Title Page

Title Of Project
Assignment.c

Name
Pritam Gurung

Candidate Number
279172

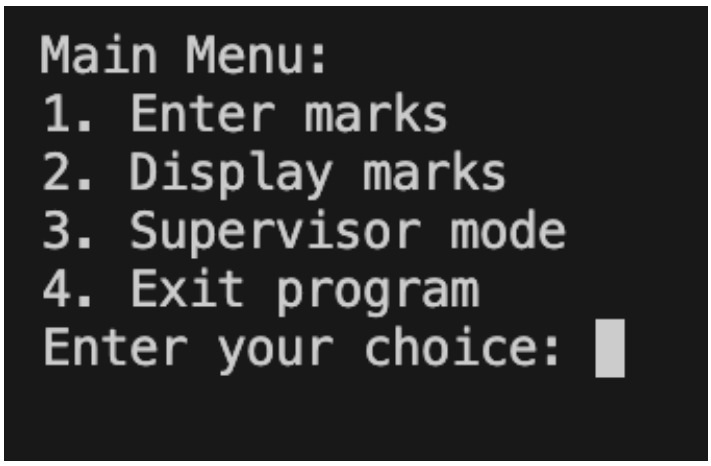
Submission Data
23/11/23

Content Page

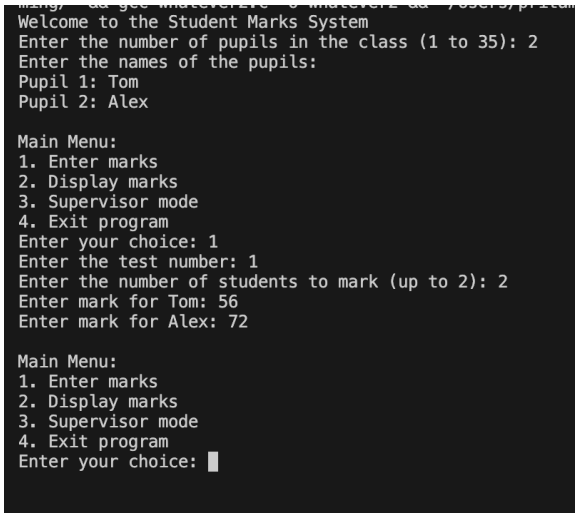
1. Content Page……Page 2
2. Program Design……Page 3
2.1 User-friendly Screen layouts… Page 3
2.2 Data Storage Array Design……Page 4
2.3 Skeleton Code…… Page 5
3. C Programming Code…… Page 9
4. Proof of testing……Page 20
5. Written Evaluation……Page 24

Program Design

2.1



This design is complex and very simple for the teacher to use which is the main menu of my program designed for managing student data. The program is an application that can be used by various teachers to handle various classroom-related tasks as shown in my screenshot of my program.



Display mark options allow the teacher to view the list of students' marks after data of each student has been put in. Supervisor mode is designed for teachers to allow and manipulate such as adding students, altering marks, and changing pins. Exit program is an option which closes this application.

The Enter marks option allows the teacher to input Students' marks after a name has been given which is given before the main menu starts and asks for the number of students to be marked. As shown above, I've named my supposed two students and was able to access the menu and able to enter the marks of two students.

By clicking on the available number options as shown above, a teacher can use various services provided by my program to execute certain tasks. This allows the teacher to manipulate data and make any necessary changes if a teacher had a mistake on inputting students' data.

2.2 Data storage and array design.

Student Names Array.

‘StudentNames’

This array holds the names of students, and each row corresponds to a different student and each column within that row represents a single character of that student's name. The length of the student's name is defined by STUDENT_MAX_NAME_LENGTH.

Marks Array.

‘marks’

This array stores the marks of students. It’s been implemented so that each row corresponds to a particular test and each column responds to a student’s mark for that test. The ‘TESTS” constant defines the number of tests, and the ‘STUDENT’ defines the total number of students.

Data Organization with Arrays.

‘Int studentNames’, ‘int currentPin’, ‘int marks’

These arrays are able to secure and handle data in the system and allow teachers in the main menu hub to input students' data, which makes it possible for the teacher to organise and manipulate students' data.

