****

**Lab#14 Project Submission**

**Course Code:**

**BSE-1101**

**Course Title:**

**Programming Fundamentals**

**Submitted By:**

**Uqba Gulzar**

**Section:**

**BSE - 1B**

**Registration Number:**

**2023 - BSE - 067**

**Submitted To:**

**Engr. Rehan Ahmed Siddiqui**

**Submission Date:**

**17 Jan, 2024**

**Problem Statement:**

Develop a C++ calculator application that performs basic arithmetic operations, including addition, subtraction, multiplication, and division, with additional support for scientific functions such as square root, power, sine, cosine, tangent, logarithm, secant, cosecant, and cotangent.

**Code:**

#include <iostream>

#include <cmath>

using namespace std;

// Function to perform addition

double add(double a, double b) {

return a + b;

}

// Function to perform subtraction

double subtract(double a, double b) {

return a - b;

}

// Function to perform multiplication

double multiply(double a, double b) {

return a \* b;

}

// Function to perform division

double divide(double a, double b) {

if (b != 0) {

return a / b;

}

else {

cout << "Error: Division by zero!" << endl;

return 0;

}

}

// Function to perform square root

double squareRoot(double a) {

if (a >= 0) {

return sqrt(a);

}

else {

cout << "Error: Cannot find square root of a negative number!" << endl;

return 0;

}

}

// Function to perform power operation

double power(double base, double exponent) {

double result = 1;

if (exponent >= 0) {

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

result \*= base;

}

}

else {

// Handling negative exponent by taking reciprocal

for (int i = 0; i < -exponent; ++i) {

result /= base;

}

}

return result;

}

// Function to perform sine

double sine(double angle) {

return sin(angle);

}

// Function to perform cosine

double cosine(double angle) {

return cos(angle);

}

// Function to perform tangent

double tangent(double angle) {

return tan(angle);

}

// Function to perform logarithm

double logarithm(double base, double x) {

return log(x) / log(base);

}

// Function to perform secant

double secant(double angle) {

return 1 / cosine(angle);

}

// Function to perform cosecant

double cosecant(double angle) {

return 1 / sine(angle);

}

// Function to perform cotangent

double cotangent(double angle) {

return 1 / tangent(angle);

}

int main() {

int operation;

double operand1, operand2, result;

cout << "Select operation:\n"

<< "1. Addition\n"

<< "2. Subtraction\n"

<< "3. Multiplication\n"

<< "4. Division\n"

<< "5. Square Root\n"

<< "6. Power\n"

<< "7. Sine\n"

<< "8. Cosine\n"

<< "9. Tangent\n"

<< "10. Logarithm\n"

<< "11. Secant\n"

<< "12. Cosecant\n"

<< "13. Cotangent\n";

cout << "Enter the operation number: ";

cin >> operation;

if (operation == 5 || operation == 6 || operation == 7 || operation == 8 || operation == 9 || operation == 10 || operation == 11 || operation == 12 || operation == 13) {

cout << "Enter operand: ";

cin >> operand1;

}

else {

cout << "Enter first operand: ";

cin >> operand1;

cout << "Enter second operand: ";

cin >> operand2;

}

switch (operation) {

case 1:

result = add(operand1, operand2);

break;

case 2:

result = subtract(operand1, operand2);

break;

case 3:

result = multiply(operand1, operand2);

break;

case 4:

result = divide(operand1, operand2);

break;

case 5:

result = squareRoot(operand1);

break;

case 6:

cout << "Enter exponent: ";

cin >> operand2;

result = power(operand1, operand2);

break;

case 7:

result = sine(operand1);

break;

case 8:

result = cosine(operand1);

break;

case 9:

result = tangent(operand1);

break;

case 10:

cout << "Enter base: ";

cin >> operand2;

result = logarithm(operand2, operand1);

break;

case 11:

result = secant(operand1);

break;

case 12:

result = cosecant(operand1);

break;

case 13:

result = cotangent(operand1);

break;

default:

cout << "Error: Invalid operation!" << endl;

return 1;

