**Objective:** This lab aims to guide you through the process of creating a simple calculator application in Visual Studio using C++, and subsequently writing comprehensive unit tests using Google Test.

**Task:** Your task is to modify Calculator application code and

* 1. Add a squareRoot (double a) function to the Calculator class. Ensure that the function handles negative inputs appropriately (e.g., return 0 or throw an exception).

Ans:

test.cpp:

#include "pch.h"

#include"C:\Users\23fa-048st\Desktop\ConsoleApplication1\ConsoleApplication1\cal.cpp"

TEST(cal, addition) {

EXPECT\_EQ(4,add(2,2));

EXPECT\_TRUE(true);}

TEST(cal, subtraction) {

EXPECT\_EQ(4, sub(2, 2));

EXPECT\_TRUE(true);}

TEST(cal, multiply) {

EXPECT\_EQ(4, mul(2, 2));

EXPECT\_TRUE(true);}

TEST(cal, division) {

EXPECT\_EQ(4, mul(2, 2));

EXPECT\_TRUE(true);}

TEST(cal, squareRoot) {

EXPECT\_EQ(2, squareRoot(4));

EXPECT\_TRUE(true);}

Cal.cpp

#include<cmath>

double add(double a, double b) {

return a + b;}

double sub(double a, double b) {

return a - b;}

double mul(double a, double b) {

return a \* b;}

double dev(double a, double b) {

if (b == 0) {

return 0;}

else {

return(a / b);}}

double squareRoot(double a) {

if (a < 0) {

return 0;}

return sqrt(4);}

Cal.h

#pragma once

double add(double a, double b);

double sub(double a, double b);

double mul(double a, double b);

double dev(double a, double b);

double squareRoot(double a);

ConsoleApplication1.cpp

#include <iostream>

#include"cal.h"

using namespace std;

int main(){

cout << "Hello World!\n";

cout << "output add:" << add(23, 43)<<endl;

cout << "output sub:" << sub(23, 43) << endl;

cout << "output mul:" << mul(23, 43) << endl;

cout << "output div:" << dev(23, 0) << endl;

cout << "output div:" << squareRoot(4) << endl;}

OUTPUT:



* 1. Add a factorial (int n) function to the Calculator class. Handle negative inputs and large inputs appropriately (e.g., return 1 for negative inputs or throw an exception for large values that would cause an integer overflow).

Ans:

test.cpp:

#include "pch.h"

#include "C:\Users\23fa-048-st\Desktop\ConsoleApplication1\ConsoleApplication1\cal.cpp"

TEST(cal, addition) {

EXPECT\_EQ(4,add(2,2));

EXPECT\_TRUE(true);}

TEST(cal, subtraction) {

EXPECT\_EQ(4, sub(2, 2));

EXPECT\_TRUE(true);}

TEST(cal, multiply) {

EXPECT\_EQ(4, mul(2, 2));

EXPECT\_TRUE(true);

}

TEST(cal, division) {

EXPECT\_EQ(4, mul(2, 2));

EXPECT\_TRUE(true);}

TEST(cal, squareRoot) {

EXPECT\_EQ(2, squareRoot(4));

EXPECT\_TRUE(true);}

TEST(cal, factorial) {

EXPECT\_EQ(24, factorial(4));}

Cal.cpp

#include<cmath>

double add(double a, double b) {

return a + b;}

double sub(double a, double b) {

return a - b;}

double mul(double a, double b) {

return a \* b;}

double dev(double a, double b) {

if (b == 0) {

return 0;}

else {

return(a / b);}}

double squareRoot(double a) {

if (a < 0) {

return 0;}

return sqrt(4);}

double factorial(double a){

if (a < 0) {

return 0; }

if (a == 0 || a == 1) {

return 1;}

return a \* factorial(a - 1);}

Cal.h

#pragma once

double add(double a, double b);

double sub(double a, double b);

double mul(double a, double b);

double dev(double a, double b);

double squareRoot(double a);

double factorial(double a);

ConsoleApplication1.cpp

#include <iostream>

#include"cal.h"

using namespace std;

int main(){

cout << "Hello World!\n";

cout << "output add:" << add(23, 43)<<endl;

cout << "output sub:" << sub(23, 43) << endl;

