SUBMITTED BY: Sadaf Akhtar Ansari

STUDENT ID : 24030177

worksheet 2

A logo with blue and red text

AI-generated content may be incorrect.A red sign with white text

AI-generated content may be incorrect.

**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:**

#include <iostream>

using namespace std;

class Student { // Class to handle student info and grades

string name; // Store student's name

int marks[3]; // Store marks for 3 subjects

public:

void getDetails() { // Ask user for name and marks

cout << "Enter student name: ";

cin >> name;

cout << "Enter marks for 3 subjects: ";

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

cin >> marks[i];

if (marks[i] < 0 || marks[i] > 100) { // Check if marks are valid

cout << "Marks must be between 0 and 100!\n";

return;

}

}

}

int calculateTotal() { // Add up all the marks

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

}

float calculateAverage() { // Find the average

return calculateTotal() / 3.0;

}

char calculateGrade() { // Give grade based on average

float avg = calculateAverage();

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

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

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

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

else return 'F';

}

void displayResults() { // Show all details

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

cout << "Total Marks: " << calculateTotal() << endl;

cout << "Average Marks: " << calculateAverage() << endl;

cout << "Grade: " << calculateGrade() << endl;

}

};

int main() { // Program starts here

Student student; // Create a student

student.getDetails(); // Input details

student.displayResults(); // Show results

return 0;

}

OUTPUT:

A screenshot of a computer

AI-generated content may be incorrect.

**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:**

#include <iostream>

using namespace std;

class Circle {

private:

float radius; // To store the radius of the circle

public:

Circle(float r) { // Constructor to set the radius

radius = r;

}

float getArea() const { // Function to find area of the circle

return 3.14159 \* radius \* radius;

}

void compareWith(const Circle& other) const { // Function to compare two circles

float area1 = getArea();

float area2 = other.getArea();

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

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

if (area1 > area2) {

cout << "First circle is larger.\n";

} else if (area2 > area1) {

cout << "Second circle is larger.\n";

} else {

cout << "Both circles are equal in area.\n";

}

}

};

int main() { // Main program starts here

float r1, r2;

cout << "Enter radius of first circle: ";

cin >> r1;

cout << "Enter radius of second circle: ";

cin >> r2;

Circle circle1(r1); // Create first circle

Circle circle2(r2); // Create second circle

circle1.compareWith(circle2); // Compare the two circles

return 0;

}

OUTPUT:

A screenshot of a computer

AI-generated content may be incorrect.

**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:**

#include <iostream>

using namespace std;

class Circle {

private:

float radius; // Stores the radius of the circle

public:

Circle(float r) { // Constructor to initialize radius

radius = r;

}

float getArea() const { // Function to calculate area of the circle

return 3.14159 \* radius \* radius;

}

friend void compareTwoCircles(const Circle& c1, const Circle& c2); // Friend function declaration

};

void compareTwoCircles(const Circle& c1, const Circle& c2) { // Friend function definition

float area1 = c1.getArea();

float area2 = c2.getArea();

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

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

if (area1 > area2) {

cout << "First circle is larger." << endl;

} else if (area2 > area1) {

cout << "Second circle is larger." << endl;

} else {

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

}

}

int main() { // Main program starts here

float r1, r2;

cout << "Enter radius of first circle: ";

cin >> r1;

cout << "Enter radius of second circle: ";

cin >> r2;

Circle circle1(r1); // Create first Circle object

Circle circle2(r2); // Create second Circle object

compareTwoCircles(circle1, circle2); // Call the friend function to compare

return 0;

}

**OUTPUT:**

**A screenshot of a computer

AI-generated content may be incorrect.**

**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:**

#include <iostream>

#include <fstream> // For file operations

#include <cstring> // For using strcmp

using namespace std;

int main() {

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

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

// Open a binary file in append mode (creates file if it doesn't exist)

