Green University of Bangladesh

Department of Computer Science and Engineering (CSE)
Semester: (Fall, Year: 2023), B.Sc. in CSE (Evening)

CGPA Analysis and Prediction

Course Title: Stracture Programming Lab
Course Code: CSE 103
Section: EB

Students Details

Name	ID
Mofasser Hossain	223015072

Submission Date: 16/06/2023
Course Teacher's Name: Fatema Akter

[For teachers use only: Don't write anything inside this box]

Lab Project Status	
Marks:	**Signature:**
Comments:	**Date:**

Contents

Chapter 1

Introduction

1.1 Overview

This project will predict user CGPA. Here is two process. One is system generated CGPA and another is user will give their input and predict their CGPA.

First user will input their previous semester completed credit and CGPA. Then it will ask user to give next semester subject and expected CGPA. and if they go for process one then they have to fill the all subject info with name credit and the mark they will get. then system will calculate and give the output

If they go for 2nd process then they just have to gave Subject info with name and credit and system will calculate the result and give output.

1.2 Motivation

In this project A student can predict their next semester CGPA. They will predict and check their CGPA.

1.2.1 Problem Definition

This project is basically for Student to calculate and Predict their CGPA will be in next semester.

1.3 Problem Statement

The CGPA Prediction System project aims to develop a program that allows users to predict their CGPA for the next semester based on their previous semester's academic performance. The system provides two processes: one where the system generates the CGPA prediction based on subject information and user inputs, and another where the user provides their own input to predict their CGPA.

Process 1: System Generated CGPA Prediction

The user inputs their previous semester's completed credit and CGPA. The system prompts the user to provide information about the subjects they will be taking in the next semester, including subject name and credit. The user enters the expected CGPA for each subject. The system calculates the predicted CGPA based on the user's inputs and provides the output.

Process 2: User Input CGPA Prediction

The user provides information about the subjects they will be taking in the next semester, including subject name and credit. The system prompts the user to enter the expected marks for each subject. The system calculates the predicted CGPA based on the user's inputs and provides the output. Inputs:

Process 1:

Previous semester's completed credit (float value) Previous semester's CGPA (float value) Next semester subject information: Subject name (string) Credit (float value) Marks (float value)

Process 2:

Next semester subject information: Subject name (string) Credit (float value) Outputs:

Predicted CGPA (float value) based on the user's inputs. Requirements:

The program should validate the user's inputs, ensuring they are within acceptable ranges (e.g., credit values are positive, marks is within the valid range). Suitable algorithms, such as weighted average or regression, should be implemented to calculate the predicted CGPA based on the user's inputs. The program should provide an interactive user interface that guides the user through the input process and displays the predicted CGPA. The system-generated CGPA prediction process should consider both the user's previous semester's performance and their expected CGPA for the upcoming subjects. The program should handle any potential errors or exceptions during the execution and provide appropriate error messages to the user. The program can include additional features, such as data storage for user profiles, visual representations of the predicted CGPA, or a comparison of the predicted CGPA with the user's desired target CGPA.

1.3.1 Complex Engineering Problem

For this project I have faced a problem to calculate CGPA. Like

Process 1: In process one i have faced issue with calculate CPGA. like if they want to get more CGPA than the previous semester then i have to calcualte the actual cgpa to get the final result.

Process 2: Since it will just generate their cgpa in each subject. So if they want to get more cgpa then their marks might be grater than 100. But it can't be. So i have to added a validation for it.

Table 1.1: Summary of the attributes touched by the mentioned projects

Name of the P Attributess	**Explain how to address**
P1: Depth of knowledge required	IDE(EXM: Code Blocks) and C Programming language(Array, loop, conditional statement etc)
P2: Depth of analysis required	System Architecture design, Data Management, PIN for security and Requirements Analysis
P3: Extent of stakeholder involvement and conflicting requirements	User friendly interface user can easy find all option. Store user cash transition using array. change pin any type of user can easy understand.

1.4 Design Goals/Objectives

Final goal is this project are

1. Student can predict their cgpa.
2. They can predict so they can know what will be their final result.
3. Studnet can check wheither they can achieve their cpga or not.

