# BBM-203 C++ Practice Assignment

Assistant Student Emirhan Yalçın

2023 Fall Term

## 1 Introduction

This assignment is designed to help you get familiar with basic operations in the **C++** programming language before you dive into relatively more complex BBM assignments. It's important to note that this assignment won't be graded, but we encourage you to solve the problems using the provided code.

The problems in this assignment are intentionally straightforward and not overly detailed. Our aim is to provide you with hands-on experience in basic C++ operations. Moreover, some of the methods you'll define in this exercise, like reading a text file or splitting a string, can be useful in future assignments.

We want to keep things simple and student-friendly as you explore the world of C++ programming. Good luck!

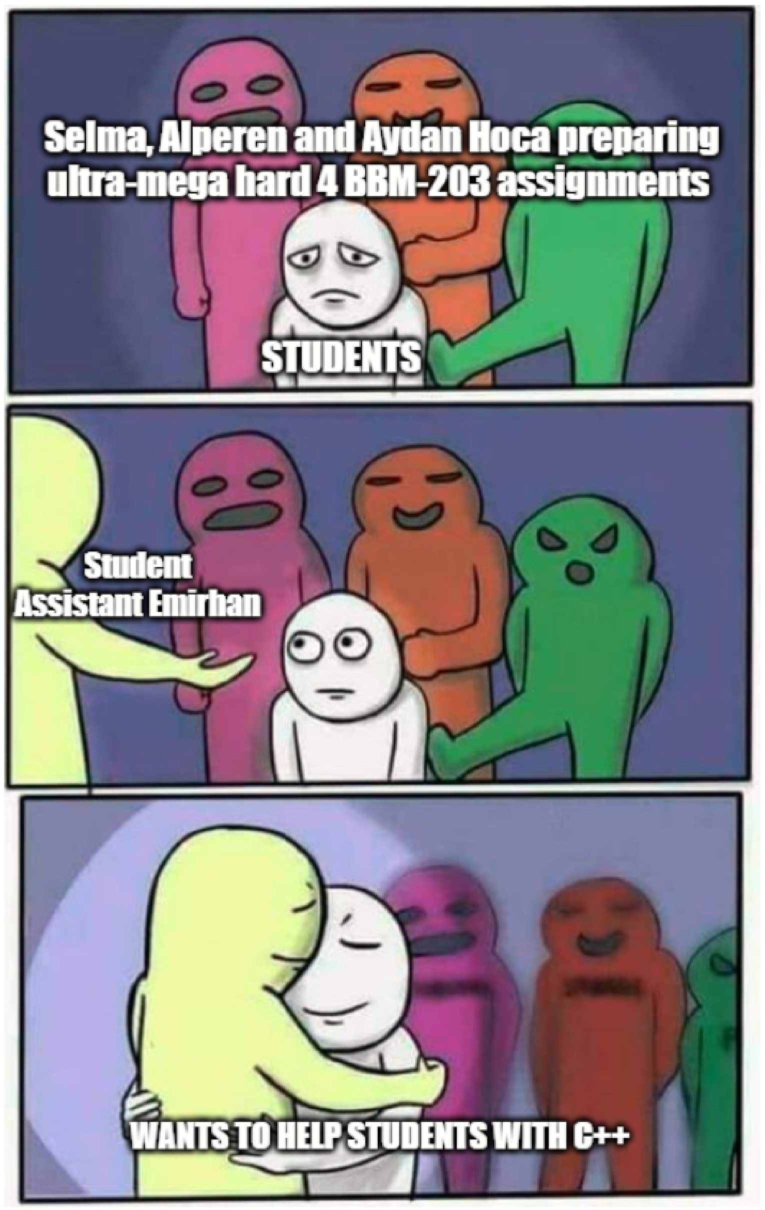

Selma, Alperen and Aydan Hoca preparing ultra-mega hard 4 BBM-203 assignments
STUDENTS
Student Assistant Emirhan
WANTS TO HELP STUDENTS WITH C++

Figure 1: We want you all to get some practice with c++ and handle your assignments faster and easier.