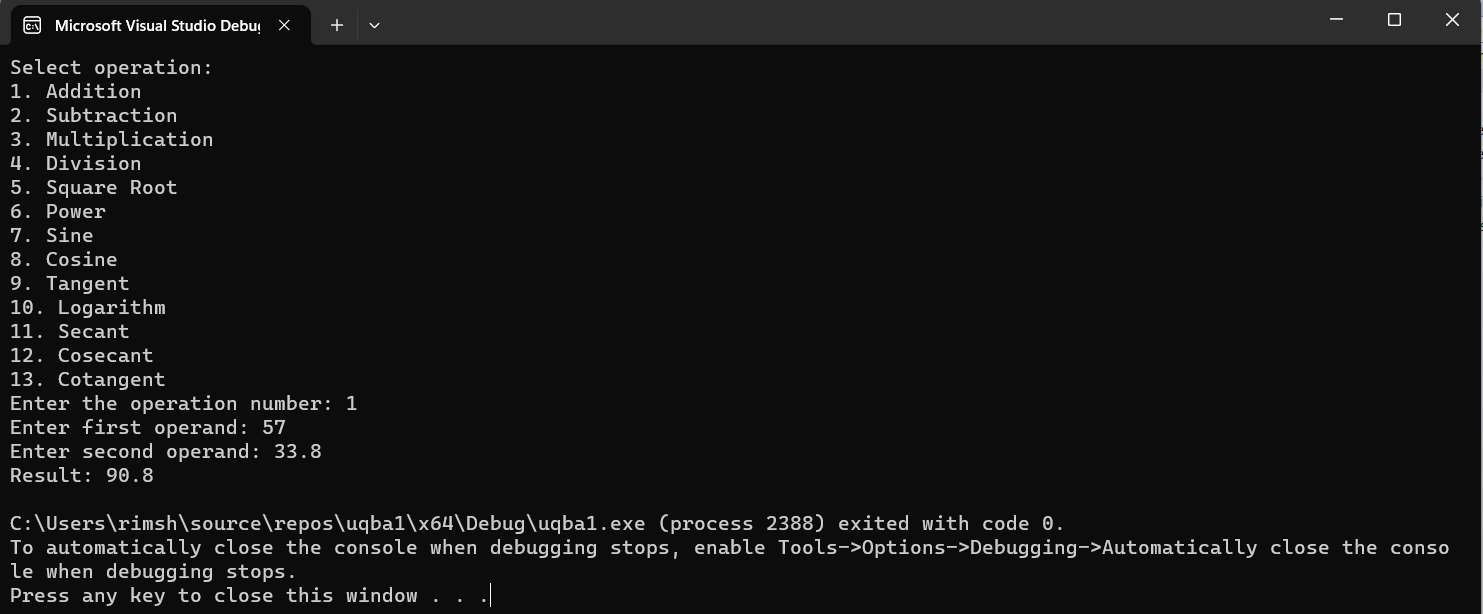
}

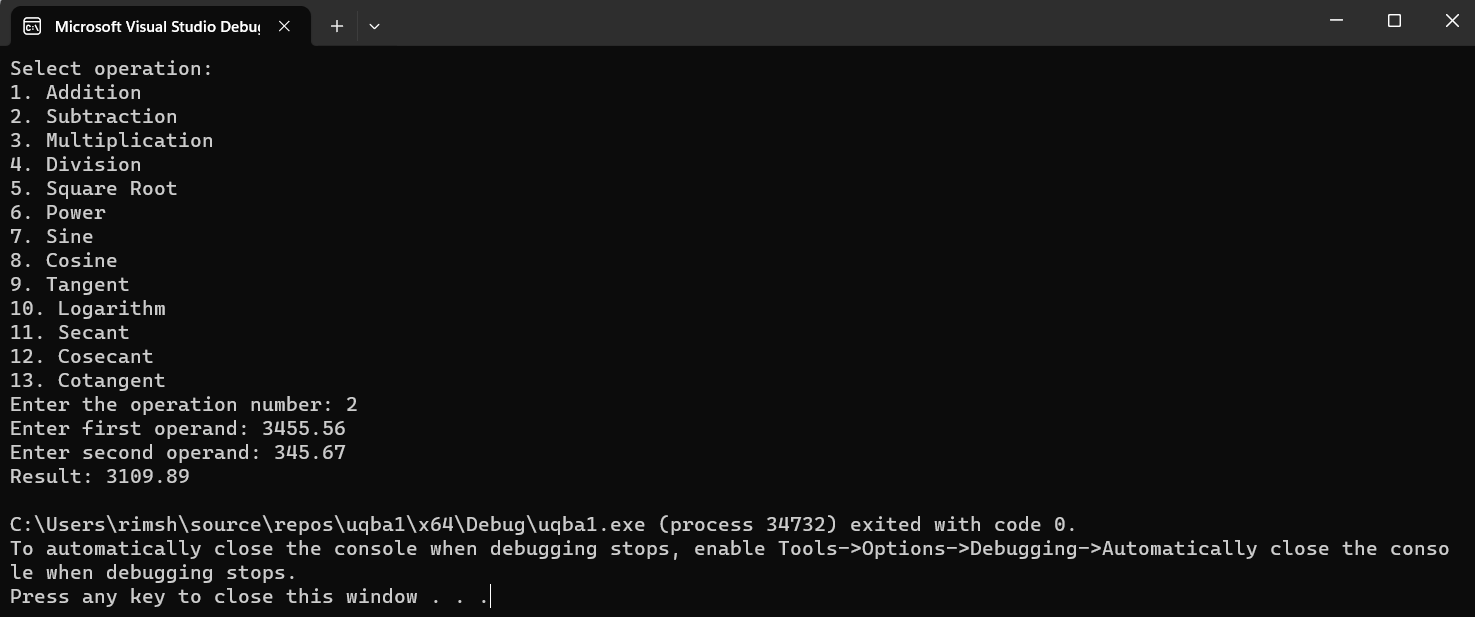
cout << "Result: " << result << endl;

return 0;

