WORKSHEET 1

24030183

SUBMITTED BY: Sujal Adhikari
STUDENT ID: 24030183

Introduction

This is a full college project that is based on my Programming in C++ module, my goal in this project was to use object-oriented programming to solve particular real-world problems. This assessment combines creating a grading system for students alongside a Circle class with specific functions, also introducing basic file handling and error handling. There is a lot of concepts C++, which I had in the course, such as classes, objects, friend functions, function overloading, and exception handling. Through this project, I got to explore those concepts as they applied to a real-world scenario, and particularly how a structured approach to programming, certain types of input validation, and documentation of the code are both useful and necessary for building robust, efficient software solutions.

Question 1.1

Task 1: Basic student grading system prototype using classes and objects. [30 Marks]

Write a program that manages a simple student grade calculator with the following requirements. Create a Student class that has:

1. Student name (string)
2. Three subject marks (integers)
3. A basic member function to calculate average

The program should:

1. Accept student details (name and marks) from user input
2. Calculate and display:
    1. Total marks
    2. Average marks
    3. Grade (A for ≥90%, B for ≥80%, C for ≥70%, D for ≥60%, F for <60%)
3. Display a message if any mark is below 0 or above 100

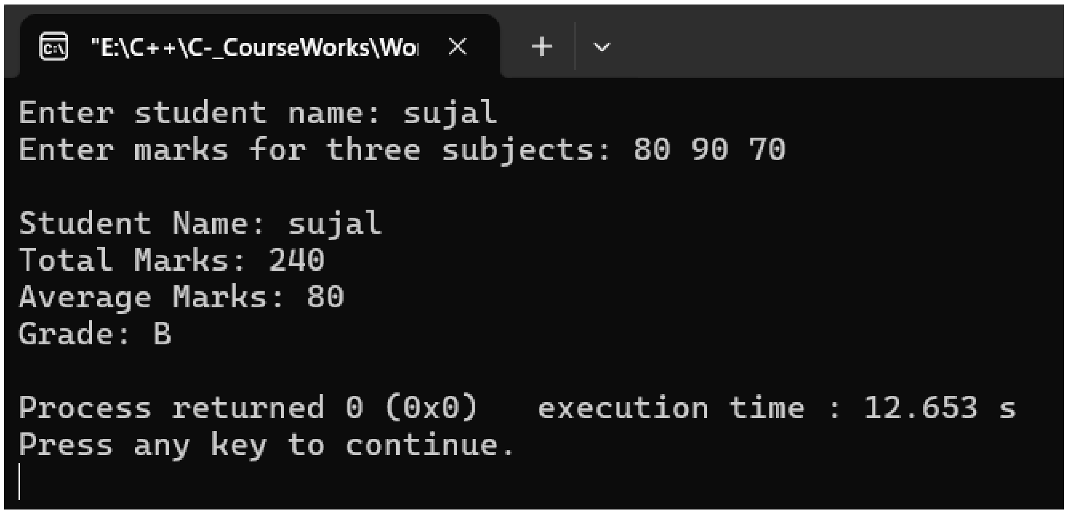
Code Implementation

```
#include <iostream>

using namespace std;

class Student { // Class to manage student details and grading

  string name; // Student name

  int marks[3]; // Array to store marks of three subjects

public:

  void getDetails() { // Function to get student details from user input

    cout << "Enter student name: ";
```

```
cout << "Invalid mark entered! Marks should be between 0 and 100." << endl;

        return;

      }

    }

  }

  int calculateTotal() { // Function to calculate total marks obtained by the student

    return marks[0] + marks[1] + marks[2];
```

```
int main() { // Main function to execute the student grading system
  Student student; // Create Student object
  student.getDetails(); // Get student details
  student.displayResults(); // Display student results
  return 0;
}
```

Output



"E:\C++\C-_CourseWorks\Wo

```
Enter student name: sujal
Enter marks for three subjects: 80 90 70

Student Name: sujal
Total Marks: 240
Average Marks: 80
Grade: B

Process returned 0 (0x0)   execution time : 12.653 s
Press any key to continue.
```