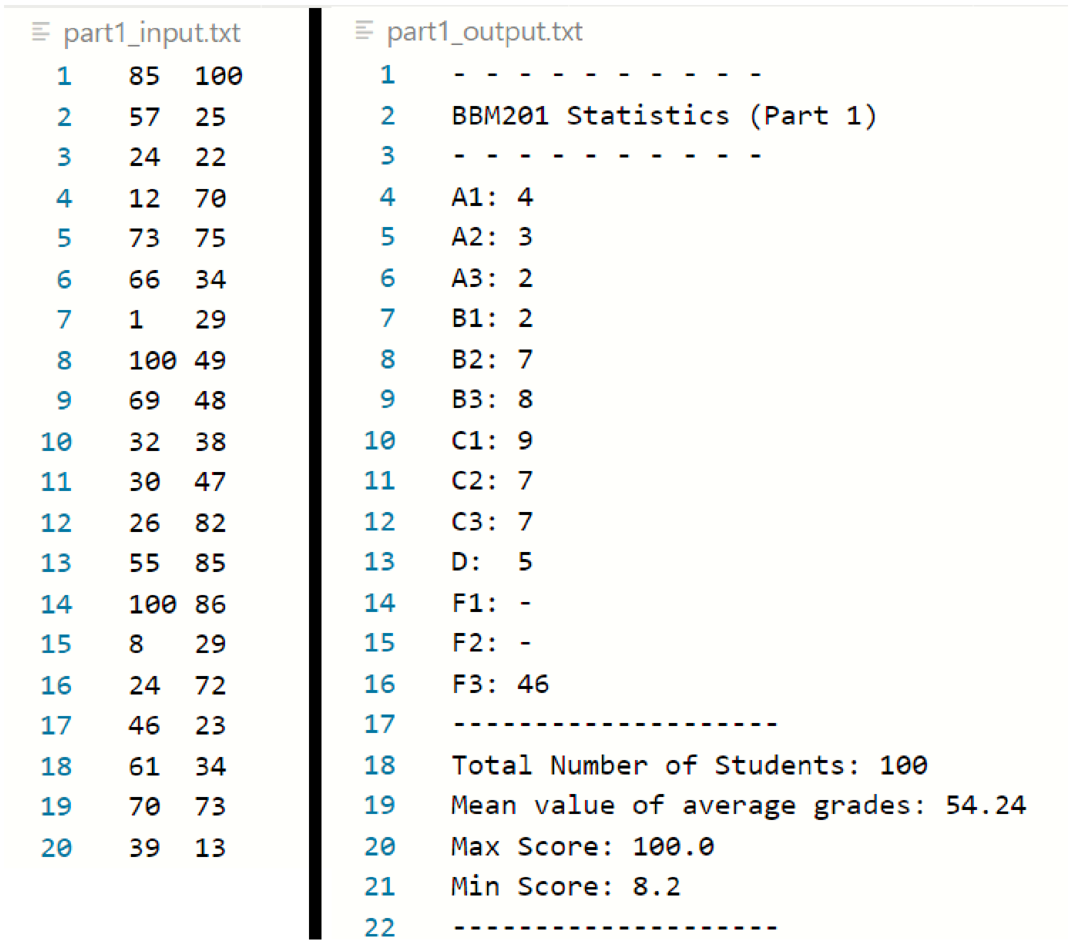
## 2 Problems

- **Part 1: Calculating Simple Statistics of a Exam Scores:**

In this part of the problem, you will given a sample input file containing midterm and final exam grades of students, seperated by a tab character. Your purpose is calculating average grades and letter grades of each student. Create a course statictics table, containing number of each letter grade, maximum and minimum score of the grades. For this course, take %40 of the midterm score and %60 of final exam score. Below you can see score ranges and corresponding letter grades:

