Project work

The project work is a more demanding programming task that requires mastering the skills learned during the course. Your task is to program ***a study register for a university.***

Timetable

23/10	Assignment is published
30/10	The submission box opens in ***CodeGrade*** (at latest)
13/11	**Deadline** for submitting the project work
27/11	By this date, you will get the grade. Resubmission opens for works to be corrected / completed.
5/12	Resubmission closes

Files

Your system will rely on three text files, which contain the data needed to run the system. You can find models of these files in Moodle folder “Files for project”. The idea is that while running your program, the content of these files change. If you manage to corrupt the content of your files, you can always start again by downloading the files from Moodle.

`courses.txt`: This file contains the information of the courses that are taught in this university. The information about one course is given in one row. Rows look like this:

```
A0400,Artificial Intelligence Applications,6,Anna Johnson
A0450,Network Security and Cryptography,6,Paul Davis,Emma Martinez
```

The data is given in comma-separated (CSV) form. First there is the course code (`A0400`), which identifies the course. Then there is the name of the course (`Artificial Intelligence Applications`). The following number (6) is the amount of study points given by the course. Finally, there are the name(s) of the teacher (`Anna Johnson`). Note that there may be several teachers on one course. Different teachers are separated by commas. The newline symbol `"\n"` ends each row in every file. The content of this file **does not** change, your program should only read information from it.

`students.txt`: This file contains information about the students. The information of one student is given in one line. Different values are separated by a comma. Lines look like this:

```
61835,David,Miller,2021,ME,david.miller@lut.fi
48927,Grace,Lee,2022,CE,grace.lee@lut.fi
```

First there is the student ID (`61835`). This integer serves as the identifier of the students. This means that there cannot be two students with the same ID. Then comes the student’s first name (`David`) and last name (`Miller`). After the name comes the starting year (`2021`) and the Bachelor program. The last value is the email address of the student (`david.miller@lut.fi`).

The Bachelor Program must be one of the following: Computational Engineering (CE), Electrical Engineering (EE), Energy Technology (ET), Mechanical Engineering (ME), Software Engineering (SE). This information is stored as 2-letter shorthand.

The content of this file will change: if a new student enters the university, her/his information is added to the file. However, students are ***never*** removed, so they remain in the system “forever”.

`passed.txt`: This file contains the actual information about what courses students have passed. One row looks like this:

```
B0350,48927,16/10/2022,4
A0120,61835,07/02/2022,5
```

First there is the course code (`B0350`) of the passed course, then the ID (`48927`) of the student who passed the course. The third value is the date (`16/10/2022`) when the student passed the course. The format is DD/MM/YYYY. The last value is the grade 1-5. The values are separated by a comma.

When a student passes a course, this information is added to the file. The order in the file is not meaningful.

Different operations

The program needs to be menu-based. When the program starts it prints the following menu:

```
You may select one of the following:
                1) Add student
                2) Search student
                3) Search course
                4) Add course completion
                5) Show student's record
                0) Exit
What is your selection?
```

This menu should be also displayed after every operation until you end the program by selecting 0.

In the following, each operation is described in detail.

`1) Add Student`: This operation adds a new student entry to the `'students.txt'` file. The process begins by prompting the user for the student's first name and last name. The system needs to check that the name contains only letters, and the names start with capital letters. Subsequently, the system automatically generates the email address in the format of `firstname.lastname@lut.fi`. Additionally, the system generates a unique study number by randomly selecting a number between 10000 and 99999, ensuring it's not already in use. The starting year is automatically set to the current year, utilizing the 'datetime' library to retrieve this information."

```
Names should contain only letters and start with capital letters.
Enter the first name of the student:
Tom7
Enter the last name of the student:
Thomas
Names should contain only letters and start with capital letters.
Enter the first name of the student:
Tom
Enter the last name of the student:
Thomas
Select student's major:
                CE: Computational Engineering
                EE: Electrical Engineering
                ET: Energy Technology
                ME: Mechanical Engineering
                SE: Software Engineering
What is your selection?
ME
Student added successfully!
```