Question 2.1

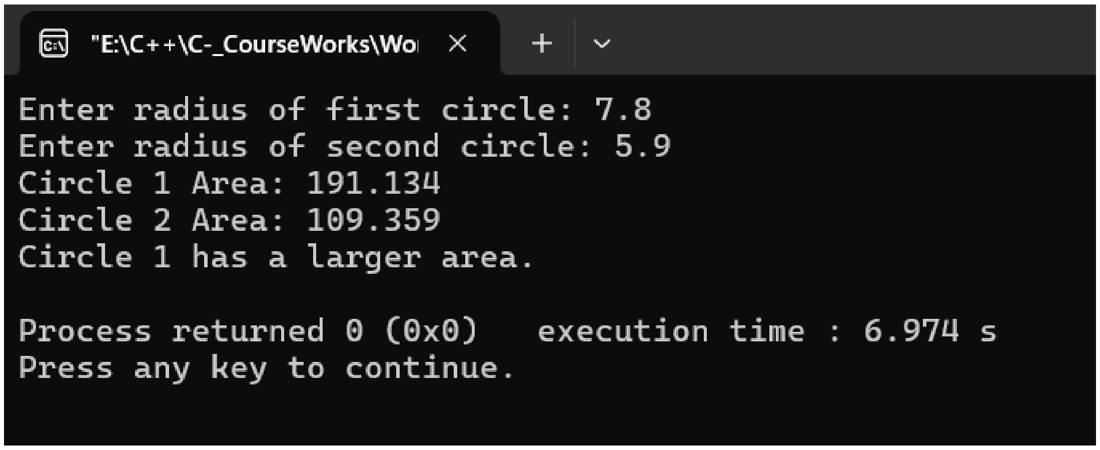
1. Write a program with a class `Circle` having:
   1. Private member: radius (float)
   2. A constructor to initialize radius
   3. A friend function `compareTwoCircles` that takes two Circle objects and prints which circle has the larger area

Code Implementation

```
#include <iostream>
using namespace std;
class Circle {
private:
  float radius;
public:
  Circle(float r) { //Constructor to initialize radius
    radius = r;
  }
  float getArea() const { //Function to calculate the area of the circle
    return 3.14159 * radius * radius;
  }
  void compareWith(const Circle& other) const { //Member function to compare two circles
    float area1 = this->getArea();
    float area2 = other.getArea();
    cout << "Circle 1 Area: " << area1 << endl;
    cout << "Circle 2 Area: " << area2 << endl;
    if (area1 > area2) {
      cout << "This circle has a larger area." << endl;
    } else if (area2 > area1) {
      cout << "The other circle has a larger area." << endl;
    } else {
      cout << "Both circles have the same area." << endl;
    }
  }
};
```

```
int main() {
float r1, r2;
cout << "Enter radius of first circle: ";
cin >> r1;
cout << "Enter radius of second circle: ";
cin >> r2;

Circle circle1(r1), circle2(r2);
circle1.compareWith(circle2); //Comparing the two circles

return 0;
}
```

Output



```
Enter radius of first circle: 7.8
Enter radius of second circle: 5.9
Circle 1 Area: 191.134
Circle 2 Area: 109.359
Circle 1 has a larger area.

Process returned 0 (0x0)   execution time : 6.974 s
Press any key to continue.
```

Question 2.2

1. Create a program with these overloaded functions named `findMax`:
    1. One that finds maximum between two integers
    2. One that finds maximum between two floating-point numbers
    3. One that finds maximum among three integers