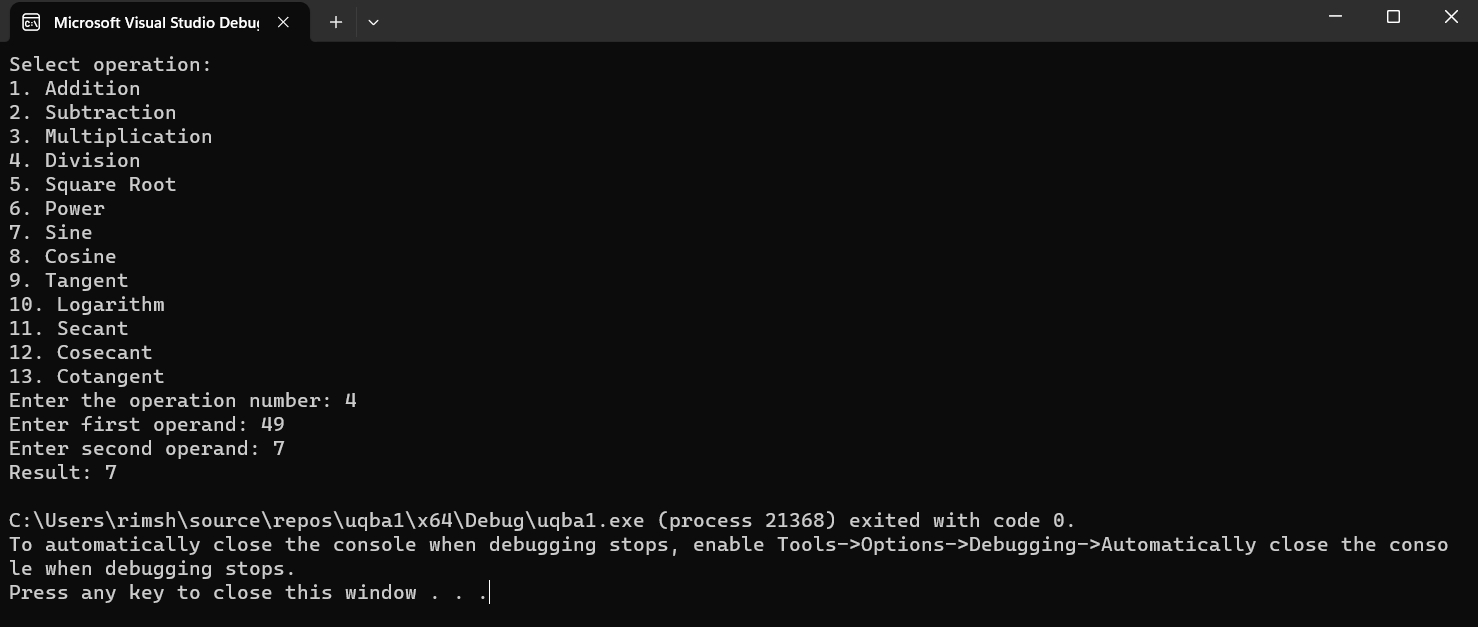
}

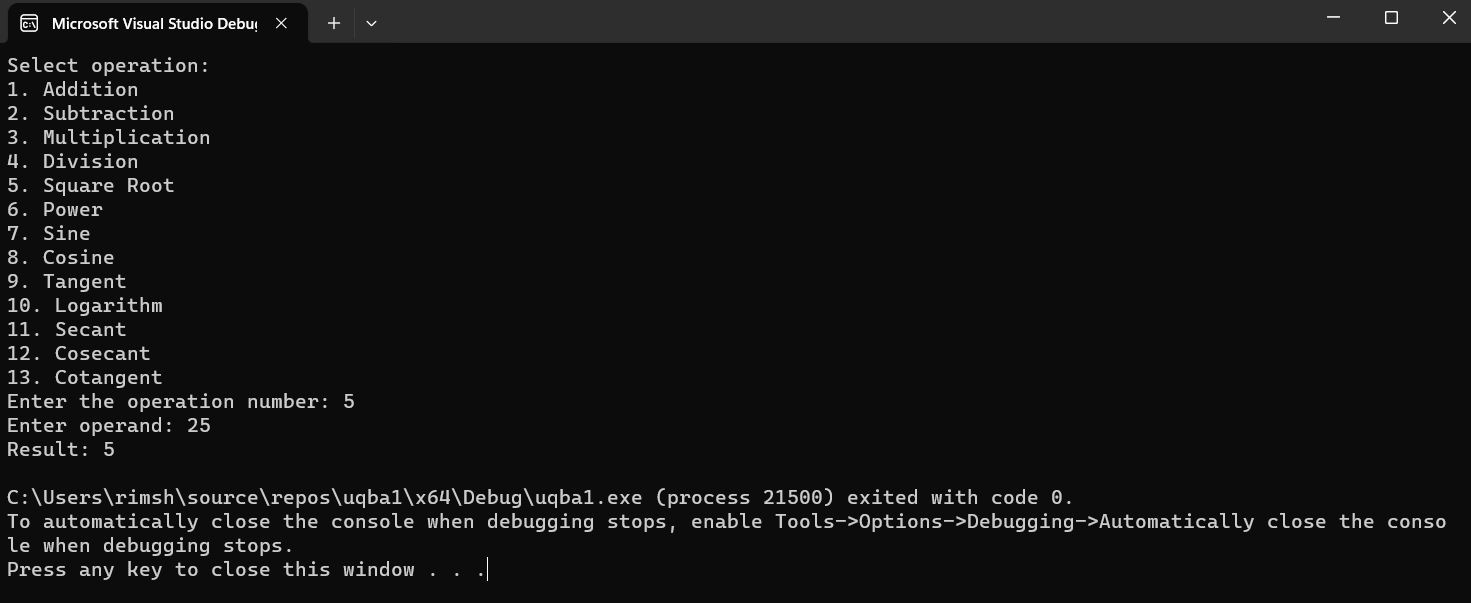