At the start, each student name is initialized using a loop that assigns “student ‘i+1’” to the ‘i-th’ row ensuring that each student is uniquely identified until names have been inputted.

‘STUDENTS’ a pre-processor defining an integer constant. Has a value of 35 as it defines the maximum number of students that the array and program can store.

‘TESTS’, a pre-processor macro defining an integer constant. The value includes 10, which the program will record marks for each student.

‘PIN’, again a pre-processor which has a value of 3750, which allows the teacher to be granted access to supervisor mode, which can also be changed during the program.

‘STUDENT_MAX_NAME_LENGTH’ a pre-processor which has a value of 50, personally the value is quite high, and it can be lowered but it helps define the maximum number of characters allowed for a student’s name.

2.3 Skeleton Code

```
/*****************************************************************************
**
 * assignment.c
 * A program designed to provide assistance to teachers by
providing the tools
 * to manage, store and make adjustments student's data.
 * Pritam Gurung
 *  * Version: a

*****************************************************************************
*/



#include <stdio.h>
#include <string.h>

#define STUDENTS 35
#define TESTS 10
#define PIN 3750
#define STUDENT_MAX_NAME_LENGTH 50

// Global variables
int marks[TESTS][STUDENTS];
int currentStudents;
int currentPIN;
```

```
char
studentNames[STUDENTS][STUDENT_MAX_NAME_LENGTH];

// All the Function declarations
void initializeStudentNames();
void enterMarks();
void displayMarks();
void changeMark();
void addingStudent();
void changePin();
void correctingStudentName();
void supervisorMode();
int main();

/* This puts student names in placeholders */
/* Stores and manages stduent names*/
void initializeStudentNames()
{
    // Function implemented here...
}

/* This allows the user to enters marks for a specific test for the
students */
/* Allowing teachers to enter the marks*/
void enterMarks()
{
    // Function to be implemented here...
}

/* This displays marks for all students */
void displayMarks()
```

```
{
    // Function to be implemented here...
}

/* This changes a mark for a specific student and test */
void changeMark()
{
    // Function to be implemented to be here...
}

/* This adds a new student to the system */
void addingStudent()
{
    // Function to be implemented here...
}

/* This changes the PIN */
void changePin()
{
    // Function to be implemented here...
}

/* This corrects a student's name in the system */
void correctingStudentName()
{
    // Function to be implemented here...
}

/* This is the supervisor mode allowing teachers to manage and
adjusts students data */
void supervisorMode()
```

```
{
    // Function to be implemented here...
}

/* Main program system */
int main()
{
    // Function to be implemented here...
}
```

