# 

**C++ Worksheet Task\_2**

**ASSESSMENT**

**WEIGHTAGE AND TYPE: 12.5%**

**YEAR: 2024-25**

**STUDENT NAME: SWOYAMRAJ SHRESTHA**

**STUDENT ID:** **24030185**

**Question\_1.1**

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:

Total marks

Average marks

Grade (A for ≥90%, B for ≥80%, C for ≥70%, D for ≥60%, F for <60%)

Display a message if any mark is below 0 or above 100

#include <iostream>

using namespace std;

class Student

{

string name;

int marks[3];

public:

void stdDetails()

{

cout << "Student's Name: ";

cin >> name;

cout << "Marks of three subjects: ";

for (int i = 0; i < 3; i++)

{

cin >> marks[i];

if (marks[i] < 0 || marks[i] > 100) // uses marks input range between 0 and 100

{

cout << "Invalid! Marks should be in between 0 and 100." << endl;

return;

}

}

}

int calcTotalmarks()

{

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

}

float calcAverage()

{

return calcTotalmarks() / 3.0;

}

char calcGrade()

{

float percentage = (calcTotalmarks() / 300.0) \* 100;

if (percentage >= 90) return 'A';

else if (percentage >= 80) return 'B';

else if (percentage >= 70) return 'C';

else if (percentage >= 60) return 'D';

else return 'F';

}

void displayResults()

{

int total = calcTotalmarks();

float avg = calcAverage();

char grade = calcGrade();

cout << "\nStudent's Name: " << name << endl;

cout << "Total Marks gained: " << total << endl;

cout << "Average Marks: " << avg << endl;

cout << "Grade: " << grade << endl;

}

};

int main()

{

Student student;

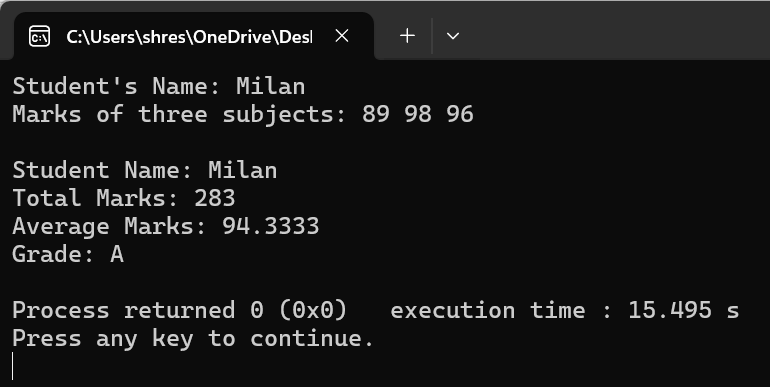
student.stdDetails();

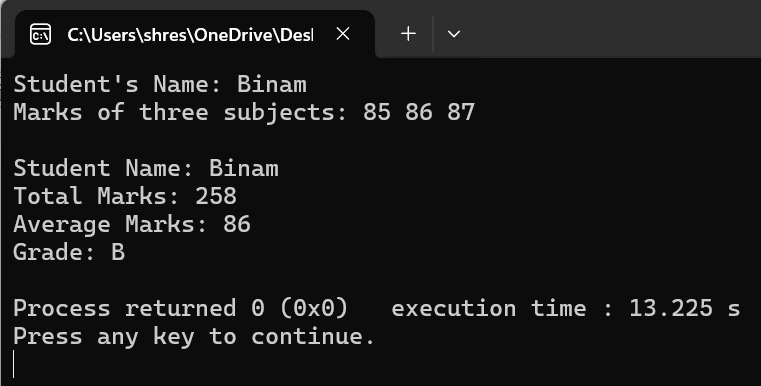
student.displayResults();

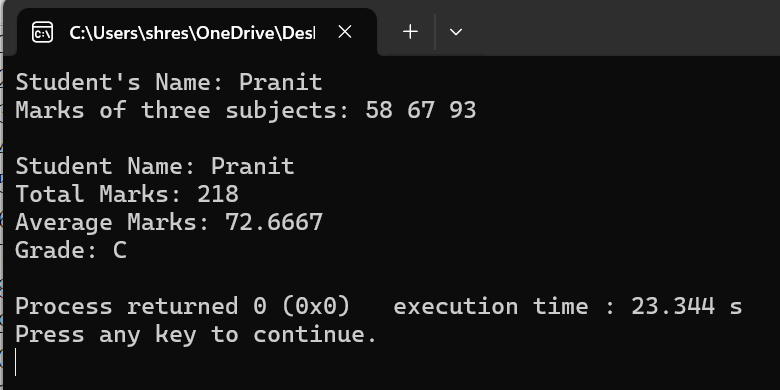
return 0;

}

**Output:**

****

****

****

**Question\_2.1**

Write a program with a class Circle having:

\*Private member: radius (float)

\*A constructor to initialize radius

\*A friend function compareTwoCircles that takes two Circle objects and prints which circle has the larger area.