1.5 Application

Key Feature of this project:

1. Prediction: Student can predict their cgpa by entering their next semester each subject mark they will get.

2. Analysis: Here student will set their expected CGPA. so they will check they cam achieve or not. but if they can't how much they can at least get in semester.

Chapter 2

Design/Development/Implementation of the Project

2.1 Introduction

This project is used for the user to calculate their CGPA and predict. User will input their previous and expected CGPA. So the program will run and give them output.

2.2 Project Details

Introduction: The CGPA Analysis and Prediction project is designed to help students analyze and predict their Cumulative Grade Point Average (CGPA) for the upcoming semester. It takes into account the student's completed credits, current CGPA, expected CGPA for the next semester, and the subjects they plan to take.

Objective: The main objective of this project is to provide students with an analysis of their current academic standing and predict their CGPA based on their expected performance in the next semester. By utilizing the subject details and credit information, the project helps students set realistic goals and understand the required effort to achieve their desired CGPA.

Functionality: a) Input:

Completed credits: The user is prompted to enter the number of credits they have successfully completed so far. Current CGPA: The user provides their current CGPA. Next semester subjects: The user enters the number of subjects they plan to take in the next semester, along with their respective details (subject name, credit, and marks). Expected CGPA: The user specifies their desired CGPA for the upcoming semester. b) Calculation:

The project calculates the total completed credit by adding the completed credits and the total credit in the next semester. It calculates the expected credit point by multiplying the completed credit with the expected CGPA. The previous credit point is calculated by multiplying the completed credits with the current CGPA. The target credit point is determined by subtracting the previous credit point from the expected credit point. c)

Analysis:

Based on the process chosen by the user (self-prediction or system prediction), the project provides different analysis outputs. If the user chooses self-prediction, the project calculates the achieved credit point and CGPA based on the subject marks entered by the user. It then compares the achieved credit point with the expected credit point and provides analysis on whether the student is on track to achieve their desired CGPA. If the user chooses system prediction, the project calculates the predicted CGPA based on the target credit point and total credit in the next semester. It then compares the predicted CGPA with the expected CGPA and provides analysis on whether the desired CGPA is achievable. d) Output:

The project displays the user's current CGPA, expected CGPA, and the achieved or predicted CGPA. It provides analysis and recommendations based on the comparison between the achieved/predicted CGPA and the expected CGPA. The project also displays the subject-wise results, including subject names, credits, marks (if available), and corresponding CGPA. Implementation: The project is implemented using the C programming language. It utilizes data structures, such as structures and arrays, to store subject details. The program incorporates input validation to ensure proper data entry. Functions are used to modularize the code and perform specific tasks, such as calculating credit points, CGPA, and analyzing results.

Conclusion: The CGPA Analysis and Prediction project aims to assist students in understanding their academic progress and predicting their CGPA for the upcoming semester. By providing analysis and recommendations, it helps students set realistic goals and make informed decisions to achieve their desired CGPA. The project's implementation in C ensures efficiency and accuracy in calculating and analyzing the data.

2.3 Implementation

Here is the code examples.

The Head

```c
#include <stdio.h>

#define MAX_SUBJECTS 10

typedef struct {
    char name[50];
    float credit;
    int marks;
    float cgpa;
} Subject;

// calculate total point in cgpa
void calculateTPoint(Subject subjects[], int numSubjects, float *cgpa);
// calculate cgpa
float calculateCgpa(float achievePoint, float prevAchievePoint, float totalCompletedCredit);
// print subject name reults
void printSubjectResult(Subject subjects[], int numSubjects);
// get CGPA from marks
float getCGPAFromMark(float mark);
// get CGPA from marks
int getMakFromCGPA(float cgpa);
// store data
void storeDataFromUser(Subject subjects[], int numSubjects, float *nextSemesterCredit, int process);
// store data
void storeDataFromUserTwo(Subject subjects[], int numSubjects, float *nextSemesterCredit);
// process one
void processOne(Subject nextSemesterSubjects[], int numNextSemesterSubjects, float previousCreditPoint, float completedCredit, float currentCGPA,
float expectedCGPA, float expectedCreditPoint);
// process two
void processTwo(float targetCreditPoint, float totalCreditInNextSemester, float previousCreditPoint, float completedCredit, float currentCGPA,
float expectedCGPA);