3. C programming Code

```c
/*****************************************************************************
**
 * assignment.c
 * A program designed to provide assistance to teachers with tools
to manage,
 * store and adjust student's data.
 * Pritam Gurung
 * Version: a

*****************************************************************************
*/

#include <stdio.h>
#include <string.h>

#define STUDENTS 35
#define TESTS 10
#define PIN 3750
#define STUDENT_MAX_NAME_LENGTH 50

// Global variables
int marks[TESTS][STUDENTS] = {0};
int currentStudents = 0;
int currentPIN = PIN;
char
studentNames[STUDENTS][STUDENT_MAX_NAME_LENGTH];

/* Puts student names in placeholders */
void initializeStudentNames()
{
    char name[STUDENT_MAX_NAME_LENGTH];
```

```
    int totalStudents;

    printf("Enter the number of pupils in the class (1 to 35): ");
    scanf("%d", &totalStudents);
    while (getchar() != '\n'); // Clear input issue

    if (totalStudents < 1 || totalStudents > STUDENTS)
    {
        printf("Invalid number. Please enter a number between 1 and
35.\n");
        return;
    }

    currentStudents = totalStudents;
    printf("Enter the names of the pupils:\n");
    for (int i = 0; i < totalStudents; i++)
    {
        printf("Pupil %d: ", i + 1);
        fgets(name, STUDENT_MAX_NAME_LENGTH, stdin);
        name[strcspn(name, "\n")] = 0; // Remove newline character
        strcpy(studentNames[i], name);
    }
}

/* Enters marks for a specific test for the students */
void enterMarks()
{
    int testNumber, i, mark, numberOfStudentsToMark;

    printf("Enter the test number: ");
    scanf("%d", &testNumber);
```

```
    testNumber--;

    if (testNumber < 0 || testNumber >= TESTS)
    {
        printf("Invalid test number.\n");
        return;
    }

    printf("Enter the number of students to mark (up to %d): ",
currentStudents);
    scanf("%d", &numberOfStudentsToMark);

    while (getchar() != '\n'); // Clear input issue

    if (numberOfStudentsToMark < 1 || numberOfStudentsToMark >
currentStudents)
    {
        printf("Invalid number of students. Please enter a number
between 1 and %d.\n", currentStudents);
        return;
    }

    for (i = 0; i < numberOfStudentsToMark; i++)
    {
        printf("Enter mark for %s: ", studentNames[i]);
        scanf("%d", &mark);

        while (getchar() != '\n'); // Clear input issue

        if (mark < 0 || mark > 100) {
```

```
            printf("Invalid mark. Please enter a value between 0 and
100.\n");
            i--; // Decrement to repeat input for specific student
            continue;
        }

        marks[testNumber][i] = mark;
    }
}

/* This isplays marks for all students */
void displayMarks()
{
    float sum, average;

    printf("Displaying marks.\n");
    for (int i = 0; i < currentStudents; i++)
    {
        sum = 0;
        printf("Student %d (%s): ", i + 1, studentNames[i]);
        for (int j = 0; j < TESTS; j++) {
            printf("%d ", marks[j][i]);
            sum += marks[j][i];
        }
        average = sum / TESTS;
        printf("| Average: %.2f\n", average);
    }
    printf("\nPress <enter> to return to the main menu.\n");
    while (getchar() != '\n'); // Clear input issue
}
```

```c
/* This changes a mark for a specific student and test */
void changeMark()
{
    int testNumber, studentNumber, newMark;

    printf("Enter test number (1-%d): ", TESTS);
    scanf("%d", &testNumber);
    testNumber--;

    printf("Enter student number (1-%d): ", currentStudents);
    scanf("%d", &studentNumber);
    studentNumber--;

    printf("Enter new mark for %s in test %d: ",
studentNames[studentNumber], testNumber + 1);
    scanf("%d", &newMark);

    marks[testNumber][studentNumber] = newMark;
    printf("Mark has been updated.\n");
}

/* This adds a new student to the system */
void addingStudent()
{
    char name[STUDENT_MAX_NAME_LENGTH];

    if (currentStudents >= STUDENTS)
    {
        printf("Cannot add more students. Maximum limit has been
reached.\n");
        return;
```

```
    }

    printf("Enter the name for the new student: ");
    fgets(name, STUDENT_MAX_NAME_LENGTH, stdin);
    name[strcspn(name, "\n")] = 0; // Remove newline character
    strcpy(studentNames[currentStudents], name);

    for (int i = 0; i < TESTS; i++)
    {
        marks[i][currentStudents] = 0;
    }

    currentStudents++;
    printf("New Student '%s' has been added. Total Students: %d\n",
name, currentStudents);
}

/* This changes the PIN */
void changePin()
{
    int newPIN;

    printf("Please enter a new PIN: ");
    scanf("%d", &newPIN);

    if (newPIN >= 1000 && newPIN <= 9999)
    {
        currentPIN = newPIN;
        printf("PIN has been changed successfully.\n");
    } else
    {
```