**Output:**

****

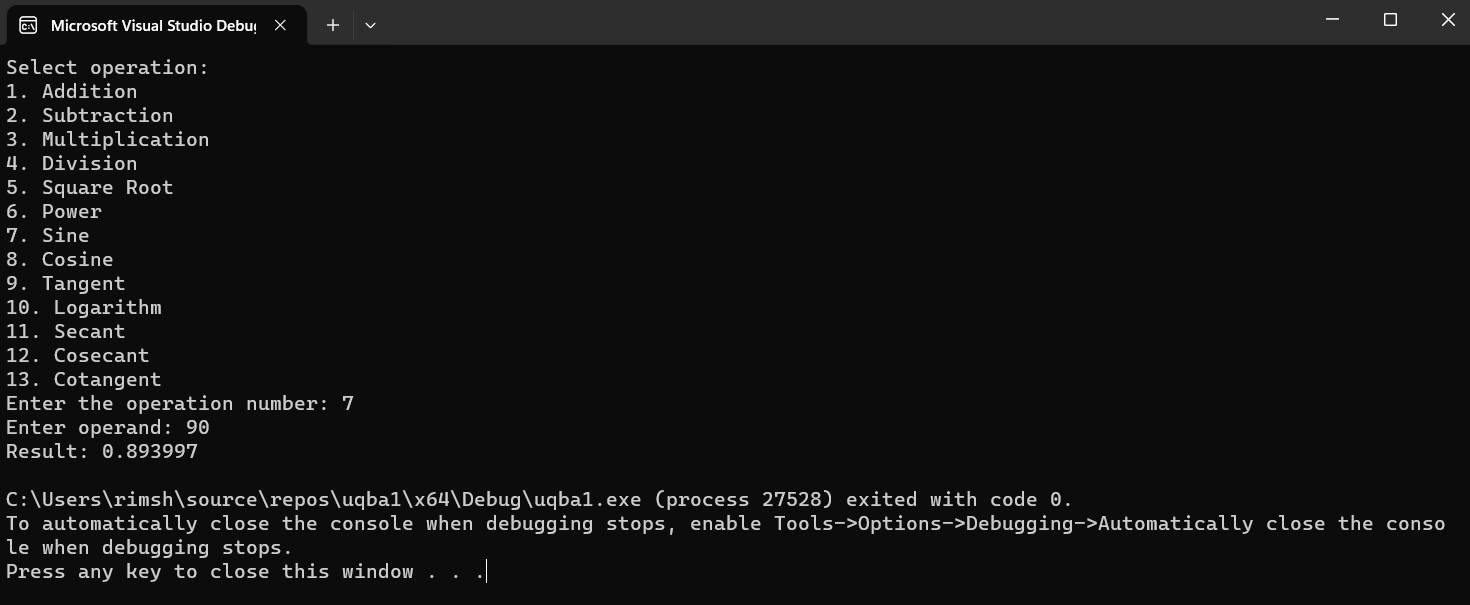