| **Range** | **Letter** |
|---|---|
| 100 - 95 | **A1** |
| 95 - 90 | **A2** |
| 90 - 85 | **A3** |
| 85 - 80 | **B1** |
| 80 - 75 | **B2** |
| 75 - 70 | **B3** |
| 70 - 65 | **C1** |
| 65 - 60 | **C2** |
| 60 - 55 | **C3** |
| 55 - 50 | **D** |
| 50 - 0 | **F3** |
| Did not take final exam | **F2** |
| Absent | **F1** |

You are expected to create an output file, which contains statistics of this course. Fill the given methods in the given cpp files. Implement the **analize** function that creates the statistics output file. If you want you can change the given code and create any additional function that you want. For part 1 and part 2, assume that no one will fail the course with F1 and F2 grades.



part1_input.txt

```
85	100
57	25
24	22
12	70
73	75
66	34
1	29
100	49
69	48
32	38
30	47
26	82
55	85
100	86
8	29
24	72
46	23
61	34
70	73
39	13
```

part1_output.txt

```
- - - - - - - - - - -
BBM201 Statistics (Part 1)
- - - - - - - - - - -
A1: 4
A2: 3
A3: 2
B1: 2
B2: 7
B3: 8
C1: 9
C2: 7
C3: 7
D:  5
F1: -
F2: -
F3: 46
--------------------
Total Number of Students: 100
Mean value of average grades: 54.24
Max Score: 100.0
Min Score: 8.2
--------------------
```

Figure 2: Sample lines from input file and output file for part 1. Note that grades separated by a tab character in input file.

## • Part 2: Applying Curve to the Grades of Students

In this part, we have a similar problem, the difference is there are 2 midterms and 1 final exam score this time. But also we will apply **curve** to this grades.

***What is a curve in grading?***

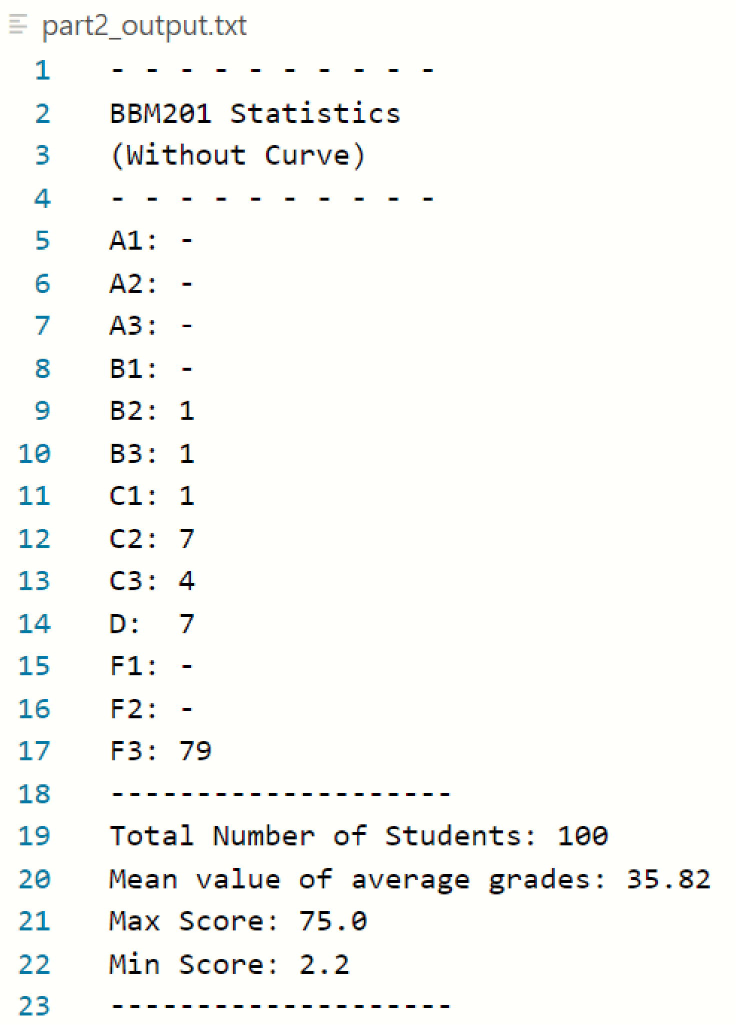
*In the realm of education, a grading curve is essentially a method for adjusting and scaling the grades of students following an assessment, such as a test, assignment, or homework. Typically, the purpose of applying a curve is to elevate the class's average score and improve the grades of individual students to some extent. Some professors never used curve. Curve is beneficial for students especially when there were and extremely hard exam material etc.*