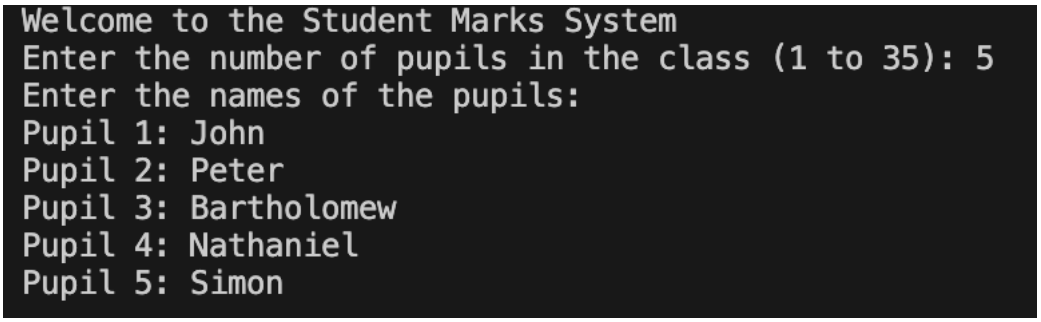
```
        printf("Invalid PIN. Enter a 4-digit number.\n");
    }
}

/* This corrects a student's name in the system */
void correctingStudentName()
{
    int studentNumber;
    char newName[STUDENT_MAX_NAME_LENGTH];

    printf("Enter student number (1-%d): ", currentStudents);
    scanf("%d", &studentNumber);
    studentNumber--;

    if (studentNumber < 0 || studentNumber >= currentStudents)
    {
        printf("Invalid student number. Try again.\n");
        return;
    }

    printf("Enter a new name for %s: ",
studentNames[studentNumber]);
    fgets(newName, STUDENT_MAX_NAME_LENGTH, stdin);
    newName[strcspn(newName, "\n")] = 0; // Remove newline
character
    strcpy(studentNames[studentNumber], newName);
    printf("Student's name has been updated to '%s'.\n", newName);
}

/* This is a supervisor mode, granting privilage of managing and
storing students data accesed only by teachers */
```

```
void supervisorMode()
{
    int pin, choice;

    printf("Enter PIN: ");
    scanf("%d", &pin);

    if (pin != currentPIN)
    {
        printf("Incorrect PIN.\n");
        return;
    }

    do
    {
        printf("\nSupervisor Menu:\n");
        printf("1. Change PIN\n");
        printf("2. Add a student\n");
        printf("3. Change a mark\n");
        printf("4. Correct a student's name\n");
        printf("5. Return to main menu\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice)
        {
            case 1:
                changePin();
                break;
            case 2:
                addingStudent();
```

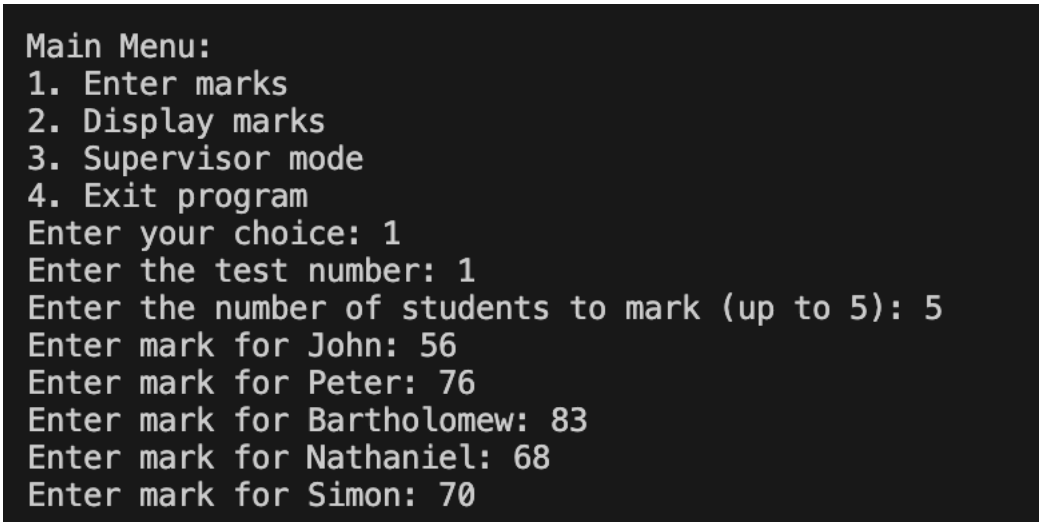
```
            break;
        case 3:
            changeMark();
            break;
        case 4:
            correctingStudentName();
            break;
        case 5:
            // break out of the loop to return to main menu
            break;
        default:
            printf("Invalid choice. Please try again.\n");
    }
  } while (choice != 5);
}

/* The main programme, introduces to main menu */
int main()
{
  int choice;

  printf("Welcome to the Student Marks System\n");
  initializeStudentNames(); // student names at the start

  do
  {
    printf("\nMain Menu:\n");
    printf("1. Enter marks\n");
    printf("2. Display marks\n");
    printf("3. Supervisor mode\n");
    printf("4. Exit program\n");
```

```
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice)
        {
            case 1:
                enterMarks();
                break;
            case 2:
                displayMarks();
                break;
            case 3:
                supervisorMode();
                break;
            case 4:
                printf("Exiting program...\n");
                break;
            default:
                printf("Invalid choice. Please try again.\n");
        }
    } while (choice != 4);

    return 0;
}
```