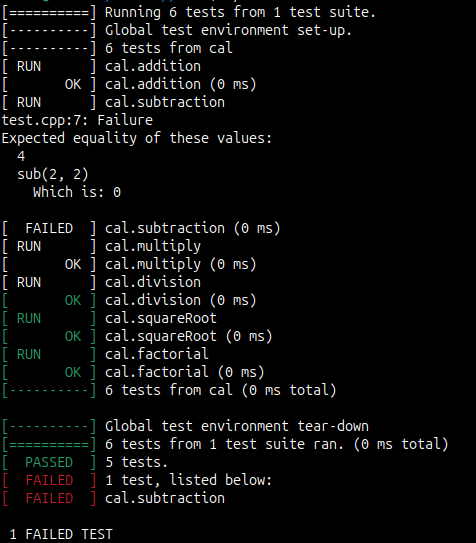
cout << "output mul:" << mul(23, 43) << endl;

cout << "output div:" << dev(23, 0) << endl;

cout << "output square-root:" << squareRoot(4) << endl;

cout << "output factorial: " << factorial(4) << endl;}

Output:



* 1. Create new test cases for the squareRoot and factorial functions. Include test cases for: Positive inputs, Negative inputs, Zero inputs, Boundary conditions (e.g., very large numbers for factorial).

Ans:

test.cpp:

#include "pch.h"

#include "C:\Users\23fa-048-st\Desktop\ConsoleApplication1\ConsoleApplication1\cal.cpp"

TEST(cal, squareRoot) {

EXPECT\_EQ(2, squareRoot(4));}

TEST(cal\_2, squareRoot\_2) {

EXPECT\_EQ(0, squareRoot(4));}

TEST(cal\_3, squareRoot\_3) {

EXPECT\_EQ(-2, squareRoot(4));}

TEST(cal\_4, squareRoot\_4) {

EXPECT\_EQ(5, squareRoot(4));}

TEST(cal, factorial) {

EXPECT\_EQ(24, factorial(4));}

TEST(cal\_2, factorial\_2) {

EXPECT\_EQ(0, factorial(4));}

TEST(cal\_3, factorial\_3) {

EXPECT\_EQ(-24, factorial(4));}

TEST(cal\_4, factorial\_4) {

EXPECT\_EQ(25, factorial(4));}

Cal.cpp

#include<cmath>

double add(double a, double b) {

return a + b;}

double sub(double a, double b) {

return a - b;}

double mul(double a, double b) {

return a \* b;}

double dev(double a, double b) {

if (b == 0) {return 0; }

else {return(a / b);}}

double squareRoot(double a) {

if (a < 0) { return 0;}

return sqrt(4);}

double factorial(double a){

if (a < 0) { return 0; }

if (a == 0 || a == 1) { return 1;}

return a \* factorial(a - 1);}

Cal.h

#pragma once

double add(double a, double b);

double sub(double a, double b);

double mul(double a, double b);

double dev(double a, double b);

double squareRoot(double a);

double factorial(double a);

ConsoleApplication1.cpp

#include <iostream>

#include"cal.h"

using namespace std;

int main(){

cout << "Hello World!\n";

cout << "output add:" << add(23, 43)<<endl;

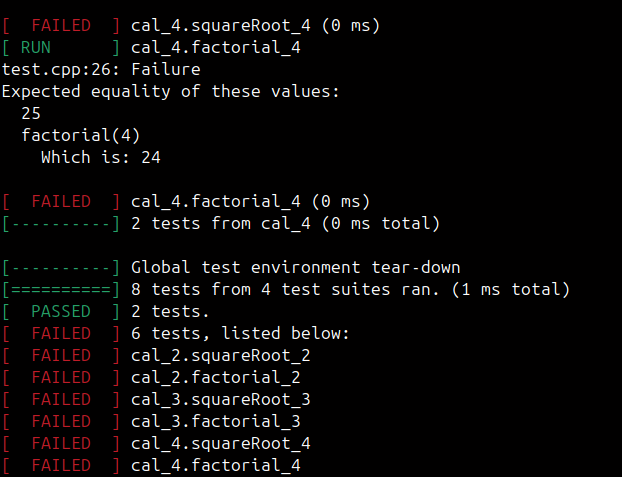
cout << "output sub:" << sub(23, 43) << endl;

cout << "output mul:" << mul(23, 43) << endl;

cout << "output div:" << dev(23, 0) << endl;

cout << "output square-root:" << squareRoot(4) << endl;

cout << "output factorial: " << factorial(4) << endl;}

Output: