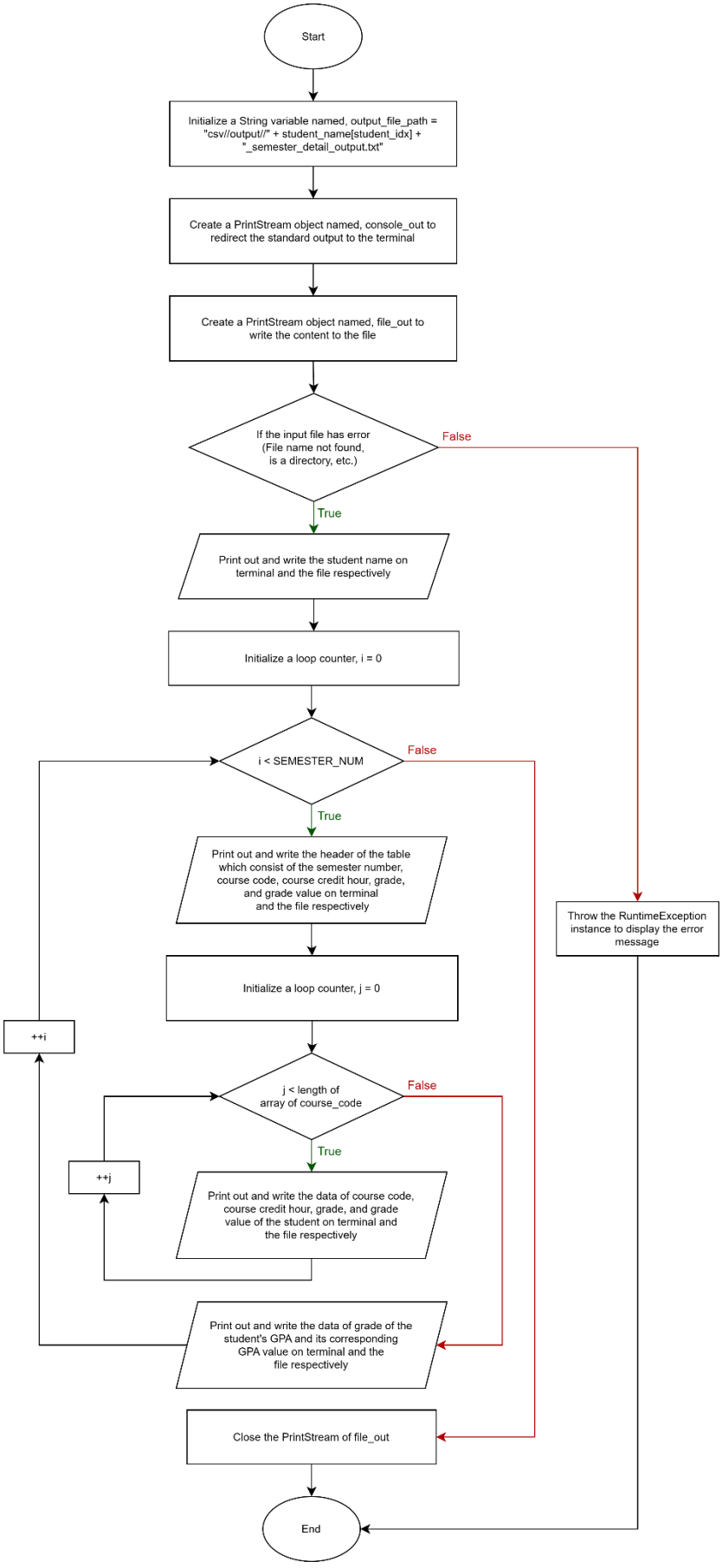

Figure 14 Flowchart for print_result()

