**Assignment 2**

(1).Write a single C++ program :

(i) To find the square root of a number using a function.

[Let the returntype of the function be void]

(ii) To increment a number using an inline function

(iii)To decrement a number using an inline function

**Program:**

#include <iostream>

#include <cmath>

void findSquareRoot(double num) {

double squareRoot = sqrt(num);

std::cout << "Square root of " << num << " is: " << squareRoot << std::endl;

}

inline void incrementNumber(int& num) {

num++;

}

inline void decrementNumber(int& num) {

num--;

}

int main() {

double number = 25.0;

int numToIncrement = 10;

int numToDecrement = 20;

findSquareRoot(number);

std::cout << "Original number to increment: " << numToIncrement << std::endl;

incrementNumber(numToIncrement);

std::cout << "Number after increment: " << numToIncrement << std::endl;

std::cout << "Original number to decrement: " << numToDecrement << std::endl;

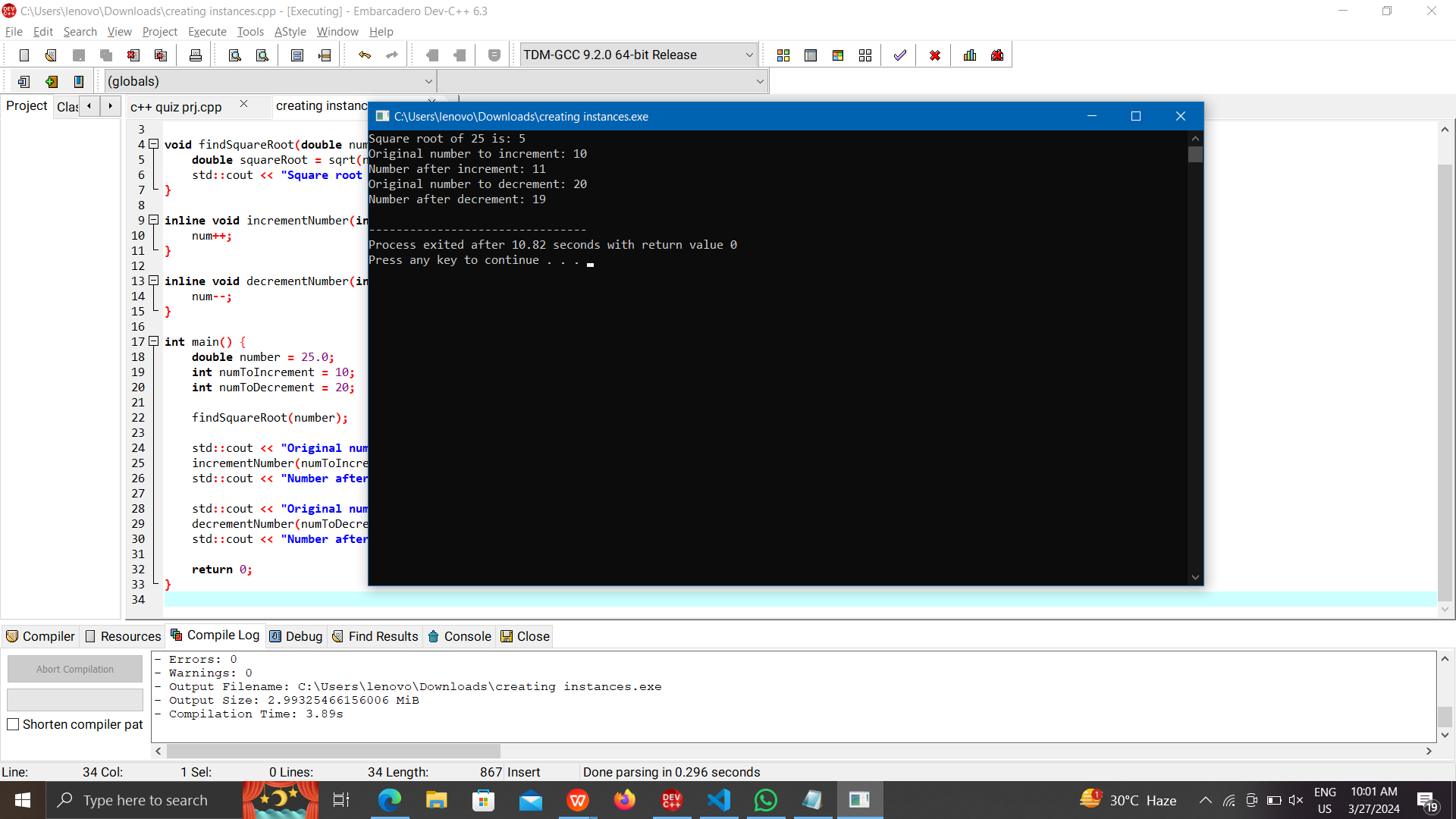
decrementNumber(numToDecrement);

std::cout << "Number after decrement: " << numToDecrement << std::endl;

return 0;

}

**Output:**



(2).Create a class student:

\* Define a function getdata() and get the name,reg.no and 5

marks of a student.

[Note : 1.Use for loop for getting marks,2.Define the function

inside the class]

\* Define a function tot\_marks() and calculate the total marks.

