**TOUSEEF AKHTAR(460700)**

**LAB 10**

**TASK 1:**

#include <iostream>

#include <vector>

using namespace std;

int main() {

vector<int> customVector; // Changed variable name to 'customVector'

for (int index = 1; index <= 10; ++index) { // Changed loop variable 'i' to 'index'

customVector.push\_back(index);

}

cout << "Original Vector: ";

for (vector<int>::iterator iter = customVector.begin(); iter != customVector.end(); ++iter) {

cout << \*iter << " ";

}

cout << endl;

customVector.push\_back(11);

if (!customVector.empty()) {

vector<int>::iterator removeIter = customVector.begin() + 1;

customVector.erase(removeIter);

}

cout << "Modified Vector: ";

for (int element : customVector) {

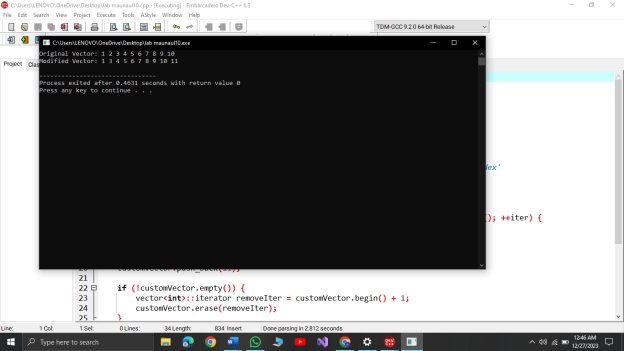
cout << element << " ";

}

cout << endl;

return 0;

}



**TASK2:**

#include <iostream>

#include <vector>

#include <algorithm>

#include <map>

using namespace std;

double calculateMean(const vector<int>& customGrades) {

if (customGrades.empty()) {

return 0.0;

}

int sum = 0;

for (int grade : customGrades) {

sum += grade;

}

return static\_cast<double>(sum) / customGrades.size();

}

double calculateMedian(vector<int>& customGrades) {

if (customGrades.empty()) {

return 0.0;

}

sort(customGrades.begin(), customGrades.end());

size\_t size = customGrades.size();

if (size % 2 == 0) {

return (customGrades[size / 2 - 1] + customGrades[size / 2]) / 2.0;

} else {

return customGrades[size / 2];

}

}

vector<int> calculateMode(const vector<int>& customGrades) {

map<int, int> countMap;

for (int grade : customGrades) {

countMap[grade]++;

}

int maxCount = 0;

for (const auto& pair : countMap) {

maxCount = max(maxCount, pair.second);

}

vector<int> modeGrades;

for (const auto& pair : countMap) {

if (pair.second == maxCount) {

modeGrades.push\_back(pair.first);

}

}

return modeGrades;

}

int main() {

int numPairs;

cout << "Enter the number of name/grade pairs: ";

cin >> numPairs;

vector<string> customNames;

vector<int> customGrades;

// Input names and grades

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

string customName;

int customGrade;

cout << "Enter name #" << i + 1 << ": ";

cin >> customName;

cout << "Enter grade #" << i + 1 << ": ";

cin >> customGrade;

customNames.push\_back(customName);

customGrades.push\_back(customGrade);

}

cout << "Mean of the grades: " << calculateMean(customGrades) << endl;

cout << "Median of the grades: " << calculateMedian(customGrades) << endl;

vector<int> modeCustomGrades = calculateMode(customGrades);

cout << "Mode of the grades: ";

for (int grade : modeCustomGrades) {

cout << grade << " ";

}

cout << endl;

cout << "Names of students with the mode grade: ";

for (size\_t i = 0; i < customGrades.size(); ++i) {

if (find(modeCustomGrades.begin(), modeCustomGrades.end(), customGrades[i]) != modeCustomGrades.end()) {

cout << customNames[i] << " ";