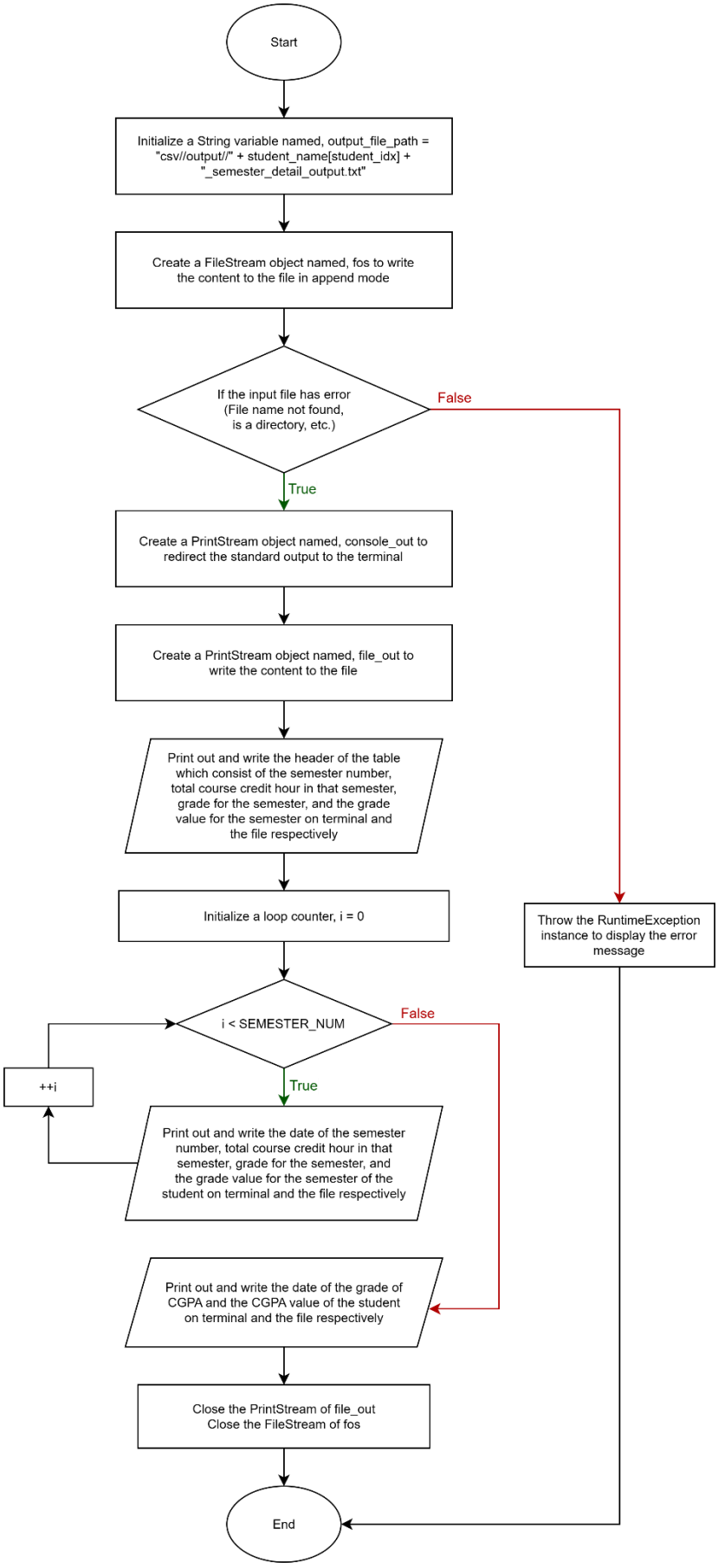

Figure 15 Flowchart for print_summary_result()

To summarize the function flowcharts stated in the figures shown above, Table 2 had stated the function name, function return type, function arguments, and the description of function.

Table 2 Summary of functions

Name	Return type	Arguments	Description
parse_course_info	void	String course_info_csv_path	Extract the information of course code and course credit hour into 2 separate arrays
parse_student_result	void	String student_result_csv_path final int student_idx	Extract the information of student name and the result of courses in all semesters
calc_gpa	void	-	Calculate the GPA of each semester for a student
calc_cgpa	void	Int student_idx	Calculate the CGPA of all semesters for a student
check_file_path	boolean	String course_info_file_path String [] student_result_file_path	To check whether the file's path given is correct
is_csv_file	boolean	String file_path	To check whether a file is CSV file
error_module	void	String code	To handle the error message used to display when expected error is occurred
convert_str_to_int	int []	String [][] str_array final int row_idx	To convert the String type element to Integer type
generate_grade	String	float grade_val	To generate the grade according to the value of result
print_result	void	final int student_idx	To display and save the table of results for all semesters on terminal and file respectively
print_summary_result	void	final int student_idx	To display and save the table of summary results for all semesters on terminal and file respectively

While developing this project, the ***opencsv*** library had been employed to facilitate the parsing of CSV files. This library has proven to be a valuable tool for efficiently reading and processing data from CSV files, simplifying the integration of external data sources into our program. By leveraging the capabilities of ***opencsv***, the program can streamline the parsing process, making it more robust and reliable. The full source code and repository for this project can be accessed via the GitHub link provided on the first page of this report. The use of ***opencsv*** not only enhances the flexibility of our program but also ensures that CSV data can be effortlessly ingested and utilized as needed.