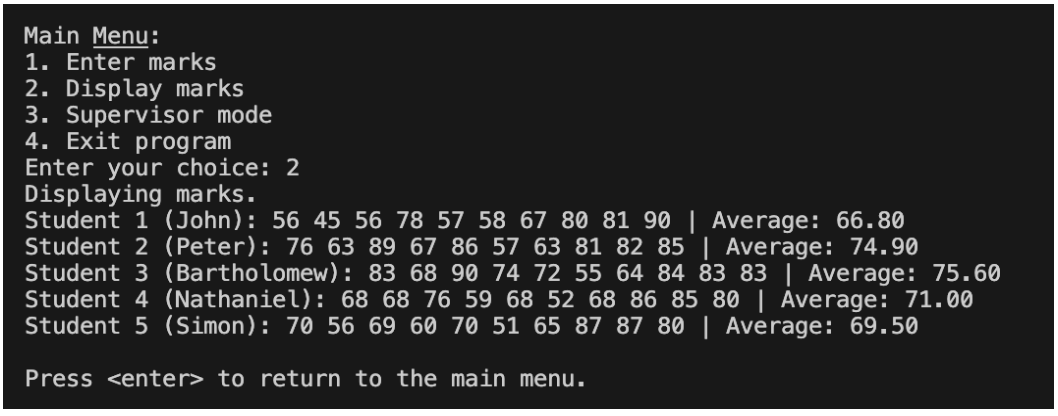
Proof Of Testing (6 Screenshots)



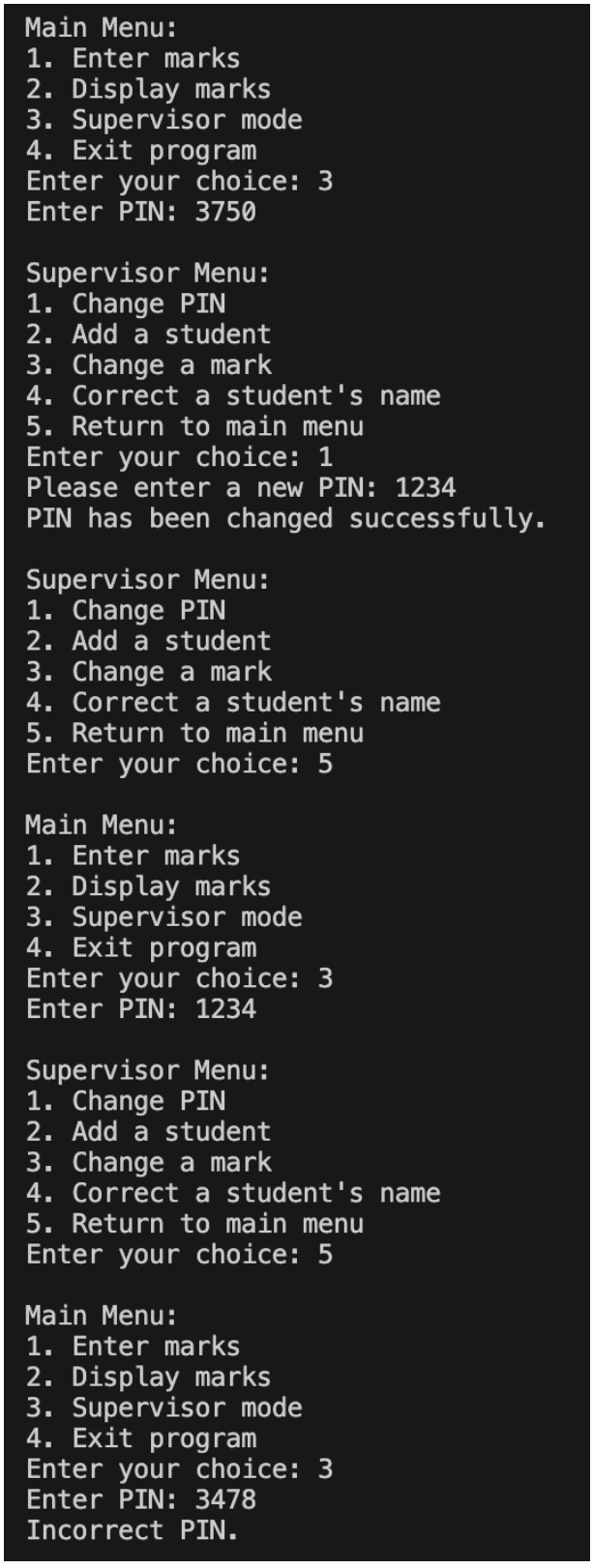
In this 1st screenshot, the teachers are greeted with a warm welcome and are asked to enter the number of pupils in class giving it an option from 1 to 35. After inputting the number of pupils, the program asks it to enter the name of the pupils.