As mentioned in the definition, we will carry out an experiment by applying grading curves in different ways and then comparing the results with the original grades. We will implement the grading curve using three distinct ways:

1. **Increase by Class Average:** This type of curve takes into account the average score in the class. Let's say the class's average score is 43, and the passing grade is set at 50. With this curve, we will add 7 points to each student's average score.
2. **Increase by Highest Score:** This curve considers the highest score achieved in the class. If the top score in the class is 92, we will increase each student's score by 8 points.
3. **Increase by Fixed Number of Failing Students:** This curve method involves a strict determination of the number of students who will fail the course. For example, if you have 200 students in the course and aim for 150 of them to pass, you should first sort the grades. Since 50 students need to fail, you'll take the grade of the 51st student and increase it to 50. After this adjustment, 150 students will successfully pass the course. In this problem, we want %75 of students to pass the course and %25 of them to fail.

For this part, you will find 3 scores in each line, midterm 1, 2 and final exam respectively. Midterms effect the average score by %30 and final exam %40. Calculate the average of each students, then apply all types of curve and analize each score. You can use the **analize** function that you already defined in the part 1.

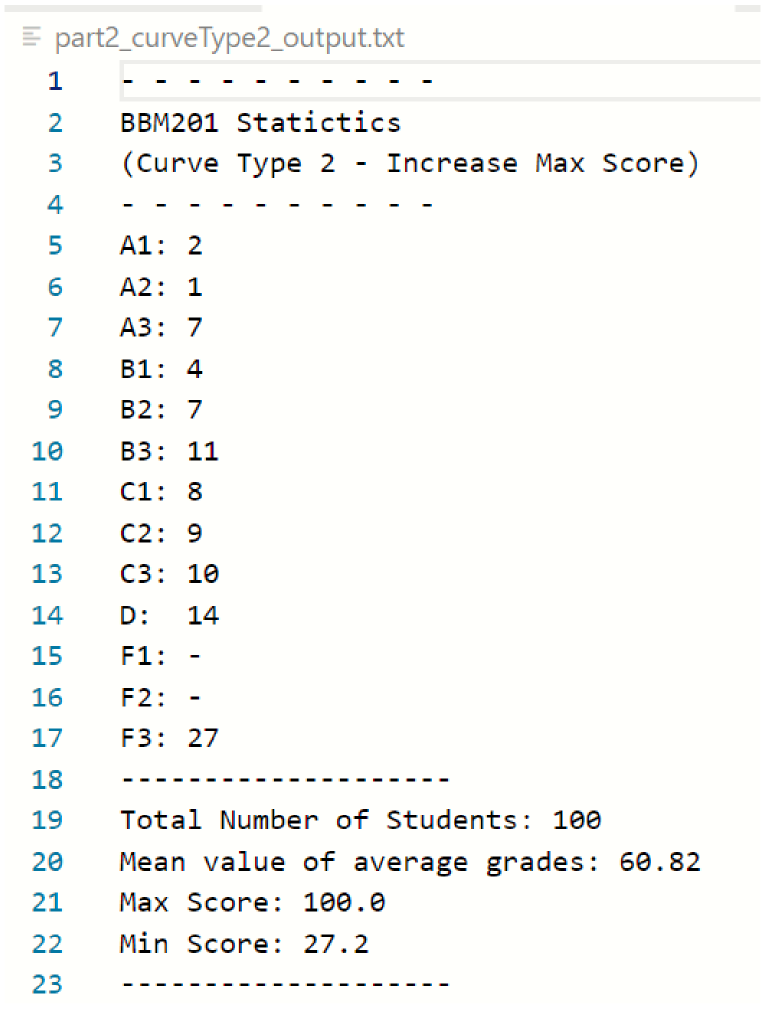
Our main purpose here is to examine how do different types of curve changes the statistics of the class. So basically, you will create 4 output files for this part. First will contain statictics before curve and each of other three will contain curved results. Don't forget to change title of the result tables in your output files.