```

The main function

```c
int main() {
    float previousCompletedCredit;
    float currentCGPA;

    printf("****************************************\n");
    printf("*                                      *\n");
    printf("*                                      *\n");
    printf("*    CGPA Analysis and Prediction      *\n");
    printf("*                                      *\n");
    printf("*                                      *\n");
    printf("****************************************\n");

    printf("\n\n");

    printf("Enter the number of credit you have completed: ");
    scanf("%f", &previousCompletedCredit);

    while (getchar() != '\n');  // Clear the input buffer
    printf("Enter your current CGPA: ");
    scanf("%f", &currentCGPA);

    int numNextSemesterSubjects;
    Subject nextSemesterSubjects[MAX_SUBJECTS];
    float expectedCGPA;


    while (getchar() != '\n');  // Clear the input buffer
    printf("Enter the number of subjects in the next semester: ");
    scanf("%d", &numNextSemesterSubjects);

    while (getchar() != '\n');  // Clear the input buffer
    printf("Enter your expected CGPA for the next semester: ");
    scanf("%f", &expectedCGPA);

    int process;

    printf("\n\nEnter 1 to get predicted CGPA by your self: \n");
    printf("Enter 2 to get predicted CGPA by system: \n\n");

    printf("Enter the process you want to continue: \t");
    int valid = 0;
    while (!valid){
      /* code */
      scanf("%d", &process);
      if(process == 1 || process == 2)
        valid = 1;
      else
        printf("\nPlease enter a valid process number(1 or 2): \t");
    }

    float totalCreditInNextSemester = 0.00;
    // get each subject data from user
    storeDataFromUser(nextSemesterSubjects, numNextSemesterSubjects, &totalCreditInNextSemester, process);

    // total semester credit number
    float completedCredit = totalCreditInNextSemester + previousCompletedCredit;
    // expected credit point in next semester;
    float expectedCreditPoint = completedCredit * expectedCGPA;
    // previous credit point
    float previousCreditPoint = previousCompletedCredit * currentCGPA;
    // targeted credit point
    float targetCreditPoint = expectedCreditPoint - previousCreditPoint;

    switch(process){
        case 1:
            processOne(nextSemesterSubjects, numNextSemesterSubjects, previousCreditPoint, completedCredit, currentCGPA, expectedCGPA,
            expectedCreditPoint);
            break;
        case 2:
            processTwo(targetCreditPoint, totalCreditInNextSemester, previousCreditPoint, completedCredit, currentCGPA, expectedCGPA);
            break;
        default:
            processTwo(targetCreditPoint, totalCreditInNextSemester, previousCreditPoint, completedCredit, currentCGPA, expectedCGPA);
            break;
    }
    printf("\n\n\n");

    return 0;
}
```

Process One

```
void processOne(Subject nextSemesterSubjects[], int numNextSemesterSubjects, float previousCreditPoint, float completedCredit, float currentCGPA,
float expectedCGPA, float expectedCreditPoint) {
    float achieveCGPAPoint;
    float predictedCGPA;
    calculateTPoint(nextSemesterSubjects, numNextSemesterSubjects, &achieveCGPAPoint);

    printf("\n achieve CGPA Point %f", achieveCGPAPoint);
    float achieveCGPA = calculateCgpa(achieveCGPAPoint, previousCreditPoint, completedCredit);

    printf("\n\n\n——— CGPA Analysis and Prediction Result ———\n\n\n");
    printf("Current CGPA: %.2f\n", currentCGPA);
    printf("Expected CGPA: %.2f\n", expectedCGPA);
    printf("Achieved CGPA: %.2f\n\n", achieveCGPA);

    if (achieveCGPAPoint ≥ expectedCreditPoint) {
        printf("Congratulations! You are on track to achieve your expected CGPA.\n\n");
    } else {
        printf("You need to work harder to achieve your expected CGPA.\n\n");
    }

    printSubjectResult(nextSemesterSubjects, numNextSemesterSubjects);
}
```