`2) Search student:` You can find any student by giving a part of the first name or a part of the last name of a student. The search string must contain at least *three* non-blank characters. This operation is suitable for finding the students ID, which is needed in operations 4 and 5. The operation prints the ID, the first and the last name according to the example below.

```
Give at least 3 characters of the students first or last name:
An
Give at least 3 characters of the students first or last name:
Ann
Matching students:
ID: 82346, First name: Anna, Last name: Johnson
```

`3) Search course`: You can give a part of the course name or part of the teacher's name. Note that there can be several teachers in one course.

```
Give at least 3 characters of the name of the course or the teacher:
Dav
ID: A0120, Name: Software Engineering Principles, Teacher(s): David Miller
ID: A0450, Name: Network Security and Cryptography, Teacher(s): Paul Davis, Emma Martinez
ID: A0220, Name: Computer Graphics and Visualization, Teacher(s): Michelle Davis
ID: B0490, Name: Cloud Computing Technologies, Teacher(s): Alexander White, Lucy Davis
```

```
Give at least 3 characters of the name of the course or the teacher:
Com
ID: A0100, Name: Introduction to Computer Science, Teacher(s): Emily Johnson
ID: B0290, Name: Computer Networks and Protocols, Teacher(s): Jason Thompson
ID: A0220, Name: Computer Graphics and Visualization, Teacher(s): Michelle Davis
ID: B0490, Name: Cloud Computing Technologies, Teacher(s): Alexander White, Lucy Davis
```

`4) Add course completion:` The system asks the required information step by step according to the following examples. If student has already passed the course, the system checks that the new grade is higher than the old grade. There can be at most one completion of one course by one student.

The system needs to also check that the completion is added to the register within 30 days. It this is not true, then the program outputs: `Input date is older than 30 days. Please contact "opinto."` In such a case, nothing is added to the file.

```
Give the course ID:
AAA323
Give the course ID:
A0400
Give the student ID:
95378
Give the grade:
1
Student has passed this course earlier with grade 3
```

```
Give the course ID:
A0400
Give the student ID:
58291
Give the grade:
6
Grade is not a correct grade.
```

```
Give the course ID:
A0400
Give the student ID:
58291
Give the grade:
4
Enter a date (DD/MM/YYYY):
32/09/2023
Invalid date format. Use DD/MM/YYYY. Try again!
```

```
Give the course ID:
A0400
Give the student ID:
58291
Give the grade:
4
Enter a date (DD/MM/YYYY):
12/12/2023
Input date is later than today. Try again!
```

```
Give the course ID:
A0400
Give the student ID:
58291
Give the grade:
4
Enter a date (DD/MM/YYYY):
15/06/2023
Input date is older than 30 days. Contact "opinto".
```

```
Give the course ID:
A0400
Give the student ID:
58291
Give the grade:
4
Enter a date (DD/MM/YYYY):
07/10/2023
Input date is valid.
Record added!
```

`5) Show student's record:` The program asks the students ID. Remember that you can search the ID by using operation 2. After this, the program displays the student's information according to the model below:

```
Student ID: 83216
Name: Johnson, Emily
Starting year: 2021
Major: Electrical Engineering
Email: emily.johnson@lut.fi

Passed courses:

Course ID: A0210, Name: Machine Learning Fundamentals, Credits: 6
Date: 16/01/2022, Teacher(s): Linda Chen, grade: 4

Course ID: A0400, Name: Artificial Intelligence Applications, Credits: 6
Date: 15/02/2022, Teacher(s): Anna Johnson, grade: 5

Course ID: B0110, Name: Mobile App Development, Credits: 3
Date: 13/05/2023, Teacher(s): Sarah Lewis, grade: 5

Course ID: A0220, Name: Computer Graphics and Visualization, Credits: 6
Date: 16/08/2023, Teacher(s): Michelle Davis, grade: 1

Course ID: A0370, Name: Information Retrieval Systems, Credits: 6
Date: 25/04/2022, Teacher(s): Thomas Moore, grade: 5

Total credits: 27, average grade: 4.0
```

