**CGPA and GPA Calculator – CS Project**

**Report**

**Group Members:**

**1) Muhamad Murtaza 2022402**

**Contribution:**

**1) Wrote The Complete Code**

**2) Added Comments**

**3) Made The Report**

**2) Saeed-Ur-Rehman 2022521**

**Contribution:**

**1) Helped In report Making**

**3) Muhammad Adnan Barki 2022332**

**Contribution:**

**1) Helped In Figuring out GPA Calculations**

**4) Muhammad Alyan 2022343**

**Contribution:**

**1) Helped In finding errors In Code**

***Written in C++***

# Introduction

The CGPA and GPA Calculator is a program developed in C++ to assist students in calculating their Grade Point Average (GPA) and Cumulative Grade Point Average (CGPA) based on their grades and credit hours. This project report aims to provide a comprehensive explanation of the code, including the reasoning behind the choice of classes, functions, and features.

# Language Choice

C++ was chosen as the programming language for this project due to its versatility, efficiency, and object-oriented capabilities. C++ provides the necessary tools for managing dynamic memory, handling file operations, and implementing control flow structures, making it suitable for developing complex applications like the CGPA and GPA Calculator.

# Data Types:

* Reasoning: The choice of data types in the program is crucial for efficient memory usage and accurate calculations. Common data types such as **int** were used to represent whole numbers, such as the number of subjects and credit hours. Arrays were used to store multiple grades and credit hours. Floating-point data types like **float** or **double** were used for precise calculation of grade points, GPA, and CGPA. These data types were selected based on the specific requirements of the calculations involved and the desired level of accuracy

# Classes and Inheritance

The code utilizes the concept of classes and inheritance to organize and structure the program's functionalities. The key classes used are **Student**, **GPACalculator**, and **CGPACalculator**.

## Student Class:

* + The **Student** class serves as the base class for the GPA and CGPA calculators, providing common attributes and methods.
  + It includes member variables to store the number of subjects, grades, and credit hours for a single semester.
  + The **getinput()** method is implemented to obtain user input for the number of subjects, credit hours, and grades.
  + Memory is allocated dynamically for the grade and credit hours arrays based on the number of subjects.
  + The destructor (**~Student()**) is responsible for freeing the dynamically allocated memory.

## GPACalculator Class:

* + The **GPACalculator** class inherits from the **Student** class, extending its functionalities.
  + It includes additional member variables to store the total credit hours, sum of points, and the calculated GPA.
  + The **gpa()** method calculates the GPA by iterating over the subjects and assigning grade points based on the grades entered.
  + It also displays the grades of each subject, total points, total credit hours, and the calculated SGPA (Semester GPA).
  + The **writeGPAResultsToFile()** method writes the GPA results to a specified output file.

## CGPACalculator Class:

* + The **CGPACalculator** class inherits from the **GPACalculator** class, further extending its functionalities.
  + It introduces the concept of multiple semesters, allowing the calculation of the CGPA.
  + The **getInput()** method prompts the user to enter the number of semesters and input data for each semester.
  + It calculates the CGPA by summing up the total points and credit hours of all semesters.
  + The **writeCGPAResultsToFile()** method writes the CGPA results, including the SGPA of

each semester, to the output file.

## Reasoning:

Classes and inheritance were used to organize the program's functionalities and establish a hierarchical structure. The base class **Student** encapsulates common attributes and methods related to a student's grades and credit hours for a single semester. Inheritance allows the derived classes (**GPACalculator** and **CGPACalculator**) to inherit the base class's properties while extending them to calculate GPA and CGPA, respectively. This design promotes code reuse, modularity, and maintainability.

# Dynamic Memory Allocation

Dynamic memory allocation is employed in the code to create arrays for storing grades and credit hours. The number of subjects entered by the user is used to determine the size of these arrays. The use of dynamic memory allocation allows flexibility in handling different numbers of subjects and prevents memory wastage.

## Reasoning:

Dynamic memory allocation was utilized to create arrays for storing grades and credit hours. By allocating memory dynamically based on the number of subjects entered by the user, the program can handle varying input sizes without wasting memory. This flexibility ensures that the program can accommodate any number of subjects the user wishes to calculate the GPA or CGPA for.

# File Handling

File handling operations are incorporated using the **fstream** library to read from and write to files. The program utilizes an **ofstream** object to create and write data to the output file. The inclusion of file handling provides the ability to save and retrieve GPA and CGPA results, enabling easy sharing and future reference.

## Reasoning:

File handling operations using the **fstream** library were incorporated to enable reading from and writing to files. This feature provides the ability to save and retrieve GPA and CGPA results to and from an external file. It allows students to store their academic data for future reference, easily share their results with others, or analyze their academic progress over time.

# Control Flow Structures

The code implements control flow structures such as the **switch** statement and **do-while** loop to provide a menu-driven interface for the user. The **switch** statement allows the selection of different options, including calculating GPA, calculating CGPA, and exiting the program. The **do-while** loop ensures that the menu is displayed repeatedly until the user chooses to exit.