Screenshot of Results

```
Name: Tan Zhi Lin
Semester 1
=========================================================
Course Code     Credit Hour     Grade      Grade Value
=========================================================
ECC3011         3               A-         3.750
ECC3005         3               C+         2.500
SKP3112         2               B-         2.750
SKP3122         2               C          2.000
PRT2009         2               C+         2.500
SKP2101         3               A-         3.750
QKS2102         1               B          3.000
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                        GPA:  B          2.969
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++

Semester 2
=========================================================
Course Code     Credit Hour     Grade      Grade Value
=========================================================
ECC3012         3               A-         3.750
ECC3116         4               A          4.000
LPE2301         3               A          4.000
ECC3115         4               A-         3.750
QKS2120         1               A          4.000
LPJ2102         3               A          4.000
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                        GPA:  A          3.903
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++

Semester 3
=========================================================
Course Code     Credit Hour     Grade      Grade Value
=========================================================
ECC3013         3               A          4.000
ECC3117         4               B+         3.500
ECC3304         4               B          3.000
LPE2501         3               A          4.000
ECC3112         3               B-         2.750
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                        GPA:  B+         3.426
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++

Semester 4
=========================================================
Course Code     Credit Hour     Grade      Grade Value
=========================================================
ECC3014         3               A          4.000
ECC3118         4               C-         1.750
ECC3119         4               B          3.000
ECC3120         3               A-         3.750
ECC3121         3               B-         2.750
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                        GPA:  B          2.971
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

Figure 16.1 Table result of a student from semester 1 to 4 printed at the terminal

```
Semester 5
=========================================================
Course Code     Credit Hour     Grade      Grade Value
=========================================================
ECC3113         4               A          4.000
ECC3204         3               B          3.000
ECC3403         3               B+         3.500
ECC3702         4               C+         2.500
EMM3612         3               B          3.000
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                        GPA:  B+         3.206
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++

Semester 6
=========================================================
Course Code     Credit Hour     Grade      Grade Value
=========================================================
ECC3114         3               B-         2.750
ECC3205         3               D+         1.500
ECC3303         3               B+         3.500
ECC3404         3               B-         2.750
ECC3603         4               B          3.000
ECC3904         1               A-         3.750
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                        GPA:  B          2.779
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++

Semester 7
=========================================================
Course Code     Credit Hour     Grade      Grade Value
=========================================================
ECC4911         5               A          4.000
ECC4949A        2               A          4.000
ECC4947         4               A          4.000
ECC4505         3               B          3.000
ECC4208         3               A          4.000
ECC4602         3               B-         2.750
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                        GPA:  A-         3.662
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++

Semester 8
=========================================================
Course Code     Credit Hour     Grade      Grade Value
=========================================================
ECC4949B        4               B+         3.500
ECV3011         3               C+         2.500
ECC4305         3               C+         2.500
ECC4306         3               A          4.000
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                        GPA:  B+         3.154
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

Figure 16.2 Table result of a student from semester 5 to 8 printed at the terminal

```
================================================
Semester      Credit Hour   Grade       GPA
================================================
1             16            B           2.969
2             18            A           3.903
3             17            B+          3.426
4             17            B           2.971
5             17            B+          3.206
6             17            B           2.779
7             20            A-          3.662
8             13            B+          3.154
++++++++++++++++++++++++++++++++++++++++++++++++
                    CGPA:   B+          3.278
++++++++++++++++++++++++++++++++++++++++++++++++
```

Figure 16.3 Table result of a student summarized from semester 1 to semester 8 printed at the terminal