#include <iostream>

using namespace std;

class Circle

{

private:

float radius;

public:

Circle(float r)

{

radius = r;

}

float calcArea() const

{

return 3.14159 \* radius \* radius;

}

void compare(const Circle& other) const //function for comparing two circles

{

float area1 = this-> calcArea();

float area2 = other.calcArea();

cout << "Area of First Circle: " << area1 << endl;

cout << "Area of Second Circle: " << area2 << endl;

if (area1 > area2)

{

cout << "First circle has Larger Area." << endl;

}

else if (area2 > area1)

{

cout << "Second Circle has Larger Area." << endl;

}

else

{

cout << "Both circle have the same area." << endl;

}

}

};

int main()

{

float r1, r2;

cout << "Radius of First Circle: ";

cin >> r1;

cout << "Radius of Second circle: ";

cin >> r2;

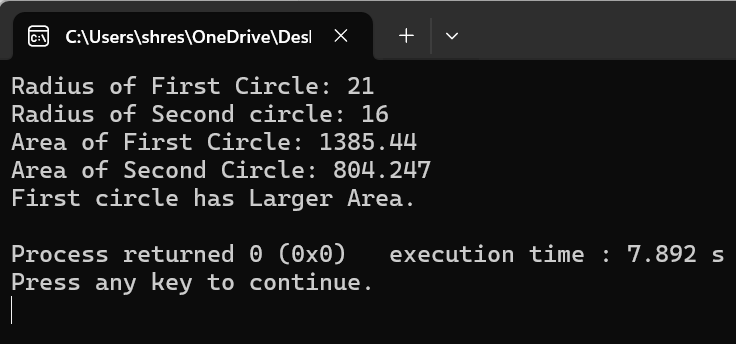
Circle circle1(r1), circle2(r2);

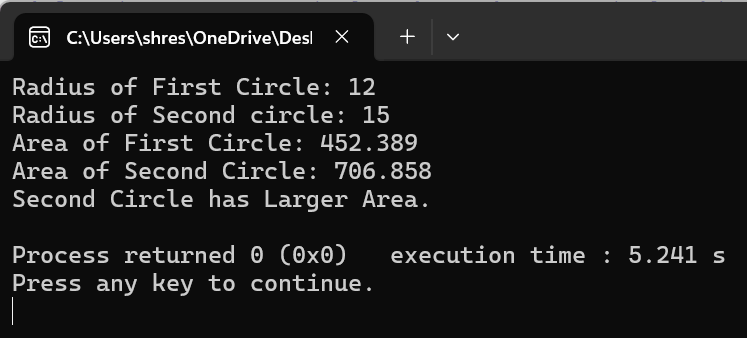
circle1.compare(circle2); //comparing two circles

return 0;

}

**Output:**

****



**Question\_2.2**

Create a program with these overloaded functions named findMax:

\*One that finds maximum between two integers

\*One that finds maximum between two floating-point numbers

\*One that finds maximum among three integers

-One that finds maximum between an integer and a float

#include <iostream>

using namespace std;

class MaxFinder

{

public:

int findMax(int x, int y)

{

return (x > y) ? x : y; //finds maximum of two integers

}

float findMax(float x, float y) //finds maximum between two floating point numbers

{

return (x > y) ? x : y;

}

int findMax(int x, int y, int z) //finds maximum among three integers

{

if (x > y && x > z)//Comparing all

{

return x;

}

else if (y > x && y > z)

{

return y;

}

else

{

return z;

}

}

float findMax(int x, float y) //finds maximum between an integer and floating-point number

{

return (x > y) ? x : y;

}

};

int main()

{

MaxFinder maxFinder; //Creating an object of MaxFinder class

int int1, int2, int3; //Declared 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: ";