****

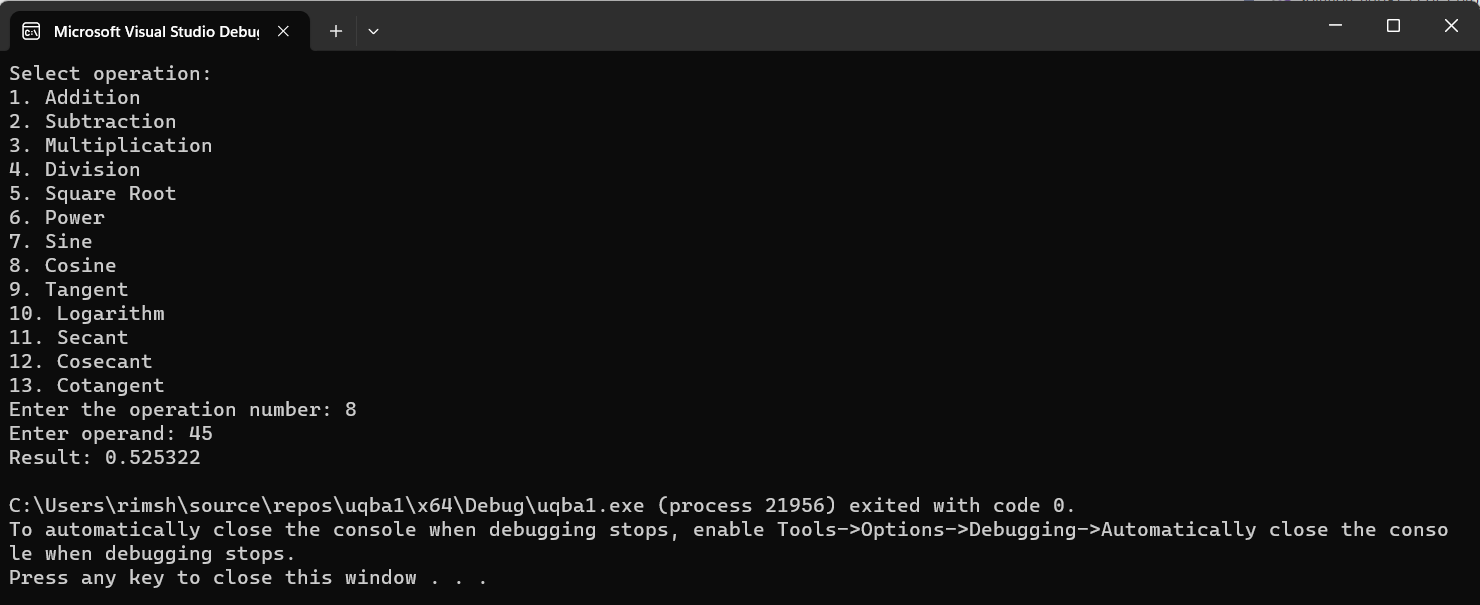
****

