German International University
Faculty of Informatics and Computer Science
Dr. John Zaki
Eng. Ahmed Sherif
Amir Haythem
Nadeen Ramadan

Software Engineering, Winter 2024
Assignment

Submission: Saturday October 19th 2024 at 11:59pm

In this assignment, you are required to implement nine JavaScript functions focused on managing student, course, and grade records. Three main files will be provided: students.txt, courses.txt, and grades.txt. The primary tasks involve reading from and writing to these text files to effectively manage the data. All functions are well documented with detailed descriptions and provide expected outputs for each function, ensuring clarity and ease of use. Before starting implementation, it is essential to read the assignment several times to ensure a thorough understanding of the requirements.

Assignment Guidelines

- A Team consists of 2-3 members
- **You are not allowed to use any external library and you can use any predefined methods. External libraries will require npm installation. You are welcome to use any library that does not require npm installation. Using an external library will result in receiving a zero grade.**
- You are allowed to use any predefined methods.
- **You will be provided with an Assignment template and check the readme file inside the Assignment template.**
- Public test cases will be provided along with the assignment template.
- You need to handle all possible cases for each function as receiving full grades in public test cases doesn't ensure that you will receive full grades in private test cases
- You need to search for fs.readFileSync() and fs.writeFileSync() functions in order to read and write from a text file
- **Submit all your code inside main.js and submitting your code in different file other than main.js will result in zero grade.**
- **You will be provided with main.js. You are not allowed to remove any lines of code inside main.js; doing so will result in a zero grade. Ensure that you keep all function signatures, even if you do not have time to implement them all. It is crucial not to remove module.exports inside main.js.**
- **You are welcome to add any functions inside main.js as well as to write any number of code lines within all your functions. Additionally, you can add new functions. However, you are not allowed to test inside main.js; doing so will result in a zero grade.**
- **Do not hard code the text directory paths inside your functions. In other words, the file paths for students.txt, courses.txt, and grades.txt should be passed as parameters to your functions. Additionally, you should not write .txt inside your functions. Writing .txt inside your functions will result in zero grade.**
- **Check the readmeFile**

Example of students text File

```
ID,Name,Email,Major
10003827,Jana Elsherif,jana.elsherif@student.giu-uni.de,Mechanical Enginnering
10002859,Hussein Dawood,hussein.dawood@student.giu-uni.de,Informatics and Computer Science
10006840,Evram Joseph,evram.joseph@student.giu-uni.de,Pharmaceutical Engineering
10004867,Omar Khairy,omar.Khairy@student.giu-uni.de,Mechanical Enginnering
10006476,Arwa Omara,arwa.omara@student.giu-uni.de,Mechanical Enginnering
10003898,Haidy Hazem,haidy.hazem@student.giu-uni.de,Mechanical Enginnering
10003857,Fatma Ashraf,fatma.ashraf@student.giu-uni.de,Mechanical Enginnering
10000864,Ahmed Humoudi,ahmed.humoudi@student.giu-uni.de,Informatics and Computer Science
10003823,Marwan Khalifa,marwan.khalifa@student.giu-uni.de,Mechanical Enginnering
7004848,Tasneem Alaaeldien,tasneem.alaaeldien@student.giu-uni.de,Electrical Enginnering
7002849,Aisha Ashraf,aisha.ashraf@student.giu-uni.de,Electrical Enginnering
7036808,Nouran Mohamed,nouran.mohamed@student.giu-uni.de,Electrical Enginnering
7005838,Yassmin Hany,yassmin.hany@student.giu-uni.de,Electrical Enginnering
7001823,Youssef Elnady,youssef.elnady@student.giu-uni.de,Business Administration
7005806,Mark Moheb,mark.moheb@student.giu-uni.de,Electrical Enginnering
7036820,Khaled Yasser,khaled.yasser@student.giu-uni.de,Electrical Enginnering
7004827,Seif Osama,seif.osama@student.giu-uni.de,Business Informatics
```

Example of courses text File

```
ID,Code,Name,creditHours
11,MATH103,Mathematics I: Calculus I,8
12,CSEN104,Introduction to Computer Science: Programming I,6
13,PHYS101,Physics,5
14,CSEN103,Digital Logic Design,4
15,AS101,English for Academic Skills,4
16,DE101,German 1,4
21,MATH204,Mathematics II: Linear Algebra,4
22,CSIS201,Computer Science II: Programming II,7
23,CNET101,Computer Networks,5
24,CSIS102,Theoretical Computer Science,5
25,CSMR101,Computer Organization,5
26,SM101,Scientific Methods,2
27,DE202,German 2,4
31,MATH105,Mathematics III: Discrete Mathematics,4
32,INCS101,Programming III,7
```

Implement the function getStudentInfo(studentFile , studentID)

- studentFile : (typeof string) : directory/path of students text file
- studentID : (typeof string) : student ID

Function Description:

The function checks if a given `studentID` exists within the `studentFile` directory.

- If the `studentID` **does not exist**, the function returns an **empty object**.
- If the `studentID` **exists**, the function returns the corresponding **student object**.

Student Object has the following properties

Property	DataType	Description
id	string	student id
name	string	student name
email	string	student email
major	string	student major

For Example

```
JS testScenerio.js > ...
2   let studentFile = "students.txt";
3   let studentID = "10002859";
4   let result = getStudentInfo(studentFile , studentID);
5   console.log("test case 1",result);
6
7   studentID = "10003215";
8   result = getStudentInfo(studentFile , studentID);
9   console.log("test case 2",result);
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  powershell

PS C:\Users\Ahmed Sherif\Documents\SE-W24\Assignment> node testScenerio.js
test case 1 {
  id: '10002859',
  name: 'Hussein Dawood',
  email: 'hussein.dawood@student.giu-uni.de',
  major: 'Informatics and Computer Science'
}
test case 2 {}
```

Implement the function getCourseInfo(courseFile , courseID)

- courseFile : (typeof string) : directory/path of courses text file
- courseID : (typeof string) : course ID

Function Description:

The function checks if a given `courseID` exists within the `courseFile` directory.

- If the `courseID` **does not exist**, the function returns an **empty object**.
- If the `courseID` **exists**, the function returns the corresponding **course object**.

Course object has the following properties

Property	DataType	Description
id	string	course id
name	string	course name
code	string	course code
creditHours	number	credit hours assigned to the course

For Example

JS testScenerio.js > getCourseInfo

```
1   const {getCourseInfo} = require("./main");
2   let courseFile = "courses.txt";
3   let courseID = "16";
4   result = getCourseInfo(courseFile , courseID);
5   console.log("test case 1",result);
6
7   courseID = "17";
8   result = getCourseInfo(courseFile , courseID);
9   console.log("test case 2",result);
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS powershell

```
PS C:\Users\Ahmed Sherif\Documents\SE-W24\Assignment> node testScenerio.js
test case 1 { id: '16', name: 'German 1', code: 'DE101', creditHours: 4 }
test case 2 {}
```

Implement the function searchCoursesByCreditHours(courseFile , creditHours)

- courseFile : (typeof string) : directory/path of courses text file
- creditHours : (typeof number) : any number between 1 and 8

Function Description:

The function returns an array of course objects. The array will include only the course objects that have the same number of credit hours as the input `creditHours`. If no course objects match the `creditHours`, the function will return an empty array.

Course object has the following properties

Property	DataType	Description
id	string	course id
name	string	course name
code	string	course code
creditHours	number	credit hours assigned to the course

For Example

testScenerio.js > ...

```
2   let courseFile = "courses.txt";
3   let creditHours = 7;
4   let result = searchCoursesByCreditHours(courseFile , creditHours);
5   console.log("test case 1",result);
6
7   creditHours = 3;
8   result = searchCoursesByCreditHours(courseFile , creditHours);
9   console.log("test case 2",result);
```

```
PS C:\Users\Ahmed Sherif\Documents\SE-W24\Assignment> node testScenerio.js
test case 1 [
  {
    id: '22',
    code: 'CSIS201',
    name: 'Computer Science II: Programming II',
    creditHours: 7
  },
  {
    id: '32',
    code: 'INCS101',
    name: 'Programming III',
    creditHours: 7
  }
]
test case 2 []
```

Implement the function deleteStudentInfo(studentFile , studentID)

- studentFile : (typeof string) : directory/path of students text file
- studentID : (typeof string) : ID of given student

Function Description:

The `deleteStudentInfo` function deletes a student's information based on `studentID` and returns "successfully deleted" if the deletion occurs. If the student is not found, it returns "student id `studentID` does not exist", where the actual value of `studentID` is displayed.

For Example

students.txt

```
ID,Name,Email,Major
10003827,Jana Elsherif,jana.elsherif@student.giu-uni.de,Mechanical Enginnering
7005838,Yassmin Hany,yassmin.hany@student.giu-uni.de,Electrical Enginnering
7001823,Youssef Elnady,youssef.elnady@student.giu-uni.de,Business Administration
```

```
const {deleteStudentInfo} = require("./main");
let studentFile = "students.txt";
let studentID = "13009834";
let result = deleteStudentInfo(studentFile , studentID);
console.log("test case 1",result);

studentID = "7005838";
result = deleteStudentInfo(studentFile , studentID);
console.log("test case 2",result);
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\Ahmed Sherif\Documents\SE-W24\Assignment> node testScenerio.js
test case 1 student id 13009834 does not exist
test case 2 successfully deleted
```

The expected studentFile directory after running the above code

students.txt

```
ID,Name,Email,Major
10003827,Jana Elsherif,jana.elsherif@student.giu-uni.de,Mechanical Enginnering
7001823,Youssef Elnady,youssef.elnady@student.giu-uni.de,Business Administration
```

Implement the function updateCourseInfo(courseFile , courseObject)

- courseFile : (typeof string) : directory/path of courses text file
- courseObject : (typeof object) : includes some or all properties of course object

Function Description:

The function modifies the course information in the `courseFile` directory without returning any value. The provided `courseObject` must include an `id`, while all other properties are optional. If present, these properties will be used to update the corresponding course information in the `courseFile` directory.

Course object has the following properties

Property	DataType	Description
id	string	course id
name	string	course name
code	string	course code
creditHours	number	credit hours assigned to the course

For Example

courses.txt

```
ID,Code,Name,creditHours
11,MATH103,Mathematics I: Calculus I,8
12,CSEN104,Introduction to Computer Science: Programming I,6
16,DE101,German 1,4
21,MATH204,Mathematics II: Linear Algebra,4
22,CSIS201,Computer Science II: Programming II,7
31,MATH105,Mathematics III: Discrete Mathematics,4
33,INCS102,Operating Systems,5
```

testScenerio.js > ...

```
const {updateCourseInfo} = require("./main");
// 33,INCS102,Operating Systems,5
let courseFile = "courses.txt";
let courseObject = {id:33,creditHours:4};
updateCourseInfo(courseFile , courseObject);

courseFile = "courses.txt";
courseObject = {id:31,code:"MATH305",creditHours:5};
updateCourseInfo(courseFile , courseObject);
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS powershell

```
PS C:\Users\Ahmed Sherif\Documents\SE-W24\Assignment> node testScenerio.js
PS C:\Users\Ahmed Sherif\Documents\SE-W24\Assignment>
```

The expected courseFile directory after running the above code

courses.txt

```
ID,Code,Name,creditHours
11,MATH103,Mathematics I: Calculus I,8
12,CSEN104,Introduction to Computer Science: Programming I,6
16,DE101,German 1,4
21,MATH204,Mathematics II: Linear Algebra,4
22,CSIS201,Computer Science II: Programming II,7
31,MATH305,Mathematics III: Discrete Mathematics,5
33,INCS102,Operating Systems,4
```

Implement the function `addStudent(studentFile, studentObject)`

- `studentFile` : (`typeof string`) : The directory/path of the student text file.
- `studentObject` : (`typeof object`) : An object containing three properties:
  - `id` : (`string`) : The unique identifier of the student.
  - `name` : (`string`) : The name of the student.
  - `major` : (`string`) : The major of the student.

Function Description:

The function checks if a student entry with the specific `id` already exists in the studentFile directory.

- If an entry with the same `id` is found, the function returns an **empty object**.
- If no matching entry is found, the function adds the new student entry to the `studentFile` directory and returns the newly created `student Object`. The function must add new entry in a specific order within the text file. The new entry will be appended as the last record.

Student object has the following properties

Property	DataType	Description
id	string	student id
name	string	student name
email	string	student email
major	string	student major

Note that email is obtained in this format firstName.lastName@student.giu-uni.de
Make sure that all letters in both firstName and lastName are lowercase letters.

The Reasoning for the below Output:

- **Test Case 1:** Omar Elhalawany's student ID (10001984) did not exist in the `studentFile` directory. Therefore, the function should return a new student object and append his record as the last entry in the `studentFile` directory.
- **Test Case 2:** In this case, we attempted to add Omar Elhalawany's student ID again. Since his ID already exists in the `studentFile` directory, no changes should be made to the `studentFile` directory, and the function should return an empty object.

For Example

students.txt

```
ID,Name,Email,Major
10003827,Jana Elsherif,jana.elsherif@student.giu-uni.de,Mechanical Enginnering
10006840,Evram Joseph,evram.joseph@student.giu-uni.de,Pharmaceutical Engineering
10004867,Omar Khairy,omar.Khairy@student.giu-uni.de,Mechanical Enginnering
10006476,Arwa Omara,arwa.omara@student.giu-uni.de,Mechanical Enginnering
10003898,Haidy Hazem,haidy.hazem@student.giu-uni.de,Mechanical Enginnering
10003857,Fatma Ashraf,fatma.ashraf@student.giu-uni.de,Mechanical Enginnering
```

testScenerio.js > ...

```
let studentFile = "students.txt";
let studentObject = {id:"10001984",name:"Omar Elhalawany",
                     major:"Mechanical Engineering"};
let result = addStudent(studentFile , studentObject);
console.log("Test case 1",result);
result = addStudent(studentFile , studentObject);
console.log("Test case 2",result);
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS powershell

```
PS C:\Users\Ahmed Sherif\Documents\SE-W24\Assignment> node testScenerio.js
Test case 1 {
  id: '10001984',
  name: 'Omar Elhalawany',
  major: 'Mechanical Engineering',
  email: 'omar.elhalawany@student.giu-uni.de'
}
Test case 2 {}
```

The expected studentFile directory after running the above code

students.txt

```
ID,Name,Email,Major
10003827,Jana Elsherif,jana.elsherif@student.giu-uni.de,Mechanical Enginnering
10006840,Evram Joseph,evram.joseph@student.giu-uni.de,Pharmaceutical Engineering
10004867,Omar Khairy,omar.Khairy@student.giu-uni.de,Mechanical Enginnering
10006476,Arwa Omara,arwa.omara@student.giu-uni.de,Mechanical Enginnering
10003898,Haidy Hazem,haidy.hazem@student.giu-uni.de,Mechanical Enginnering
10003857,Fatma Ashraf,fatma.ashraf@student.giu-uni.de,Mechanical Enginnering
10001984,Omar Elhalawany,omar.elhalawany@student.giu-uni.de,Mechanical Engineering
```

Implement the function `addCourse(courseFile , courseObject)`

- `courseFile` : (`typeof string`) : The directory/path of the course text file.
- `courseObject` : (`typeof object`) : An object containing four properties.

Function Description:

The function checks if a course entry with the specific `id` already exists in the courseFile directory.

- If an entry with the same `id` is found, the function returns an **empty object**.
- If no matching entry is found, the function adds the new course entry to the `courseFile` directory and returns the `course Object`. The function must add new entry in a specific order within the text file. The new entry will be appended as the last record.

Course object has the following properties

Property	DataType	Description
id	string	course id
name	string	course name
code	string	course code
creditHours	number	credit hours assigned to the course

The Reasoning for the below Output:

- **Test Case 1:** course ID (51) did not exist in the `courseFile` directory. Therefore, the function should return a course object and append a record as the last entry in the `courseFile` directory.
- **Test Case 2:** In this case, we attempted to add the same course ID (51) again. Since the course ID already exists in the `courseFile` directory, no changes should be made to the `courseFile` directory, and the function should return an empty object.

For Example

courses.txt

```
ID,Code,Name,creditHours
11,MATH103,Mathematics I: Calculus I,8
12,CSEN104,Introduction to Computer Science: Programming I,6
13,PHYS101,Physics,5
14,CSEN103,Digital Logic Design,4
15,AS101,English for Academic Skills,4
16,DE101,German 1,4
```

testScenerio.js > result