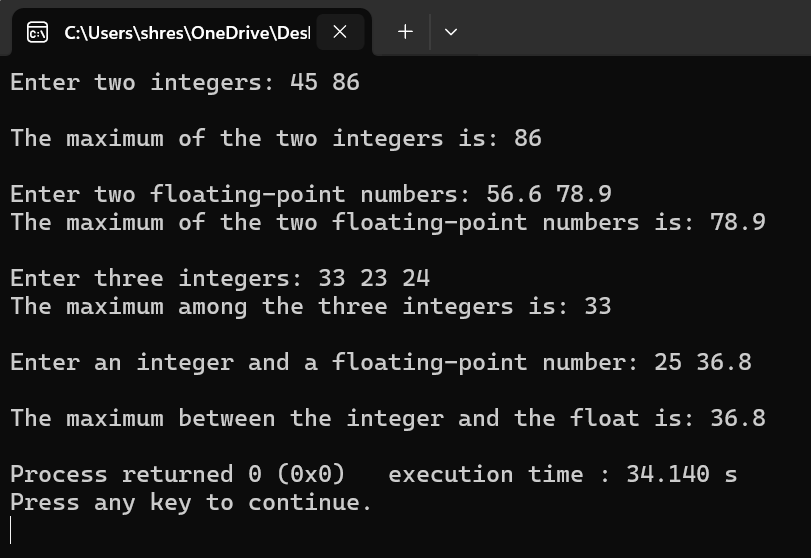
cin >> int1 >> float1;

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

return 0;

}

**Output:**



**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.

#include <iostream>

#include <fstream>

#include <cstring>

using namespace std;

int main()

{

char books[10][150]; //using array to store 10 book titles with each of 150 characters

ofstream outputFile;

outputFile.open("books.dat", ios::binary | ios::app);

if (!outputFile)

{

cout << "Unable to open file for writing!" << endl;

return 1;

}

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

cin.ignore();

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

{

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

cin.getline(books[i], 150);

outputFile.write(books[i], sizeof(books[i]));

}

outputFile.close(); //closes file

char searchTitle[150];

cout << "Enter the book title you want to search for: ";

cin.getline(searchTitle, 150);

ifstream testFile("books.dat", ios::binary);

if (!testFile)

{

ofstream createFile("books.dat", ios::binary);

createFile.close();

}

testFile.close();

ifstream inFile("books.dat", ios::binary); //binary file

if (!inFile)

{

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

return 1;

}

bool isfound = false;

char title[150];

while (inFile.read(title, sizeof(title)))

{

if (strcmp(title, searchTitle) == 0)

{

isfound = true;

break;

}

}

if (isfound)

{

cout << "The book title is available in the file." << endl;

}

else

{

cout << "Sorry, the book title is not available in the file." << endl;