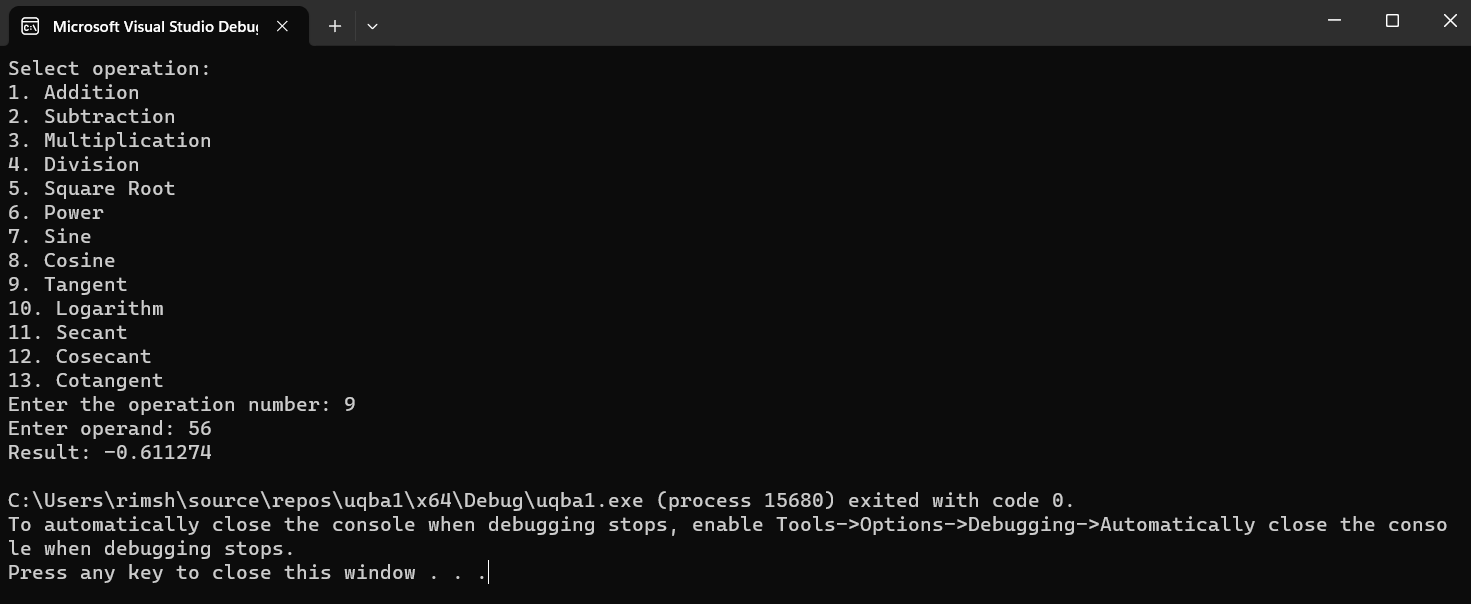
****

****

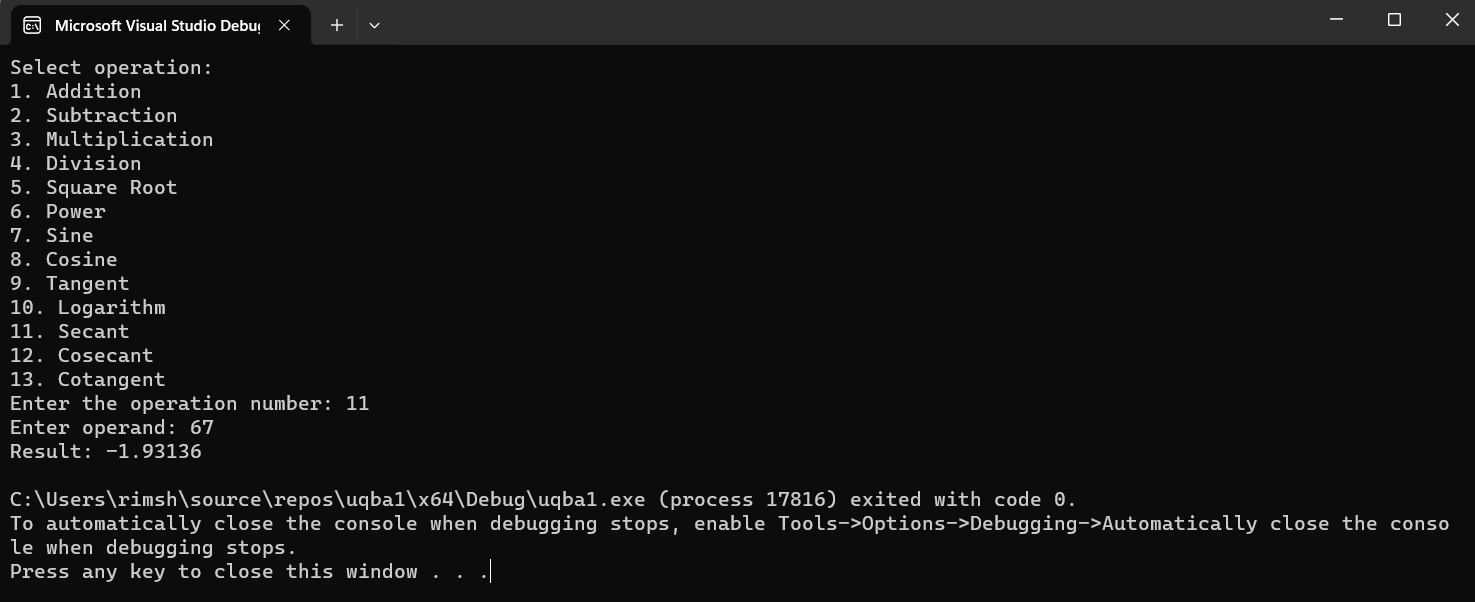