part2_output.txt

```
- - - - - - - - - -
BBM201 Statistics
(Without Curve)
- - - - - - - - - -
A1: -
A2: -
A3: -
B1: -
B2: 1
B3: 1
C1: 1
C2: 7
C3: 4
D:  7
F1: -
F2: -
F3: 79
--------------------
Total Number of Students: 100
Mean value of average grades: 35.82
Max Score: 75.0
Min Score: 2.2
--------------------
```

Figure 3: Class statistics without curve


part2_curveType2_output.txt

```
- - - - - - - - - - -
BBM201 Statictics
(Curve Type 2 - Increase Max Score)
- - - - - - - - - - -
A1: 2
A2: 1
A3: 7
B1: 4
B2: 7
B3: 11
C1: 8
C2: 9
C3: 10
D:  14
F1: -
F2: -
F3: 27
--------------------
Total Number of Students: 100
Mean value of average grades: 60.82
Max Score: 100.0
Min Score: 27.2
--------------------
```

Figure 4: Class statictics after curve type - 2

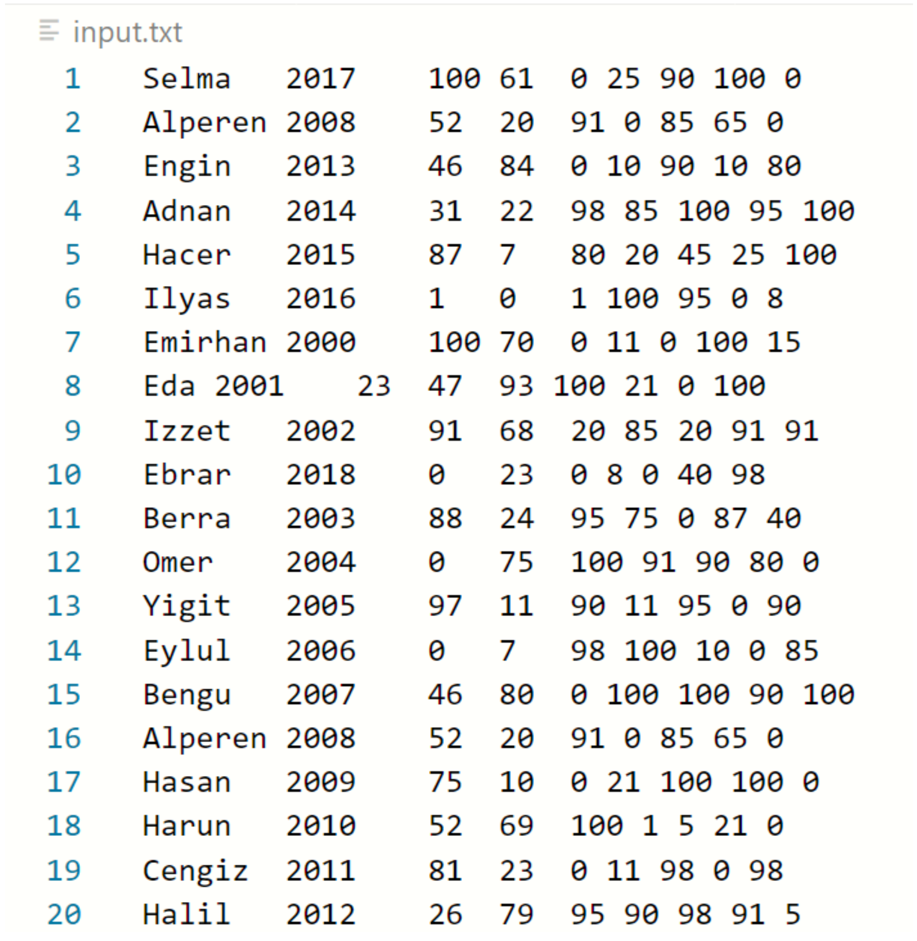
- **Part 3: Using Object-Oriented Programming and Detailed Analysis of Grades**