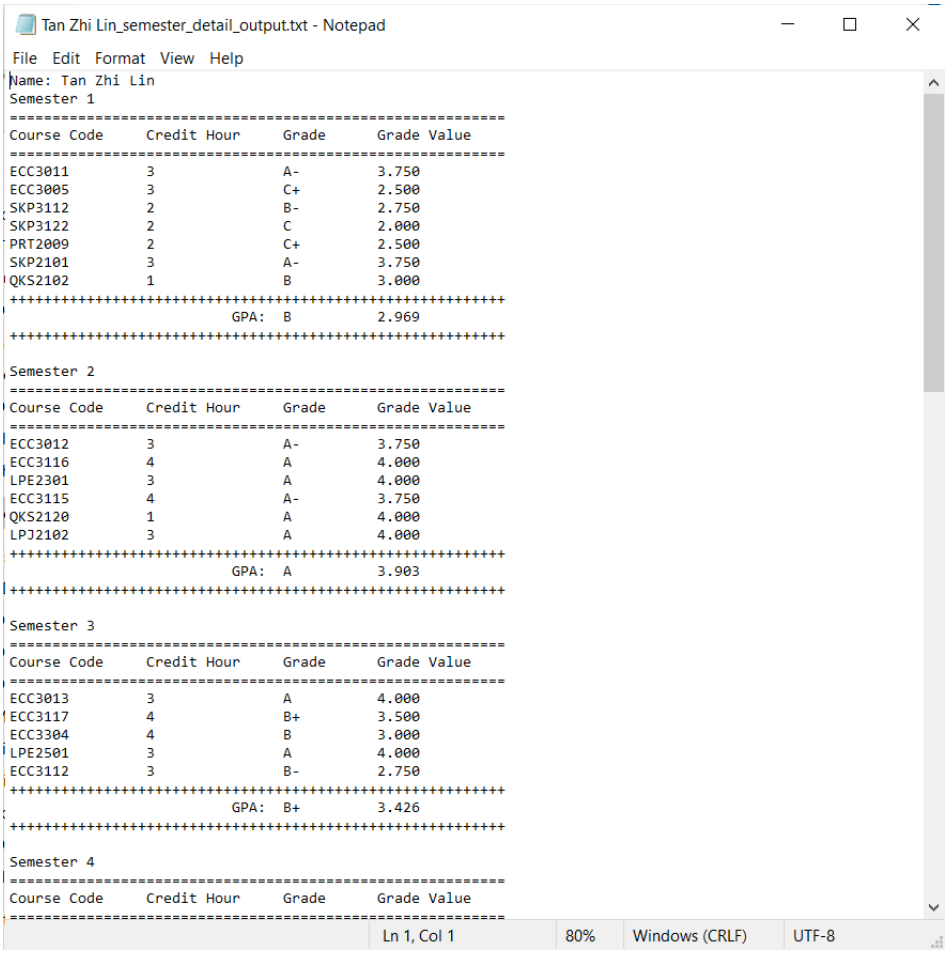

Figure 17 Table result of a student shown in terminal had been written into a text file