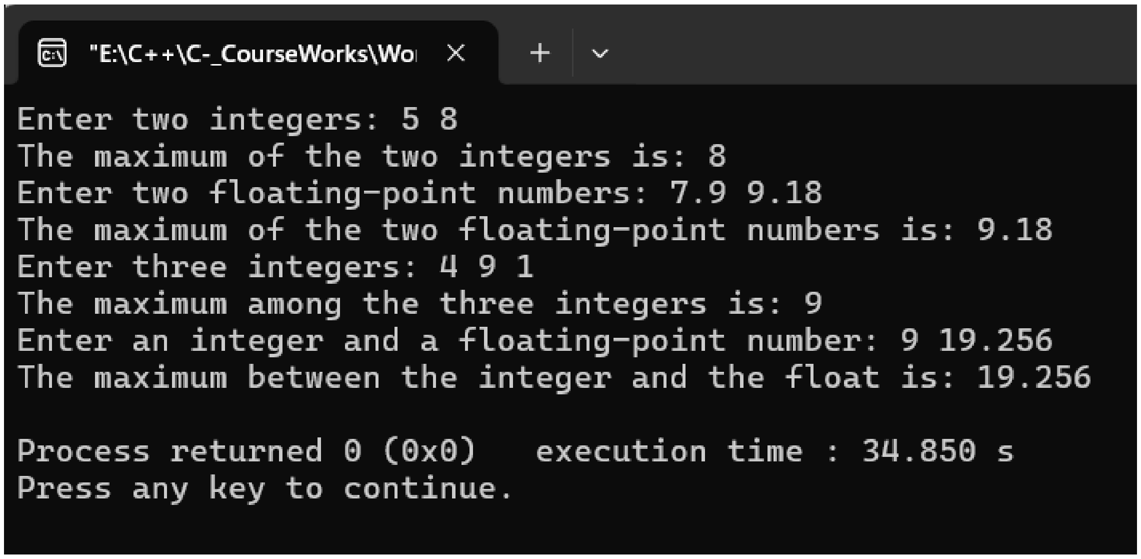
One that finds maximum between an integer and a float

Code Implementation

```
#include <iostream>

using namespace std;

class MaxFinder {

public:

int findMax(int a, int b) { //Function to find maximum between two integers

return (a > b) ? a : b; //Return the greater of the two integers

}

float findMax(float a, float b) { //Function to find maximum between two floating-point numbers

return (a > b) ? a : b; //Return the greater of the two floats

}

int findMax(int a, int b, int c) { //Function to find maximum among three integers

if (a > b && a > c) { //Compare all three integers and return the maximum

return a;

} else if (b > a && b > c) {

return b;

} else {

return c;

}

}
```

```
int main() {

MaxFinder maxFinder;  //Creating an object of MaxFinder class

int int1, int2, int3; //Declare variables to store user input

float float1, float2;

cout << "Enter two integers: ";

cin >> int1 >> int2;

cout << "The maximum of the two integers is: " << maxFinder.findMax(int1, int2) << endl;

cout << "Enter two floating-point numbers: ";

cin >> float1 >> float2;

cout << "The maximum of the two floating-point numbers is: " << maxFinder.findMax(float1, float2)
<< endl;

cout << "Enter three integers: ";

cin >> int1 >> int2 >> int3;

cout << "The maximum among the three integers is: " << maxFinder.findMax(int1, int2, int3) <<
endl;

cout << "Enter an integer and a floating-point number: ";

cin >> int1 >> float1;

cout << "The maximum between the integer and the float is: " << maxFinder.findMax(int1, float1) <<
endl;

return 0;

}
```

Output



```
Enter two integers: 5 8
The maximum of the two integers is: 8
Enter two floating-point numbers: 7.9 9.18
The maximum of the two floating-point numbers is: 9.18
Enter three integers: 4 9 1
The maximum among the three integers is: 9
Enter an integer and a floating-point number: 9 19.256
The maximum between the integer and the float is: 19.256

Process returned 0 (0x0)   execution time : 34.850 s
Press any key to continue.
```