Submission and grading

The program is submitted to ***CodeGrade***, but teachers will grade the work according to the checklist below. From each item it is possible to obtain 0-4 points. This means that it is possible to gather 100 points. Note that you do not have to implement all the functionality, but general rule is that more you implement, the better grade you will have.

To pass the project work, you need to gather ***at least 50 points***. The grade of the project work is determined by this table:

Points	50-59	60-69	70-79	80-89	90-100
Grade	1	2	3	4	5

The items in the following **checklist** will be evaluated:

1) The menu is printed correctly. Display appropriate message if the input is not between 0-5. Program ends when 0 is given in menu.
2) The program is divided into *functions* in a proper manner. Required data is passed as arguments to the functions.
3) All opened files are closed.
4) The program should not crash in any situation.
5) When adding a new student, check that the names contain only letters, and they begin with capital letters.
6) The email address is generated correctly.

7) The study number is generated correctly (between 10000-99999, no duplicates)
8) The starting year is added using `datetime` module.
9) The major subject is added correctly.
10) The data is added correctly to the file (e.g., no empty lines)
11) When searching a student, it is checked that the search term contains at least 3 non-blank characters. This search term is searched both from the first name and the last name.
12) When searching a course, it is checked that the search term contains at least 3 non-blank characters. This search term is searched both from the course name and the names of the teacher(s). Note that there can be several teachers.
13) When add a new course completion, check that the course and the student really exist.
14) Check if the student has passed the course earlier. If the student has not passed the course, add a new row to the `passed.txt`.
15) If the student has passed the course, if the old grade is better than the new grade, do nothing.
16) If the student has passed the course, if the new grade is better than the old grade, then “update” the information in `passed.txt.` Make sure that the student does not have two entries for the same course in the file `passed.txt`.
17) Check that the grade is correctly between 1-5.
18) Check that the date when the course was passed is of the form DD/MM/YYYY. IT is advised to use `datetime` module.
19) Check that the date when the course was passed is not in the future.
20) Check that the date when the course was passed is not older than 30 days.
21) When printing the record, make sure that the student’s personal information is printed correctly.
22) When printing the courses of the student, you need to combine data from the files passed.txt and courses.txt. Please be certain that you are only printing courses which the student has passed. And you print only these courses.
23) Check that in the listing each course contains the following information: Course ID, Name of the course, number of Credits, the date when the student passed the course, the Name(s) of the teacher(s) of the course, the Grade student got from this course.
24) The total credits are computed correctly.
25) The average grade is computed correctly.

If you get less than 50 points, your project work needs to be corrected / completed. Also, those students who got the grade 1 or 2 in the first submission can resubmit their work. Note that **the maximum grade for resubmitted works is 3**.

Plagiarism

Plagiarism is the act of presenting someone else's work, ideas, code, or solutions as your own without providing proper credit. Plagiarism is considered unethical and a violation of academic or professional integrity, constituting a serious offense at LUT. To address this, all projects are carefully examined using CodeGrade's plagiarism detection tool.

If plagiarism is detected, the project work is deemed a failure, and the authorities at LUT are promptly notified. The plagiarism detection process involves comparing each submitted work against all other projects.

As the system cannot determine the origin of the copied content, all submissions with high plagiarism score are treated equally. In this kind of situation, it is impossible to assess who has produced the code. Sharing your code with another student for the purpose of replication is not permissible.

AI

LUT's AI-based tools policies: **https://elut.lut.fi/en/completing-studies/rules-and-regulations/ai-based-tools-policies**