Process Two

```


void processTwo(float targetCreditPoint, float totalCreditInNextSemester, float previousCreditPoint, float completedCredit, float currentCGPA,
float expectedCGPA){
    // check predicted CGPA
    float predictedCGPA = targetCreditPoint / totalCreditInNextSemester;

    printf("\n\n\n——— CGPA Analysis and Prediction Result ———\n\n\n");
    printf("Current CGPA: %.2f\n", currentCGPA);
    printf("Expected CGPA: %.2f\n\n\n", expectedCGPA);

    if (predictedCGPA ≤ 4){
        printf("To achieve your expected CGPA,\n");
        int marks = getMakFromCGPA(predictedCGPA);
        printf("you have to get average CGPA: %.2f or %d%% above marks in each subject\n", predictedCGPA, marks);
    } else {
        float finalCPGA = ((totalCreditInNextSemester * 4) + previousCreditPoint) / completedCredit;
        printf("Sorry! Your expected CGPA is too high. You can't achieve it.\nBut if you get 80%% marks in each subject,\nYou can achieve Maximum
        CGPA: %.2f\n", finalCPGA);
    }
}


```

Calculation Function

```


void calculateTPoint(Subject subjects[], int numSubjects, float *achievedCgpa) {
    int totalCredits = 0;
    for (int i = 0; i < numSubjects; i++) {
        float achiveCredit = getCGPAFromMark(subjects[i].marks);
        totalCredits += subjects[i].credit * achiveCredit;
        subjects[i].cgpa = achiveCredit;
    }
    *achievedCgpa = totalCredits;
}


float calculateCgpa(float achievePoint, float prevAchievePoint, float totalCompletedCredit) {
    return (achievePoint + prevAchievePoint) / totalCompletedCredit;
}


void printSubjectResult(Subject subjects[], int numSubjects){
    printf("Your output in each subject \n \n");
    printf("%-25s%-20s%-10s%-10s\n", "Subject Name", "Credit", "Marks", "CGPA");
    for (int i = 0; i < numSubjects; i++) {
       printf("%-25s%-20.2f%-10d%-10.2f\n", subjects[i].name, subjects[i].credit, subjects[i].marks, subjects[i].cgpa);
    }
}


```

User Data

```c

void storeDataFromUser(Subject subjects[], int numSubjects ,float *nextSemesterCredit,int process){
    printf("\n\n\nEnter details for each subject in the next semester:\n\n\n");
    for (int i = 0; i < numSubjects; i++) {
        printf("Subject %d:\n", i + 1);

        while (getchar() != '\n');  // Clear the input buffer
        printf("   Name: ");
        scanf("%s", subjects[i].name);

        while (getchar() != '\n');  // Clear the input buffer
        float credit = 0.0;
        printf("   Credit: ");
        float cred = scanf("%f", &credit);
        if (cred != 1) {
            printf("Invalid input. Please enter a valid credit.\n");
            i--;
            continue;
        }

        subjects[i].credit = credit;
        *nextSemesterCredit += credit;

        if(process == 1){
            while (getchar() != '\n');  // Clear the input buffer
            printf("   Marks (out of 100): ");
            int marks;
            int result = scanf("%d", &marks);
            if (result != 1) {
                printf("Invalid input. Please enter a valid marks.\n");
                i--;
                continue;
            }
            // printf("\n %d", result);
            if (marks <= 0 || marks >= 100) {
                printf("Invalid marks. Please enter an marks between 1 and 100.\n");
                i--;
                continue;
            }
            subjects[i].marks = marks;
        }
    }
}

```

Convert Mark to CGPA

```c
float getCGPAFromMark (float mark){
    float result = 0.00;
    if(mark >= 80){
        result = 4.00;
    }else if(mark >= 75){
        result = 3.75;
    }else if(mark >= 70){
        result  = 3.50;
    }else if(mark >= 65){
        result  = 3.25;
    }else if(mark >= 60){
        result  = 3.00;
    }else if(mark >= 55){
        result  = 2.75;
    }else if(mark >= 50){
        result  = 2.50;
    }else if(mark >= 45){
        result  = 2.25;
    }else if(mark >= 40){
        result  = 2.00;
    }else{
        result = 0.00;
    }
    return result;
}

int getMakFromCGPA (float cgpa){
    float mark = 0;
      if(cgpa >= 4){
        mark = 80;
    }else if(cgpa >= 3.75){
        mark = 75;
```