Source Code

```
import com.opencsv.CSVReader;
import com.opencsv.exceptions.CsvValidationException;

import java.io.*;
import java.util.Objects;

public class CalculateCGPA {
    static String error_code;
    static String error_msg;
    static final int SEMESTER_NUM   = 8;
    static final int MAX_COURSE_NUM = 10;
    static final int STUDENT_NUM    = 5;

    static String [][] course_code = new String[SEMESTER_NUM][MAX_COURSE_NUM];
    static String [][] course_credit = new String[SEMESTER_NUM][MAX_COURSE_NUM];
    static float [][] student_result = new float[SEMESTER_NUM][MAX_COURSE_NUM];
    static String [] student_name = new String[STUDENT_NUM];
    static float [] student_gpa = new float[SEMESTER_NUM];
    static float [] sum_grade = new float[SEMESTER_NUM];
    static int [] semester_total_credit = new int[SEMESTER_NUM];
    static float [] student_cgpa = new float [STUDENT_NUM];

    public static void main(String [] args) {
        String course_info_file_path = "csv\\course_info.csv";
        String [] student_result_file_path = {
                "csv\\zhilin_result.csv",
                "csv\\yewy_result.csv",
                "csv\\tabina_result.csv",
                "csv\\shisilia_result.csv",
                "csv\\hasif_result.csv"
        };

        if (check_file_path(course_info_file_path, student_result_file_path)) {
            parse_course_info(course_info_file_path);

            for (int student_idx = 0; student_idx < STUDENT_NUM; student_idx++) {
                if (parse_student_result(student_result_file_path[student_idx], student_idx)) {
                    calc_gpa();
                    calc_cgpa(student_idx);

                    print_result(student_idx);
                    print_summary_result(student_idx);
                }
                else
                    break;
            }
        }

        if (error_code != null)
            System.out.println(error_msg);
        else
            System.out.println("Success");
    }

    private static boolean is_csv_file(String file_path) {
        boolean status = file_path != null && file_path.toLowerCase().endsWith(".csv");

        if (!status)
            error_module("E001");

        return status;
    }
```

```
private static boolean check_file_path(String course_info_file_path, String []
student_result_file_path) {
        boolean status = is_csv_file(course_info_file_path);

        if (status) {
            for (String file_path : student_result_file_path) {
                if (!is_csv_file(file_path)) {
                    status = false;
                    break;
                }
            }
        }

        return status;
    }

    private static int [] convert_str_to_int(String [][] str_array, final int row_idx) {
        int [] int_array = new int[str_array[row_idx].length];

        // Convert String to int
        for (int i = 0; i < int_array.length; ++i) {
            String temp_str = str_array[row_idx][i];

            if (temp_str != null)
                int_array[i] = Integer.parseInt(temp_str);
            else
                break;
        }

        return int_array;
    }

    private static String generate_grade(float grade_val) {
        String grade;

        if (grade_val > 3.750 && grade_val <= 4.000) {
            grade = "A";
        } else if (grade_val > 3.500 && grade_val <= 3.750) {
            grade = "A-";
        } else if (grade_val > 3.000 && grade_val <= 3.500) {
            grade = "B+";
        } else if (grade_val > 2.750 && grade_val <= 3.00) {
            grade = "B";
        } else if (grade_val > 2.500 && grade_val <= 2.750) {
            grade = "B-";
        } else if (grade_val > 2.000 && grade_val <= 2.500) {
            grade = "C+";
        } else if (grade_val > 1.750 && grade_val <= 2.000) {
            grade = "C";
        } else if (grade_val > 1.500 && grade_val <= 1.750) {
            grade = "C-";
        } else if (grade_val > 1.000 && grade_val <= 1.500) {
            grade = "D+";
        } else if (grade_val == 1.000) {
            grade = "D";
        } else {
            grade = "F";
        }

        return grade;
    }

    private static void error_module(String code) {
        error_code = code;

        if (Objects.equals(error_code, "E001")) {
            error_msg = "There is an error file type";
        }
        else if (Objects.equals(error_code, "E002")) {
            error_msg = "The data in the student result has invalid data”;
        }
    }
```

```java
public static void parse_course_info(String course_info_csv_path) {
        String [] next_line;
        String [][] course_container;
        int semester_idx = 0;
        int course_idx = 0;
        int line_idx = 0;

        try (CSVReader reader = new CSVReader(new FileReader(course_info_csv_path))) {
            try {
                while ((next_line = reader.readNext()) != null) {
                    // switching the container to store the course code and course credit
interleave
                    if (line_idx % 2 == 0) {
                        course_container = course_code;
                    }
                    else {
                        course_container = course_credit;
                    }

                    // next_line is an array of values from the line
                    for (String val : next_line) {
                        if (val.startsWith("\uFEFF")) {
                            val = val.substring(1); // Remove the BOM
                        }

                        course_container[semester_idx][course_idx] = val;
                        ++course_idx;
                    }

                    if (line_idx % 2 != 0)
                        ++semester_idx;

                    ++line_idx;
                    course_idx = 0;
                }
            } catch (IOException | CsvValidationException e) {
                throw new RuntimeException(e);
            }
        } catch (IOException e) {
            throw new RuntimeException(e);
        }
    }

    public static boolean parse_student_result(String student_result_csv_path, final int
student_idx) {
        boolean status = true;
        String[] next_line;
        int semester_idx = 0;
        int course_idx = 0;

        try (CSVReader reader = new CSVReader(new FileReader(student_result_csv_path))) {
            try {
                while ((next_line = reader.readNext()) != null) {
                    // Only the first line has BOM char and contain the student name
                    if (next_line[0].startsWith("\uFEFF")) {
                        student_name[student_idx] = next_line[0].substring(1);
                        continue;
                    }

                    // next_line is an array of values from the line
                    for (String val : next_line) {
                        try {
                            student_result[semester_idx][course_idx] = Float.parseFloat(val);

                            // Modify the point to 0.0 if it is less than 1.000
                            if (student_result[semester_idx][course_idx] < 1.000)
                                student_result[semester_idx][course_idx] = 0;