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

if (!outFile) {

cout << "Error: Could not open file for writing!" << endl;

return 1; // Exit if file can't be opened

}

// Take 10 book titles from the user

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

cin.ignore(); // Clear the input buffer before taking input

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

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

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

outFile.write(bookTitles[i], sizeof(bookTitles[i])); // Save the title to the file

}

outFile.close(); // Close the file after writing all titles

// Now ask the user for a title to search

char searchTitle[150];

cout << "\nEnter a book title to search: ";

cin.getline(searchTitle, 150); // Take the title to search

ifstream inFile("bookTitles.dat", ios::binary); // Open the file for reading

if (!inFile) {

cout << "Error: Could not open file for reading!" << endl;

return 1; // Exit if file can't be opened

}

bool found = false; // To keep track if title is found

char title[150]; // Temporary array to hold titles from file

// Read titles one by one and compare

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

if (strcmp(title, searchTitle) == 0) { // Compare the searched title with file title

found = true;

break; // No need to search further

}

}

inFile.close(); // Always close the file after reading

// Show result

if (found) {

cout << "Title found in the file!" << endl;

} else {

cout << "Title not found in the file." << endl;

}

return 0; // End of program

}

**OUTPUT:**

A computer screen shot of a black screen

AI-generated content may be incorrect.

**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:**

#include <iostream>

#include <fstream> // for reading and writing files

#include <stdexcept> // for throwing exceptions

#include <string>

#include <vector> // to store multiple student records

using namespace std;

// Structure to store student information

struct Student {

int roll;

string name;

int marks;

};

// Function to check if marks are valid (0 to 100)

void validateMarks(int marks) {

if (marks < 0 || marks > 100) {

throw out\_of\_range("Marks must be between 0 and 100."); // Throw error if marks are invalid

}

}

// Function to read students from a file

vector<Student> readRecords(string fileName) {

vector<Student> students;

ifstream inFile(fileName); // Open file for reading

if (!inFile) {

cout << "File not found. A new file will be created later.\n";

return students; // Return empty list if file does not exist

}

Student student;

// Read data until end of file

while (inFile >> student.roll >> student.name >> student.marks) {

students.push\_back(student); // Add student to the list

}

inFile.close(); // Close the file

return students;

}

// Function to save all student records to a file

void saveRecords(string fileName, vector<Student> students) {

ofstream outFile(fileName); // Open file for writing (this will overwrite old content)

if (!outFile) {

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

return;

}

for (const auto& student : students) {

outFile << student.roll << " " << student.name << " " << student.marks << endl; // Write each student's data

}

outFile.close(); // Close the file

}

int main() {

string fileName = "students.txt";

vector<Student> students = readRecords(fileName); // Read old records

// Show old student records

if (!students.empty()) {

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

for (const auto& student : students) {

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

}

} else {

cout << "No student records available.\n";

}

int choice;

cout << "\nWhat do you want to do?\n";

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

cout << "2. Update marks of a student\n";

cout << "Enter your choice: ";

cin >> choice;

if (choice == 1) {

// Add a new student

Student newStudent;

cout << "Enter Roll Number: ";

cin >> newStudent.roll;

cin.ignore(); // Clear the buffer

cout << "Enter Name: ";

getline(cin, newStudent.name);

cout << "Enter Marks: ";

cin >> newStudent.marks;

try {

validateMarks(newStudent.marks); // Check if marks are valid

students.push\_back(newStudent); // Add student to the list

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

} catch (const out\_of\_range& e) {

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

}

} else if (choice == 2) {

// Modify marks of an existing student

int rollNumber;

cout << "Enter Roll Number to update marks: ";

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); // Check if new marks are valid

student.marks = newMarks; // Update marks

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! Please select 1 or 2.\n";

}

// Save all records back into the file

saveRecords(fileName, students);

return 0;

}

**OUTPUT:**

A screenshot of a computer program

AI-generated content may be incorrect.