****

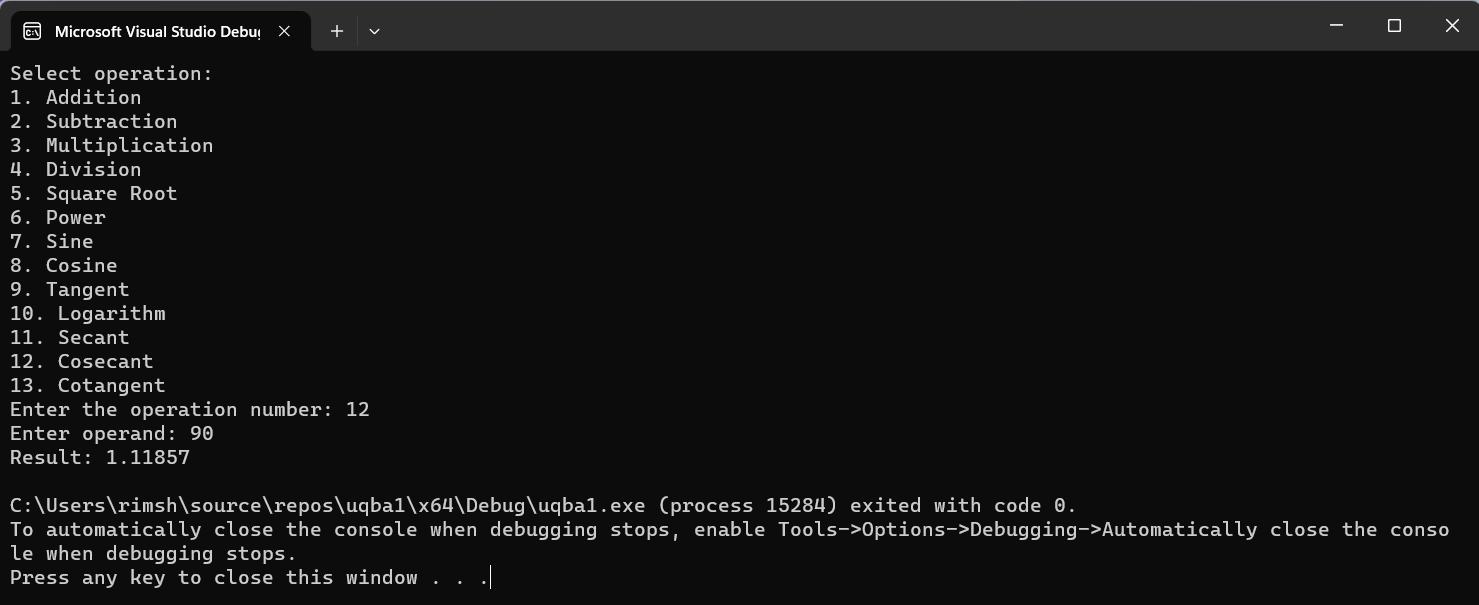
****

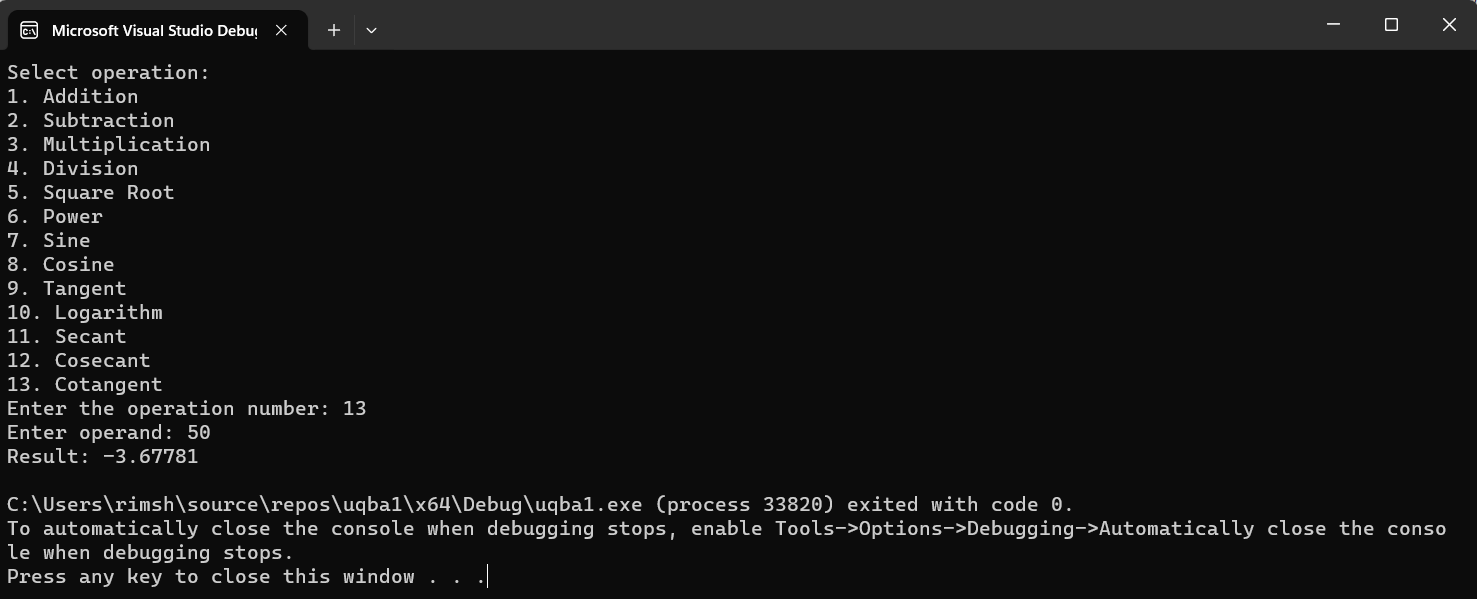
****

****

****

****

****

****