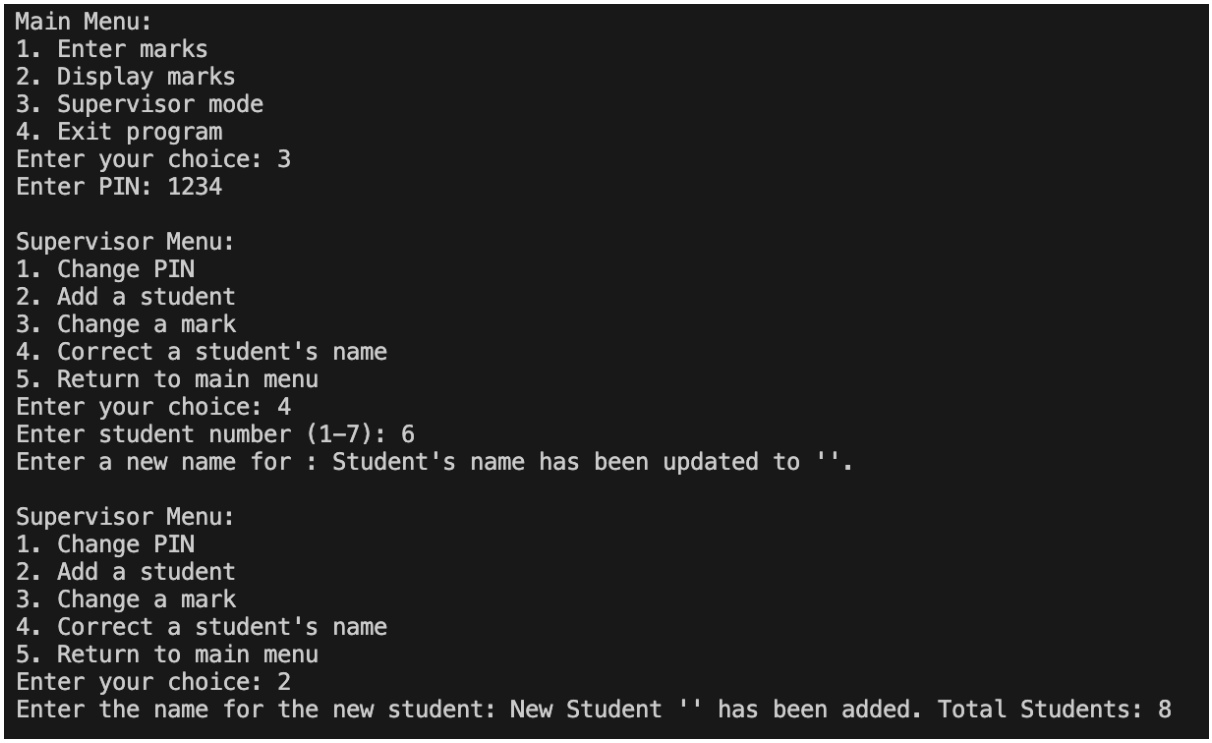
In the 2nd screenshot, the program asks the user(teacher) to enter a choice from the menu between 1 to 4. As 1 has been entered, the program asks for the test number and to enter the marks for 5 students displayed.



In the 3rd screenshot, after the user(teacher) has entered the marks, it returns the user to the main menu. When the user enters 2, it displays the marks of all the students that the user has put in. As the marks have been displayed, it also calculates the average of the marks the student has achieved. The user can hit enter to return to the main menu.



In the 4th screenshot, the user(teacher) enters 3, where it's asked to enter a PIN. After the pin has been entered, it grants access to privileged options uniquely available for the teachers. If the user wants to change the PIN, they enter 1, where it then asks the user to enter a new PIN. After a new PIN has been set, the program informs the user that the "PIN has been changed successfully." After the PIN has been changed, you can now enter the new PIN, If and is entered successfully, should grant access to the supervisor menu however, if the user enters the wrong PIN, they are denied access to the supervisor menu and a message pop up from the program stating, "Incorrect PIN." This denies access to the supervisor menu, which is designed to protect and preserve students' data.



In the 5th screenshot, the user(teacher) enters the correct PIN and is granted access to the supervisor menu. Once in the supervisor menu, the user enters 4 to correct a student's name once a student number has been put in. After that, the user enters 2 to add additional students increasing the number of students from previous 1-7 to 1-8. These additional students are now subjected to have marks once marks have been entered by the user.

In the last and final 6th screenshot of my program, the user(teacher) is granted access to the supervisor menu after entering the right PIN. The user enters 3, which allows the user to adjust any student's marks. The user is told by the program to enter the test number, and student number and asked to enter a new mark for the student. After exiting the supervisor menu, the user can now enter 2 and see the changes made of the student's mark. After everything has been done, the user enters 4 which exists the program.

Evaluation (word count 497)

My program is well-organized as my code is neatly divided into different parts and each part is doing a specific function and job. This makes it easier to understand and change things later to be implemented and perhaps to adjust when compiling it and then running it if it works. Furthermore, handling specific tasks like ‘enterMarks’ enhances readability and maintainability. Naming conventions are also consistent which aids clarity. In addition, my program checks if the user which is the teacher types something wrong and informs them to fix it however, I have used global variables, and it can lead to maintenance challenges and coding errors. Constants that have been defined using #define for values like ‘STUDENTS’ and ‘TESTS’ is easily readable and easy to modify. In terms of user interface, my program instructions and feedback are clear and understandable, which makes it easier for the teacher to use. The menu structure allows users to navigate easily between different functionality which makes the application easy to use and efficient. For security, #define PIN, can pose a real risk because anyone can access the code when it’s visible in my code and people are then able to breach in people's privacy. The current implementation lacks such security measures. This can be improved.

In terms of improvements, I’ve used global variables which is a lot easier but can also be hard to maintain as everyone can access and change. This can cause problems such as debugs issues or potential coding error. If a student happens to have a very long name, my program might not be able to handle it well, so I could improve this by making changes and allowing students longer names to be in the system. There could also be more detailed documentation and comments explaining the purpose, inputs and outputs of each function which would enhance understanding and ease of use. When running the program, especially when entering the code for supervisor mode and adding new students, I fail to name the newly added students, and this can be concerning as the teacher who uses this program will have a hard time keeping track of all the student's data. This can be improved and something I should’ve worked on. I should also try improving its memory recall capabilities, as it’s better if my program could remember the marks and names or add a simpler explanation to my code of what each part does which makes it easier for others and me to understand better and improve.

Overall, my program for student mark management demonstrates an understanding of structured programming principles with user-friendly design and clear functionality. I think it also stands out however it could be further improved by focusing on security practises, improvement on error handling, better input validation and memory management. By learning and constantly improving in coding, I can address this issue and by doing so, would significantly improve program capabilities and become suitable for real-world applications, like secure tools for educational purposes.