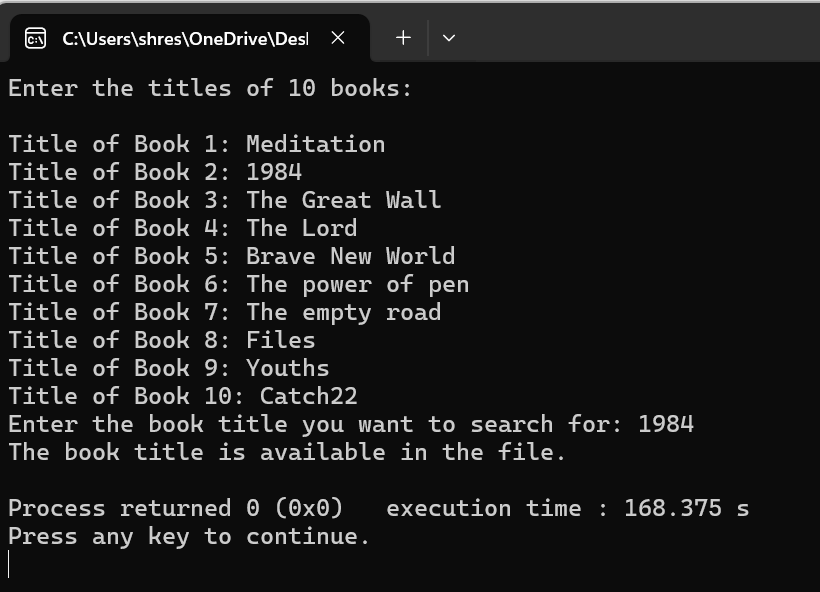
}

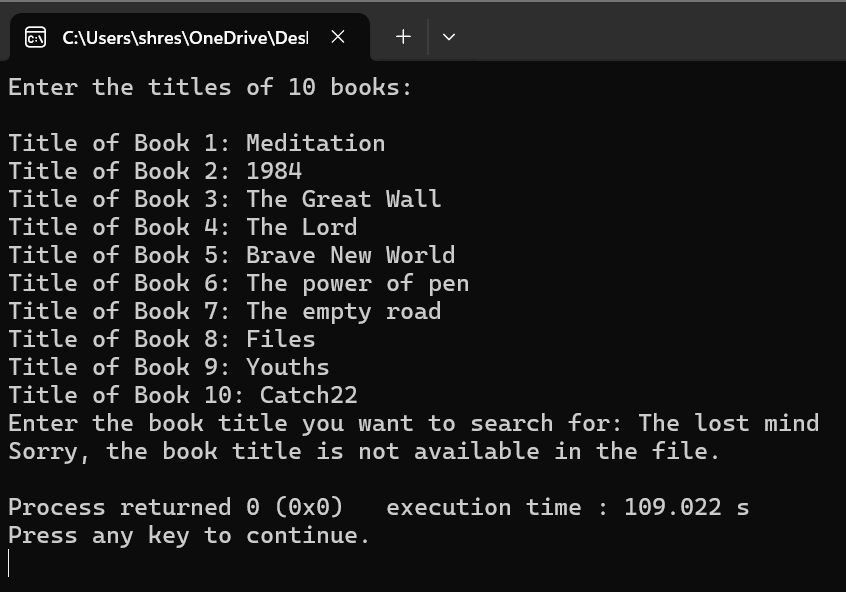
inFile.close();

return 0;

}

**Output:**





**Question\_3.2**

Create a program that:

\*Reads student records (roll, name, marks) from a text file

\*Throws an exception if marks are not between 0 and 100

\*Allows adding new records with proper validation

\*Saves modified records back to file

#include <iostream>

#include <fstream>

#include <stdexcept>

#include <string>

#include <vector>

using namespace std;

struct Student

{

int roll;

string name;

int marks;

};

void validateMarks(int marks)

{

if (marks < 0 || marks > 100)

{

throw out\_of\_range("Marks should be between 0 and 100.");

}

}

vector<Student> readRecords(string fileName)

{

vector<Student> records;

ifstream File(fileName);

if (!File)

{

cout << "The file does not exist.\n";

return records;

}

Student student;

while (File >> student.roll >> student.name >> student.marks)

{

records.push\_back(student);

}

File.close();

return records;

}

void saveStdRecords(string fileName, vector<Student> records)

{

ofstream File(fileName);

if (!File)

{

cout << "Error opening file for writing!\n";

return;

}

for (const auto& records : records)

{

File << records.roll << " " << records.name << " " << records.marks << endl;

}

File.close();

}

int main()

{

string fileName = "Student's Record.txt";

vector<Student> studentlist = readRecords(fileName);

if (!studentlist.empty())

{

cout << "Existing Student Records:\n";

for (const auto& student : studentlist)

{

cout << "Roll: " << student.roll << ", Name: " << student.name << ", Marks: " << student.marks << endl;

}

}

else

{

cout << "No records found.\n";

}

bool running = true;

while (running)

{

int userchoice;

cout << "\nChoose an option:\n";

cout << "1. Add new student record\n";

cout << "2. Modify existing student record\n";

cout << "3. Save and Exit\n";

cout << "Enter choice: ";

cin >> userchoice;

if (userchoice == 1)

{

Student newStudent; // Option to add a new student

cout << "Enter Roll: ";

cin >> newStudent.roll;

cin.ignore(); // To clear the buffer after taking integer input

cout << "Enter Name: ";

getline(cin, newStudent.name);

cout << "Enter Marks: ";

cin >> newStudent.marks;

try

{

validateMarks(newStudent.marks);

studentlist.push\_back(newStudent);

cout << "New student record added successfully.\n";

}

catch (const out\_of\_range& e)

{

cout << "Error: " << e.what() << endl;

}

}

else if (userchoice == 2)

{

int rollNoToModify;

cout << "Enter Roll No of student to modify: ";

cin >> rollNoToModify;

bool recfound = false;

for (auto& student : studentlist)

{

if (student.roll == rollNoToModify)

{

recfound = 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 (!recfound)

{

cout << "Student with Roll No " << rollNoToModify << " not found.\n";

}

}

else if (userchoice == 3)

{

saveStdRecords(fileName, studentlist);

cout << "Records saved successfully! Exiting program.\n";

running = false;

}

else

{

cout << "Invalid choice. Please try again.\n";

}

}

return 0;

}

**Output:**