                        } catch (NumberFormatException e ) {
                            status = false;
                            error_module("E002");
                        }
```

```
                    if (status) {
                        ++semester_idx;
                        course_idx = 0;
                    }
                    else
                        break;
                }
            } catch (IOException | CsvValidationException e) {
                throw new RuntimeException(e);
            }
        } catch (IOException e) {
            throw new RuntimeException(e);
        }

        return status;
    }

    public static void calc_gpa() {
        for (int i = 0; i < SEMESTER_NUM; ++i) {
            int [] credit;
            float temp_sum_grade = 0;
            int sum_credit = 0;

            credit = convert_str_to_int(course_credit, i);

            for (int j = 0; j < student_result[i].length; ++j) {
                temp_sum_grade += credit[j] * student_result[i][j];
                sum_credit += credit[j];
            }

            sum_grade[i] = temp_sum_grade;
            student_gpa[i] =  temp_sum_grade / sum_credit;
            semester_total_credit[i] = sum_credit;
        }
    }

    public static void calc_cgpa(final int student_idx) {
        float sum_grade_result = 0;
        int sum_credit_result = 0;

        for (int i = 0; i < SEMESTER_NUM; i++) {
            sum_grade_result += sum_grade[i];
            sum_credit_result += semester_total_credit[i];
        }

        student_cgpa[student_idx] = sum_grade_result / sum_credit_result;
    }

    public static void print_result(final int student_idx) {
        String output_file_path = "csv//output//" + student_name[student_idx] +
"_semester_detail_output.txt";

        try {
            // Create a FileOutputStream to write to the file
            FileOutputStream fos = new FileOutputStream(output_file_path);
            // Redirect standard output to the terminal
            PrintStream console_out = System.out;
            // Write the content to the file
            PrintStream file_out = new PrintStream(output_file_path);
            System.setOut(file_out);

            // Display on console
            console_out.println("Name: " + student_name[student_idx]);
            // Print to file
            file_out.println("Name: " + student_name[student_idx]);

            for (int i = 0; i < SEMESTER_NUM; ++i) {
                // Display on console
                console_out.println("Semester " + (i + 1));
                console_out.println("============================================================");
                console_out.printf("%-15s %-15s %-10s %-15s%n", "Course Code", "Credit Hour",
"Grade", "Grade Value");
                console_out.println("============================================================");
```

```
                for (int j = 0; j < course_code.length; ++j) {
                    if (course_code[i][j] != null) {
                        // Display on console
                        console_out.printf("%-15s %-15s %-10s %.3f%n", course_code[i][j],
course_credit[i][j], generate_grade(student_result[i][j]), student_result[i][j]);
                        // Print to file
                        file_out.printf("%-15s %-15s %-10s %.3f%n", course_code[i][j],
course_credit[i][j], generate_grade(student_result[i][j]), student_result[i][j]);
                    }
                }

                // Display on console
                console_out.println("++++++++++++++++++++++++++++++++++++++++++++++++++++++++");
                console_out.printf("%-22s    GPA:  %-10s %.3f%n", "" ,
generate_grade(student_gpa[i]), student_gpa[i]);
                console_out.println("++++++++++++++++++++++++++++++++++++++++++++++++++++++++\n"
);
                // Print to file
                file_out.println("++++++++++++++++++++++++++++++++++++++++++++++++++++++++");
                file_out.printf("%-22s    GPA:  %-10s %.3f%n", "", generate_grade(student_gpa[i]),
student_gpa[i]);
                file_out.println("++++++++++++++++++++++++++++++++++++++++++++++++++++++++\n");
            }

            // Close the file of FileStream
            fos.close();
            // Close the file of PrintStream
            file_out.close();
            // Reset the System.out ro original PrintStream for terminal
            System.setOut(console_out);
        } catch (IOException e) {
            throw new RuntimeException(e);
        }
    }

    public static void print_summary_result(final int student_idx) {
        String output_file_path = "csv//output//" + student_name[student_idx] +
"_semester_detail_output.txt";

        try {
            // Create a FileOutputStream to write to the file in append mode
            FileOutputStream fos = new FileOutputStream(output_file_path, true);

            // Redirect standard output to the file
            PrintStream console_out = System.out;
            // Write the content to the file
            PrintStream file_out = new PrintStream(fos);

            // Display on console
            console_out.println("============================================");
            console_out.printf("%-12s %-12s %-10s %-10s%n", "Semester", "Credit Hour", "Grade",
"GPA");
            console_out.println("============================================");
            // Print to file
            file_out.println("============================================");
            file_out.printf("%-12s %-12s %-10s %-10s%n", "Semester", "Credit Hour", "Grade",
"GPA");
            file_out.println("============================================");

            for (int i = 0; i < SEMESTER_NUM; i++) {
                // Display on console
                console_out.printf("%-12s %-12s %-10s %.3f%n", (i + 1), semester_total_credit[i],
generate_grade(student_gpa[i]), student_gpa[i]);
                // Print to file
                file_out.printf("%-12s %-12s %-10s %.3f%n", (i + 1), semester_total_credit[i],
generate_grade(student_gpa[i]), student_gpa[i]);
            }
```

```
            // Display on console
            console_out.println("+++++++++++++++++++++++++++++++++++++++++++++");
            console_out.printf("%-16s   CGPA:  %-10s %.3f%n", "",
generate_grade(student_cgpa[student_idx]), student_cgpa[student_idx]);
            console_out.println("+++++++++++++++++++++++++++++++++++++++++++++\n");
            // Print to file
            file_out.println("+++++++++++++++++++++++++++++++++++++++++++++");
            file_out.printf("%-16s   CGPA:  %-10s %.3f%n", "",
generate_grade(student_cgpa[student_idx]), student_cgpa[student_idx]);
            file_out.println("+++++++++++++++++++++++++++++++++++++++++++++");

            // Close the file of FileStream
            fos.close();
            // Close the file of PrintStream
            file_out.close();
            // Reset the System.out ro original PrintStream for terminal
            System.setOut(console_out);
        } catch (IOException e) {
            throw new RuntimeException(e);
        }
    }
}
```