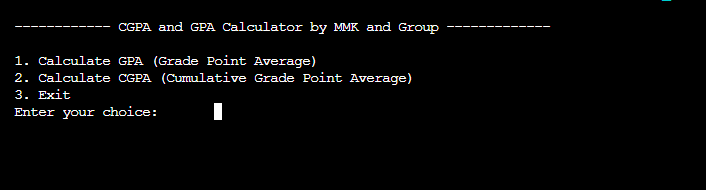
## Reasoning:

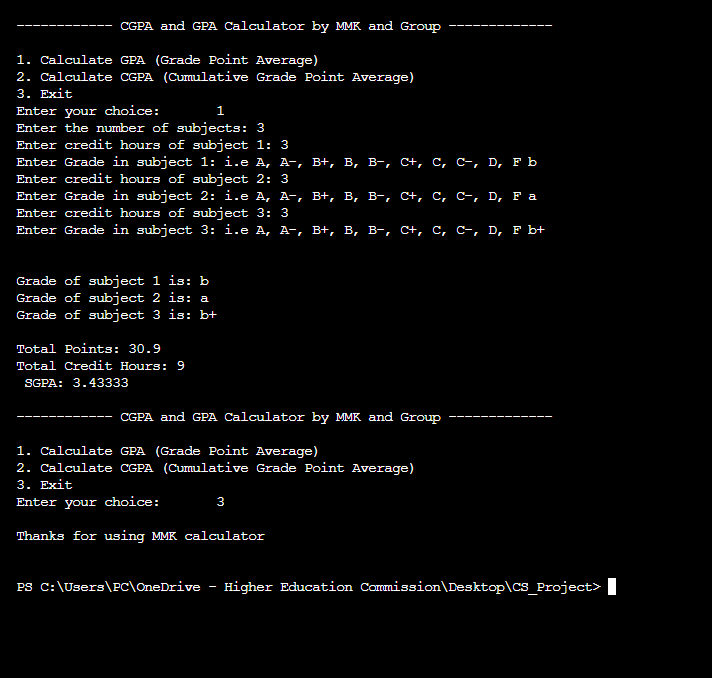
The program implements control flow structures such as the **switch** statement and **do-while** loop to create a user-friendly menu-driven interface. The **switch** statement allows the user to select different options, such as calculating GPA, calculating CGPA, or exiting the program. The **do-while** loop ensures that the menu is displayed repeatedly until the user chooses to exit, enhancing the program's usability and interaction with the user.

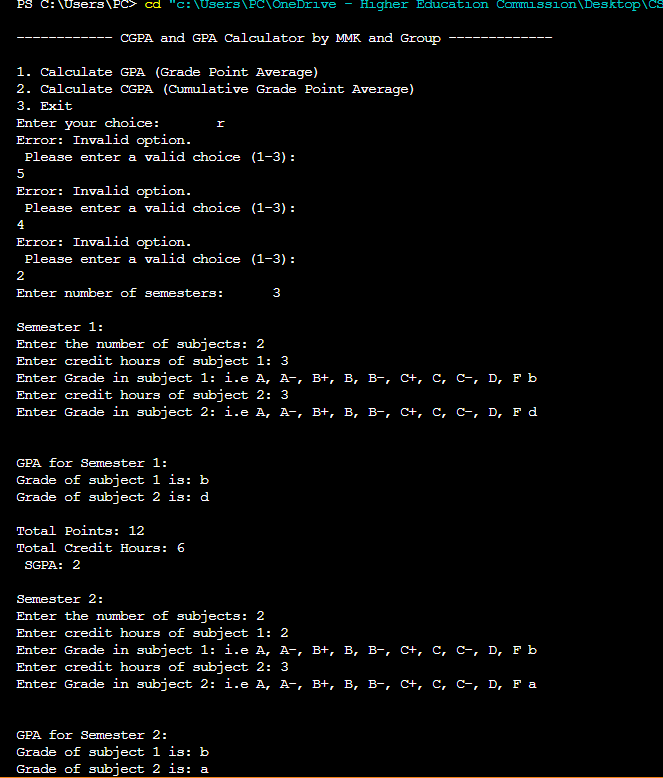
# Conclusion

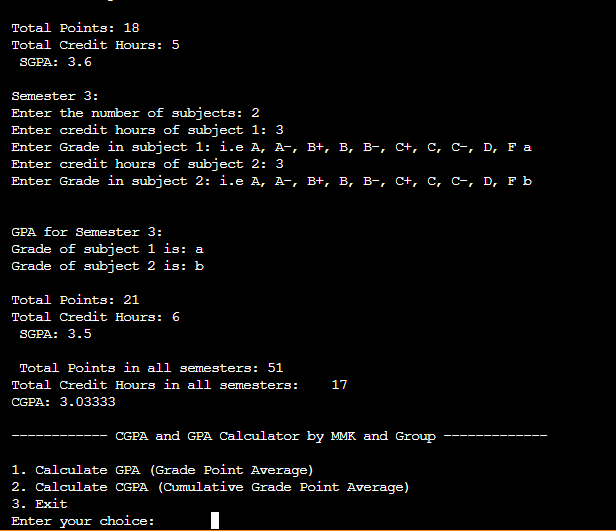
The CGPA and GPA Calculator project implemented in C++ demonstrates the use of classes, inheritance, dynamic memory allocation, file handling, and control flow structures. The code provides an intuitive and efficient solution for students to calculate their GPA and CGPA, facilitating better academic tracking and planning. The program's modular design allows for easy expansion and customization, making it a useful tool for students across various educational institutions.

# User Interface:









# Text File