Question 3.1

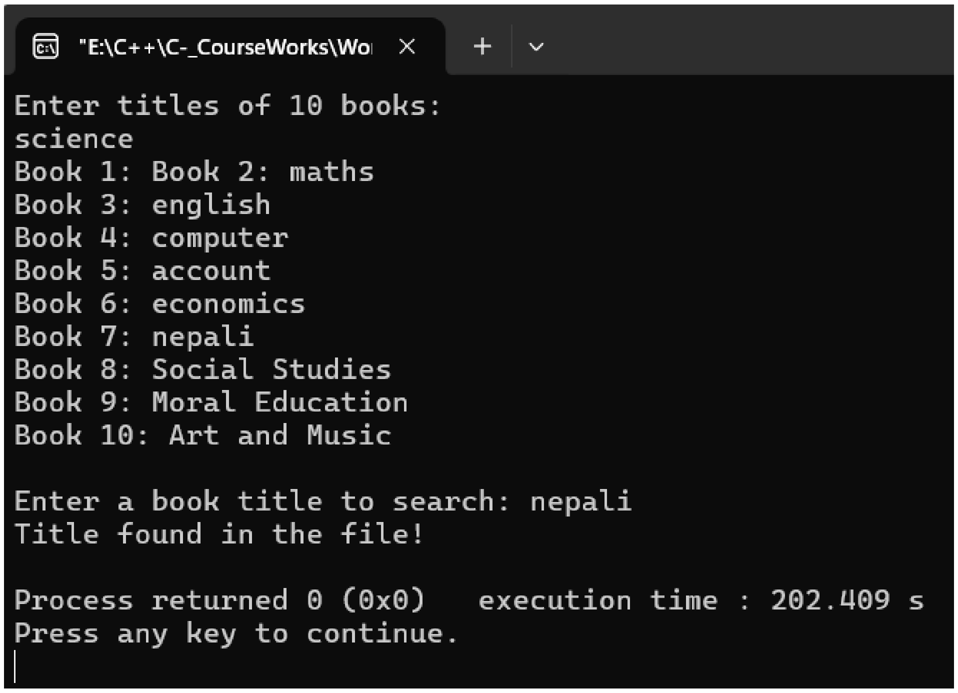
Write a program that reads the titles of 10 books (use an array of 150 characters) and writes them in a binary file selected by the user. The program should read a title and display a message to indicate if it is contained in the file or not.

Code Implementation

```
#include <iostream>

#include <fstream>  //For file handling

#include <cstring>  //For string handling

using namespace std;

int main() {

   char bookTitles[10][150];  //Array to store 10 book titles (each 150 characters)

   ofstream outFile;         //Output file stream to write to file

   //Open the binary file for writing (append mode, creates if not exists)

   outFile.open("bookTitles.dat", ios::binary | ios::app);

   if (!outFile) {

      cout << "Error opening file for writing!" << endl;

      return 1;

   }

   //Read 10 book titles from the user

   cout << "Enter titles of 10 books:" << endl;

   cin.ignore();  //Ignore newline character left in the buffer

   for (int i = 0; i < 10; i++) {

      cout << "Book " << i + 1 << ": ";

      cin.getline(bookTitles[i], 150);  //Read the book title

      outFile.write(bookTitles[i], sizeof(bookTitles[i]));  //Write to binary file
}
outFile.close();  //Close the file after writing
//Ask the user to enter a book title to search
char searchTitle[150];
cout << "\nEnter a book title to search: ";
cin.getline(searchTitle, 150);
//Ensure the file exists (create an empty file if missing)
ifstream testFile("bookTitles.dat", ios::binary);
if (!testFile) {
   ofstream createFile("bookTitles.dat", ios::binary);
   createFile.close();
}
testFile.close();
//Open the binary file for reading
ifstream inFile("bookTitles.dat", ios::binary);
if (!inFile) {
   cout << "Error opening file for reading!" << endl;
   return 1;
}
bool found = false;
char title[150];
//Read until the end of the file
while (inFile.read(title, sizeof(title))) {
   if (strcmp(title, searchTitle) == 0) {
      found = true;
      break;
   }
}
```

```
//Display result based on the search
if (found) {
cout << "Title found in the file!" << endl;
} else {
cout << "Title not found in the file." << endl;
}
inFile.close();  //Close the file after reading
return 0;
}
```