Convert CGPA to mark

```
int getMarkFromCGPA (float cgpa){
    float mark = 0;
      if(cgpa >= 4){
         mark = 80;
    }else if(cgpa >= 3.75){
         mark = 75;
    }else if(cgpa >= 3.50){
         mark = 70;
    }else if(cgpa >= 3.25){
         mark  = 65;
    }else if(cgpa >= 3){
         mark  = 60;
    }else if(cgpa >= 2.75){
         mark  = 55;
    }else if(cgpa >= 2.50){
         mark  = 50;
    }else if(cgpa >= 2.25){
         mark  = 45;
    } else{
         mark  = 40;
    }
    return mark;
}

```

Tools and libraries

IDE(Code Blocks) and C language

2.4 Algorithms

The algorithms and the programming codes in detail should be included .
Pseudo-codes are also encouraged very much to be included in this chapter for your project.

Algorithm 1: CGPA Analysis and Prediction

Input: Prevous credit
Input: current CGPA
Input: num Next Semester Subjects
Input: expected CGPA
Input: process

switch *process* **do**
  case *1* **do**
    processOne();
  case default **do**
    processTwo();

Function *processOne()*:
  for $i \leftarrow 0$ **to** *numNextSemesterSubjects* **do**
    Input: Subject name
    Input: Subject credit
    Input: Subject marks
    Input: Subject cgpa
  Calculation: Calculate CGPA from marks; **Output:** Current CGPA, Expected CGPA, Achieved CGPA
  if *achieveCGPAPoint* $\geq$ *expectedCreditPoint* **then**
    Output: You have achieved your expected CGPA.
  else
    Output: You have not achieved your expected CGPA.
  Output:

Subject Name	Credit	Marks	CGPA
M	3.00	90	4.00

Function *processTwo()*:
  Calculation: Calculate CGPA from marks; **Output:** Current CGPA, Expected CGPA, Achieved CGPA
  if *achieveCGPAPoint* < 4 **then**
    Output: To achieve your expected CGPA, you have to get average CGPA: *averageCGPA* or % above marks in each subject.
  else
    Output: Sorry! Your expected CGPA is too high. You can't achieve it. But if you get 80% marks in each subject, you can achieve maximum CGPA: *maxCGPA*.

Chapter 3

Performance Evaluation

3.1 Results Testing

Process One output will look like

```
----- CGPA Analysis and Prediction Result -----


Current CGPA: 3.61
Expected CGPA: 3.90
Achieved CGPA: 3.77

You need to work harder to achieve your expected CGPA.

Your output in each subject

Subject Name            Credit              Marks     CGPA
M                       3.00                90        4.00
S                       3.00                90        4.00
C                       3.00                80        4.00
```

Process Two output will look like

```
Enter your current CGPA: 3.61
Enter the number of subjects in the next semester: 3
Enter your expected CGPA for the next semester: 3.90


Enter 1 to get predicted CGPA by your self:
Enter 2 to get predicted CGPA by system:

Enter the process you want to continue:         2



Enter details for each subject in the next semester:


Subject 1:
   Name: M
   Credit: 3
Subject 2:
   Name: SP
   Credit: 2
Subject 3:
   Name: CH
   Credit: 3



----- CGPA Analysis and Prediction Result -----


Current CGPA: 3.61
Expected CGPA: 3.90


Sorry! Your expected CGPA is too high. You can't achieve it.
But if you get 80% marks in each subject,
You can achieve Maximum CGPA: 3.76
```

3.2 Results Overall Discussion

As we can see you have gotten the exact output as we expected from the code. It gives the predicted CGPA to the user.

Chapter 4

Conclusion

4.1 Discussion

This project is only for getting CGPA prediction and analysis. Although there are some limitations to this project. But you can check and predict your CGPA.

4.2 Limitations

1. This is just a one-time process.
2. Data will not store in a file.
3. Students can check only one process at a time.

4.3 Scope of Future Work

1. Data store in the file system.
2. User login system.
3. Multi-process at a time

References