  In this section, you will encounter a more complex input file and additional details for analyzing a class of students. In this course, each student takes the usual midterm and final exams, but they are also required to submit five assignments to pass the course. There are some additional conditions to consider:

- If a student fails to submit more than two assignments, they will automatically fail the course with a grade of **F1**. It's important to note that even if a student submits a homework, they will receive at least 1 point for submission. For non-submission cases, the score for that assignment will be 0.
- If a student does not take the final exam, they will fail the course with a grade of **F2**. If a student takes the final exam, their paper will be graded and receive at least 1 point. A final score of 0 indicates that the exam was not taken.
- There is a passing threshold for the final exam. Students must score at least 50 points on the final exam to pass the course. Otherwise, they will fail the course with a grade of **F3**, regardless of their overall average score.
- The midterm exams contribute 30%, and the final exam contributes 40% to the overall grade, and the average of assignment scores affects 30% also. Each assignment carries a weight of 6% on the average. This time, there will be no grading curve.

In this part of the assignment, your task is to design a **Student** class with the necessary attributes and methods to simplify solving this problem and facilitate the creation of the output file. You can use the provided Student file and fill in the empty methods. You are also free to implement new methods if necessary or create a new C++ file from scratch.

Your input file now includes more elements. Each student has a name, student ID, midterm grade, final exam grade, and five homework grades.