Output



"E:\C++\C-_CourseWorks\Wo

```
Enter titles of 10 books:
science
Book 1: Book 2: maths
Book 3: english
Book 4: computer
Book 5: account
Book 6: economics
Book 7: nepali
Book 8: Social Studies
Book 9: Moral Education
Book 10: Art and Music

Enter a book title to search: nepali
Title found in the file!

Process returned 0 (0x0)   execution time : 202.409 s
Press any key to continue.
```

Question 3.2

Create a program that:

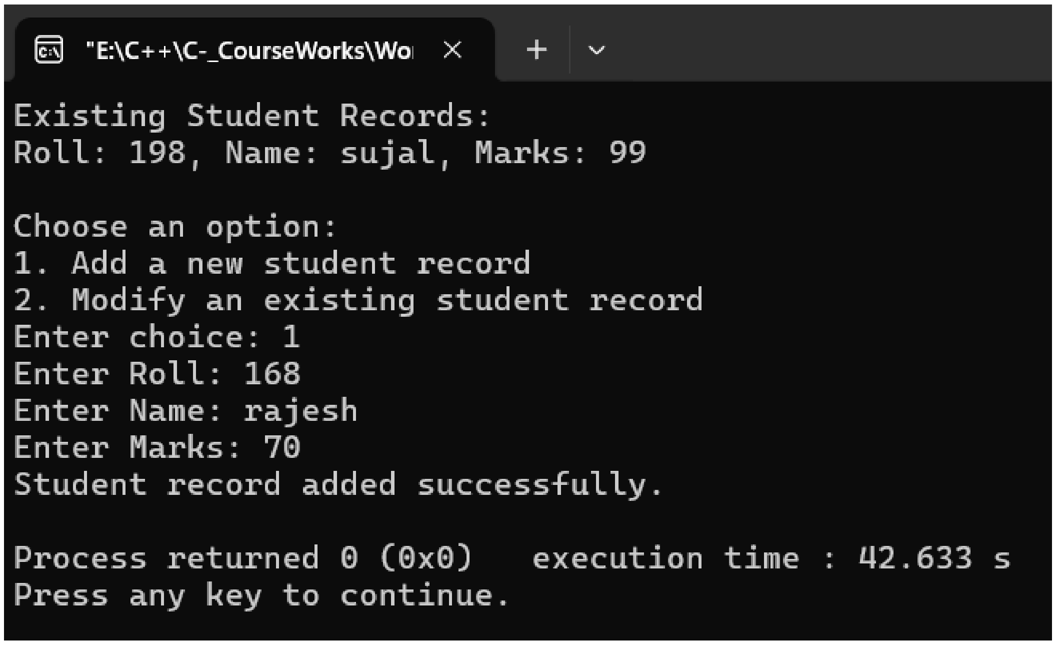
1. Reads student records (roll, name, marks) from a text file
2. Throws an exception if marks are not between 0 and 100
3. Allows adding new records with proper validation
4. Saves modified records back to file