Program Testing

In the program, the input is primarily comprised of hard-coded file paths, student information stored in an array of String, and course information stored in a String variable. Proper handling of these inputs, particularly the parsing process, plays a vital role in ensuring the robustness and reliability of the software. The exception handling had been done for the program in 2 situations, which is when the file path given is not CSV file and the program fail to parse a String value to Float value which assigned to error code *E001* and *E002* respectively. When the error is trigged, the program will halt immediately and printing the designed error message onto the terminal to give a clue for user on when the program is halt.

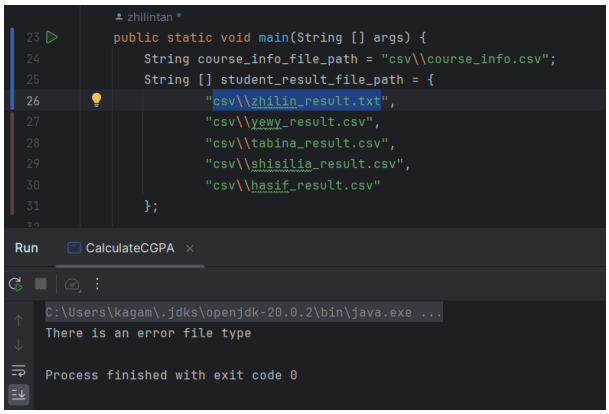

Figure 18 Error message displayed on terminal when the type of input file is not CSV

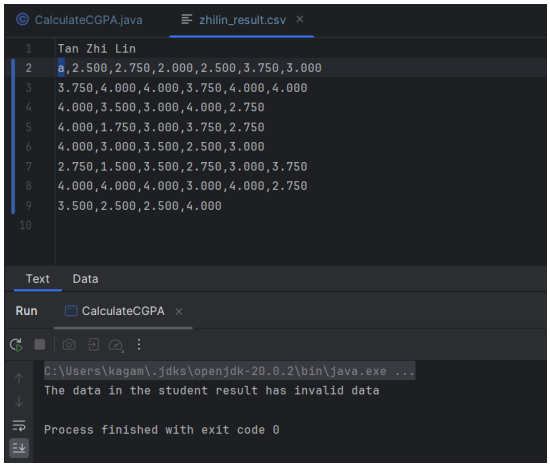

Figure 19 Error message displayed on terminal when the parser fails to convert the read value from String to Float