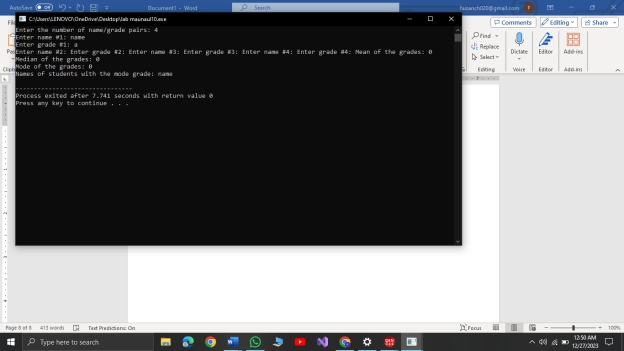
}

}

cout << endl;

return 0;

}



**Task3:**

#include <iostream>

#include <cmath>

using namespace std;

class CustomTriangle {

private:

double customSide1, customSide2, customSide3;

public:

CustomTriangle(double s1, double s2, double s3) : customSide1(s1), customSide2(s2), customSide3(s3) {}

double calculateCustomPerimeter() const {

return customSide1 + customSide2 + customSide3;

}

double calculateCustomArea() const {

double s = calculateCustomPerimeter() / 2.0;

return sqrt(s \* (s - customSide1) \* (s - customSide2) \* (s - customSide3));

}

void printCustomInfo() const {

cout << "Custom Triangle with sides: " << customSide1 << " m, " << customSide2 << " m, " << customSide3 << " m" << endl;

cout << "Perimeter: " << calculateCustomPerimeter() << " m" << endl;

cout << "Area: " << calculateCustomArea() << " square meters" << endl;

}

};

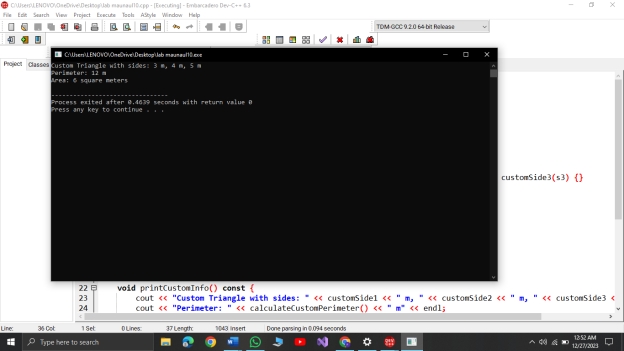
int main() {

CustomTriangle myCustomTriangle(3.0, 4.0, 5.0);

myCustomTriangle.printCustomInfo();

return 0;

}



**Task4:**

#include <iostream>

#include <string>

using namespace std;

struct CustomEmployee {

string customName;

double customSalary;

int customHoursOfWorkPerDay;

};

void increaseCustomSalary(CustomEmployee& customEmployee) {

if (customEmployee.customHoursOfWorkPerDay >= 12) {

customEmployee.customSalary += 150;

} else if (customEmployee.customHoursOfWorkPerDay >= 10) {

customEmployee.customSalary += 100;

} else if (customEmployee.customHoursOfWorkPerDay >= 8) {

customEmployee.customSalary += 50;

}

}

int main() {

CustomEmployee customEmployees[10];

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

cout << "Enter name for customEmployee #" << i + 1 << ": ";

cin >> customEmployees[i].customName;

cout << "Enter salary for customEmployee #" << i + 1 << ": ";

cin >> customEmployees[i].customSalary;

cout << "Enter hours of work per day for customEmployee #" << i + 1 << ": ";

cin >> customEmployees[i].customHoursOfWorkPerDay;

increaseCustomSalary(customEmployees[i]);

}

cout << "\nFinal Salaries:\n";

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

cout << "customEmployee #" << i + 1 << ": " << customEmployees[i].customName << " - $" << customEmployees[i].customSalary << endl;

}

return 0;

}