Code Implementation

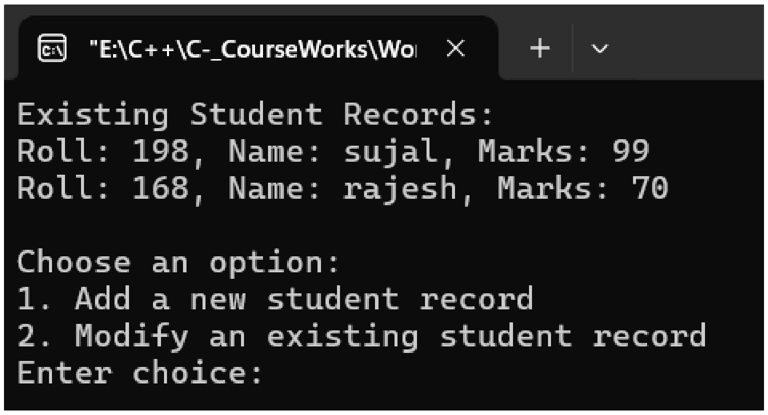
```
#include <iostream>
#include <fstream>  //For file handling
#include <stdexcept>  //For exception handling
#include <string>
#include <vector>  //For storing records in memory
using namespace std;
//Structure to store student data
struct Student {
int roll;
string name;
int marks;
};
//Function to validate marks
void validateMarks(int marks) {
if (marks < 0 || marks > 100) {
throw out_of_range("Marks must be between 0 and 100.");
}
}
//Function to read student records from a file
vector<Student> readRecords(string fileName) {
vector<Student> students;
ifstream inFile(fileName);
if (!inFile) {
cout << "File doesn't exist. It will be created when saving new records.\n";
return students;  //Return empty list
}
```

```
//Function to write student records back to the file
void saveRecords(string fileName, vector<Student> students) {
ofstream outFile(fileName);
if (!outFile) {
cout << "Error opening file for writing!\n";
return;
}
for (const auto& student : students) {
outFile << student.roll << " " << student.name << " " << student.marks << endl;
}
outFile.close();
}
int main() {
string fileName = "students.txt";
vector<Student> students = readRecords(fileName);
// Display existing records
if (!students.empty()) {
cout << "Existing Student Records:\n";
for (const auto& student : students) {
cout << "Roll: " << student.roll << ", Name: " << student.name << ", Marks: " << student.marks <<
endl;
}
} else {
cout << "No student records found.\n";
}
int choice;
cout << "\nChoose an option:\n";
cout << "1. Add a new student record\n";
cout << "2. Modify an existing student record\n";
cout << "Enter choice: ";
cin >> choice;
if (choice == 1) {
// Adding a new student
Student newStudent;
cout << "Enter Roll: ";
cin >> newStudent.roll;
cin.ignore();  //Ignore newline left by previous input
cout << "Enter Name: ";
getline(cin, newStudent.name);
cout << "Enter Marks: ";
cin >> newStudent.marks;
try {
validateMarks(newStudent.marks);
students.push_back(newStudent);
cout << "Student record added successfully.\n";
} catch (const out_of_range& e) {
cout << "Error: " << e.what() << endl;
}
} else if (choice == 2) {
//Modifying an existing student record
int rollNumber;
cout << "Enter roll number to modify: ";
cin >> rollNumber;
bool found = false;
for (auto& student : students) {
if (student.roll == rollNumber) {
found = true;
cout << "Enter new marks: ";
int newMarks;
cin >> newMarks;

try {
validateMarks(newMarks);
student.marks = newMarks;
cout << "Marks updated successfully.\n";
} catch (const out_of_range& e) {
cout << "Error: " << e.what() << endl;
}
break;
}
}
if (!found) {
cout << "Student with roll number " << rollNumber << " not found.\n";
}
} else {
cout << "Invalid choice.\n";
}
//Save the modified records back to the file
saveRecords(fileName, students);
return 0;
}
```

Output



```
Existing Student Records:
Roll: 198, Name: sujal, Marks: 99

Choose an option:
1. Add a new student record
2. Modify an existing student record
Enter choice: 1
Enter Roll: 168
Enter Name: rajesh
Enter Marks: 70
Student record added successfully.

Process returned 0 (0x0)   execution time : 42.633 s
Press any key to continue.
```



```
Existing Student Records:
Roll: 198, Name: sujal, Marks: 99
Roll: 168, Name: rajesh, Marks: 70

Choose an option:
1. Add a new student record
2. Modify an existing student record
Enter choice:
```

Conclusion

Throughout this project, I have substantially enhanced my knowledge of C++ programming and the principles of OOP. Now I was introduced to design classes that could bundle both data and behavior, implement friend functions and overloaded functions to reuse code, and manage files and exceptions. This practical application has enhanced my problem-solving abilities and provided me with a better understanding of real-world software development issues. I used my prior experience as a foundation, building on it to complete even complex programming tasks, and I am happy with how far I have come.