```
let courseFile = "courses.txt";
let courseObject = {id:51,code:"ICS501",name:"Software Project",creditHours:4};
let result = addCourse(courseFile , courseObject);
console.log("Test case 1",result);
result = addCourse(courseFile , courseObject);
console.log("Test case 2",result);
addCourse(courseFile , courseObject)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS powershell

```
PS C:\Users\Ahmed Sherif\Documents\SE-W24\Assignment> node testScenerio.js
Test case 1 { id: 51, code: 'ICS501', name: 'Software Project', creditHours: 4 }
Test case 2 {}
```

The expected studentFile directory after running the above code

courses.txt

```
ID,Code,Name,creditHours
11,MATH103,Mathematics I: Calculus I,8
12,CSEN104,Introduction to Computer Science: Programming I,6
13,PHYS101,Physics,5
14,CSEN103,Digital Logic Design,4
15,AS101,English for Academic Skills,4
16,DE101,German 1,4
51,ICS501,Software Project,4
```

Implement the function `function addGrade(gradeFile , gradeObject)`

- `gradeFile` : (`typeof string`) : The directory/path of the grades text file.
- `gradeObject` : (`typeof object`) : An object containing four properties.

Function Description:

The function checks if an entry with a specific `studentId` and `courseId` exists in the `gradeFile` directory.

- If no matching entry is found, the function adds the new grade entry to the `gradeFile` directory. The new grade entry will be appended as the last record.
- If a matching entry is found and the student's current grade is better than their previous grade, the old record is removed from the `gradeFile` directory, and the new grade entry is appended as the last record. Otherwise, the file remains unchanged.

Grade object has the following properties

Property	DataType	Description
studentId	string	student id
courseId	string	course id
semester	string	semester
grade	string	A+,A,A-,B+,B,B-,C+,C,C-,D+,D,F

The Reasoning for the below GradeFile Output:

- **Test Case 1:** There is no row that contains both student ID (10003827) and course ID (46) in the `gradeFile`. Therefore, the record was inserted at line 7 as the last entry in the `gradeFile` directory.
- **Test Case 2:** The row containing student ID (7004848) and course ID (13) already exists in the `gradeFile`. However, student ID (7004848) obtained a higher grade (A+) compared to her previous grade (D) for the same course ID (13). Therefore, we need to remove her old grade at line 4 (blue box) and add a new entry with the updated grade at line 8 as the last entry in the `gradeFile`.
- **Test Case 3:** The row containing student ID (7005848) and course ID (37) already exists in the `gradeFile` directory. However, student ID (7005848) obtained a lower grade (C) compared to her previous grade (C+) for the same course ID (37). Therefore, her old grade at line 3 (black box) should be kept, and no changes should be made to the `gradeFile` directory as she received a lower grade than her previous one.

Note that the blue, black, and red rectangular boxes are not part of the actual output. They were used only as visual aids to point out specific row locations in the two files: the file before running the function and the file after running the function. The boxes were simply there to help reference rows in each file.

For Example

grades.txt

```
StudentID,CourseID,Semester,Grade
10003827,47,Summer22,A
7005838,37,Winter21,C+
7004848,13,Spring22,D
7002849,48,Winter22,D+
10006840,15,Spring21,B-
10006840,14,Summer21,B-
```

testScenerio.js > ...

```
const {addGrade} = require("./main");
let gradeFile = "grades.txt";
let gradeObject = {studentId : "10003827",courseId : "46",
                   semester : "Spring23",grade : "A+",};
addGrade(gradeFile , gradeObject); // Test Case 1

gradeFile = "grades.txt";
gradeObject = {studentId : "7004848",courseId : "13",
                 semester : "Spring23",grade : "A+",};
addGrade(gradeFile , gradeObject); // Test Case 2

gradeFile = "grades.txt";
gradeObject = {studentId : "7005838",courseId : "37",
                 semester : "Spring23",grade : "C",};
addGrade(gradeFile , gradeObject); // Test Case 3

```

The expected gradeFile directory after running the above code

grades.txt

```
StudentID,CourseID,Semester,Grade
10003827,47,Summer22,A
7005838,37,Winter21,C+
7002849,48,Winter22,D+
10006840,15,Spring21,B-
10006840,14,Summer21,B-
10003827,46,Spring23,A+
7004848,13,Spring23,A+
```

Implement the function `function calculateGPA(gradeFile , courseFile , studentId , semester)`

- `gradeFile` : (`typeof string`) : The directory/path of the grades text file.
- `courseFile` : (`typeof string`) : The directory/path of the courses text file.
- `studentId` : (`typeof string`) : student id.
- `semester` : (`typeof string`)

Function Description:

The function returns the GPA for a given `studentId` in the specified `semester`. If the `semester` is left empty, the function returns the cumulative GPA across all semesters. If the `studentId` does not exist in the `gradeFile`, the function returns the message "not eligible for GPA calculation". If the `studentId` exists, the function returns a numeric value representing the GPA, rounded up to two decimal points.