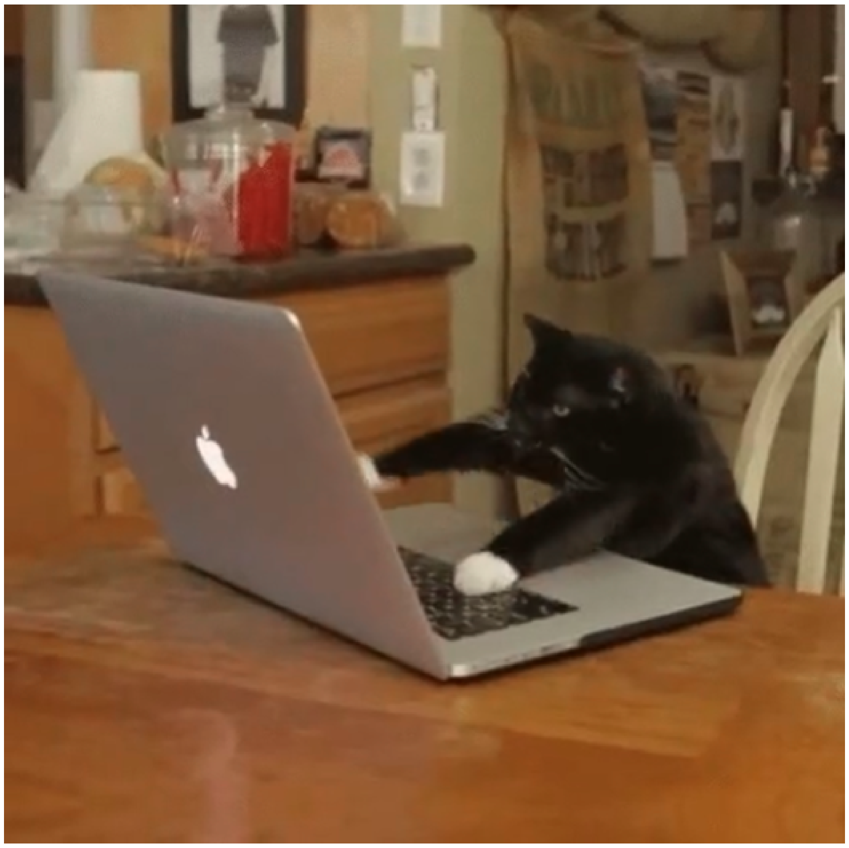


```
Selma   2017    100 61  0 25 90 100 0
Alperen 2008    52  20  91 0 85 65 0
Engin   2013    46  84  0 10 90 10 80
Adnan   2014    31  22  98 85 100 95 100
Hacer   2015    87  7   80 20 45 25 100
Ilyas   2016    1   0   1 100 95 0 8
Emirhan 2000    100 70  0 11 0 100 15
Eda 2001    23  47  93 100 21 0 100
Izzet   2002    91  68  20 85 20 91 91
Ebrar   2018    0   23  0 8 0 40 98
Berra   2003    88  24  95 75 0 87 40
Omer    2004    0   75  100 91 90 80 0
Yigit   2005    97  11  90 11 95 0 90
Eylul   2006    0   7   98 100 10 0 85
Bengu   2007    46  80  0 100 100 90 100
Alperen 2008    52  20  91 0 85 65 0
Hasan   2009    75  10  0 21 100 100 0
Harun   2010    52  69  100 1 5 21 0
Cengiz  2011    81  23  0 11 98 0 98
Halil   2012    26  79  95 90 98 91 5
```

Figure 5: Sample lines from the input file for Part 3. Note that major sections (name, ID, midterm, final, homeworks) are separated by tab characters, while homework grades are separated by white spaces within.

In this part, we have a few goals:

- Create the usual statistics file. (You can't use the analize function that you already defined and used in part 1 and part 2. Because analize function does not considers F1 and F2 cases and just considers the average grades and distibutes the scores into letter grades. You need a another analize function that counts all contitions to pass this course.)
- Organize students by their names; sort all students by their name and generate a result.txt file that includes name,, id and letter grade. We can make this process easier by using operator overloading, which is similar to a method called **toString()** in Java.
- Display the names and GPAs of the top 10 students with the highest averages in an another txt file. (You can use the same operator overloading for printing to the file.)But also include gpa this time.
- Create an F1 List containing the names and IDs of students who failed the course with an **F1** grade. You can print the F1 List to a txt file in any order.



## 3 Final Notes

– This assignment won't be graded. Its purpose is to help you practice basic C++ operations like file I/O and using object-oriented programming (OOP) concepts. It's recommended to solve these problems before diving into more complex data structures and real assignments.

– You'll be provided with sample codes that include some function and class definitions with empty bodies. You can use and fill in these functions and classes based on the provided design, and you're free to create new functions if needed. Alternatively, you can create a new C++ file and start from scratch.

– If you encounter difficulties while implementing a function, you can seek help from artificial intelligence. You can use AI to better understand C++ rules. For example, you can ask questions like, "Why is the '&' used as a parameter in this function?" If you encounter errors that you can't resolve, AI can assist you. However, avoid copying and pasting solutions; instead, do your best to implement functions and debug errors while understanding the process.

– Saving the code you write for this assignment can be beneficial. You might be able to reuse some of the functions you define in future assignments without having to redefine them.

– Best of luck with your assignment!