GPA Calculation:

- For each course taken by the student in the specified semester (or all semesters if the semester is empty), retrieve the grade.
- Convert the letter grades to their corresponding numeric values (e.g., A+ = 0.7, A = 1, A- = 1.3, B+ = 1.7,B = 2, B- = 2.3, C+ = 2.7,C = 3, C- = 3.3, D+ =3.7 , D = 4, F =5).
- Calculate the total quality points by multiplying each course's grade points by the number of credit hours for that course.
- Divide the total quality points by the total number of credits attempted to obtain the GPA.
- Round the GPA down to two decimal points before returning it (1.798 is rounded down to 1.79).

The Reasoning for the below Output:

- **Test Case 1:** Student id 10006840 took two courses in Spring 21. The first course has an ID of 15 with 4 credit hours and got a B- (2.3). The second course has an ID of 23 with 5 credit hours and got an A+ (0.7).
- **Test Case 2:** The semester was provided as an empty string, so we must calculate the cumulative GPA for the given student ID across all semesters. Student ID 10006840 took two courses in Spring 21. Additionally, in Summer 21, the student took one course with 4 credit hours and received a B-.
- **Test Case 3:** Student ID 10003827 did not take any courses in the Winter 23 semester. Therefore, the output was "not eligible for GPA calculation".

To calculate GPA in test case 1, total quality points and credit hours should be determined:

```
Total Quality Points = (2.3 * 4) + (0.7 * 5) = 12.7
Total Credit Hours = 4 + 5 = 9
GPA = Total Quality Points / Total Credit Hour = 1.411111;
The GPA is rounded down to 1.41
```

For Example

grades.txt

```
StudentID,CourseID,Semester,Grade
10003827,47,Summer22,A
7005838,37,Winter21,C+
7002849,48,Winter22,D+
10006840,15,Spring21,B-
10006840,23,Spring21,A+
10006840,14,Summer21,B-
10003827,46,Spring23,A+
7004848,13,Spring23,A+
```

testScenerio.js > ...

```
const {calculateGPA} = require("./main");
let gradeFile = "grades.txt";
let courseFile = "courses.txt";
let studentId = "10006840";
let semester = "Spring21";
let result = calculateGPA(gradeFile , courseFile , studentId , semester);
console.log("Test Case 1",result)

semester = "";
result = calculateGPA(gradeFile , courseFile , studentId , semester);
console.log("Test Case 2",result);

studentId = "10003827";
semester = "Winter23";
result = calculateGPA(gradeFile , courseFile , studentId , semester);
console.log("Test Case 3",result);
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\Ahmed Sherif\Documents\SE-W24\Assignment> node testScenerio.js
Test Case 1 1.41
Test Case 2 1.68
Test Case 3 not eligible for GPA calculation
PS C:\Users\Ahmed Sherif\Documents\SE-W24\Assignment>
```

Submission guidelines:

- **You are not allowed to use any external library and you can use any predefined methods. External libraries will require npm installation. You are welcome to use any library that does not require npm installation. Using an external library will result in receiving a zero grade.**
- **Submit all your code inside main.js**
- **Submitting your code in different file other than main.js will result in zero grade**
- Submit all your code in **ONE** .zip file.
- The team can consist of 2 to 3 members.
- **Rename the .zip folder in the following format id-Tutorial-id-Tutorial (1000546-T8-1000678-T9)**
- **Renaming file is an important step for receiving your grade**
- Make sure to submit before deadline
- **The submission team link is https://forms.gle/mBACTUTgJgv9N4F59**
- **The submission link is https://forms.gle/rZv4VAB8mTUF6r5z8**
- It is **YOUR** responsibility **to ensure that you have submitted the correct .zip file and saved the correct content before submitting as well as submitting your colleague's correct student IDs in the renaming the zip folder.**
- **Good luck! :)**