[Note: 1.Use for loop for calculating the total, 2.Define the

function outside the class ]

**Program:**

#include <iostream>

#include <string>

class Student {

private:

std::string name;

int regNo;

int marks[5];

public:

void getData() {

std::cout << "Enter student name: ";

std::cin >> name;

std::cout << "Enter registration number: ";

std::cin >> regNo;

std::cout << "Enter 5 marks: ";

for (int i = 0; i < 5; i++) {

std::cin >> marks[i];

}

}

int getTotalMarks();

};

int Student::getTotalMarks() {

int total = 0;

for (int i = 0; i < 5; i++) {

total += marks[i];

}

return total;

}

int main() {

Student student;

student.getData();

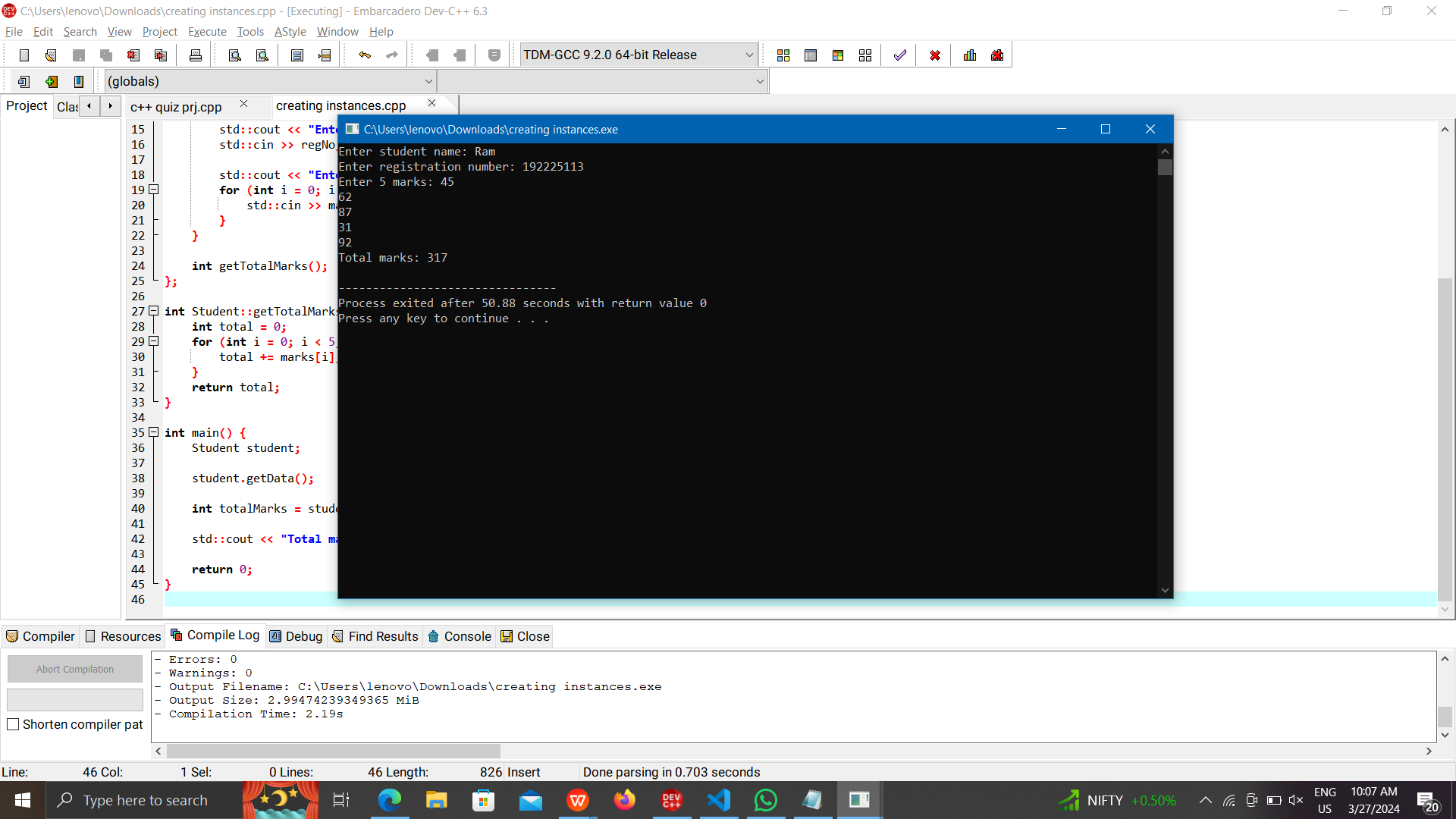
int totalMarks = student.getTotalMarks();

std::cout << "Total marks: " << totalMarks << std::endl;

return 0;

}

**Output:**



(3). Create a class product.

\* Define a function get\_product() and get the name of the

product and its price.

\* Define a function print\_product() and display the product and

its price.

\* Create an array of object to call the above functions.[Note:

Array size: generalized]

**Program:**

#include <iostream>

#include <string>

class Product {

private:

std::string name;

double price;

public:

void getProduct() {

std::cout << "Enter product name: ";

std::cin >> name;

std::cout << "Enter product price: ";

std::cin >> price;

}

void printProduct() {

std::cout << "Product: " << name << std::endl;

std::cout << "Price: $" << price << std::endl;

}

};

int main() {

int numProducts;

std::cout << "Enter the number of products: ";

std::cin >> numProducts;

Product products[numProducts];

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

std::cout << "\nEnter details for Product " << i + 1 << std::endl;

products[i].getProduct();

}

std::cout << "\nProduct Details:\n";

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

std::cout << "\nProduct " << i + 1 << ":\n";

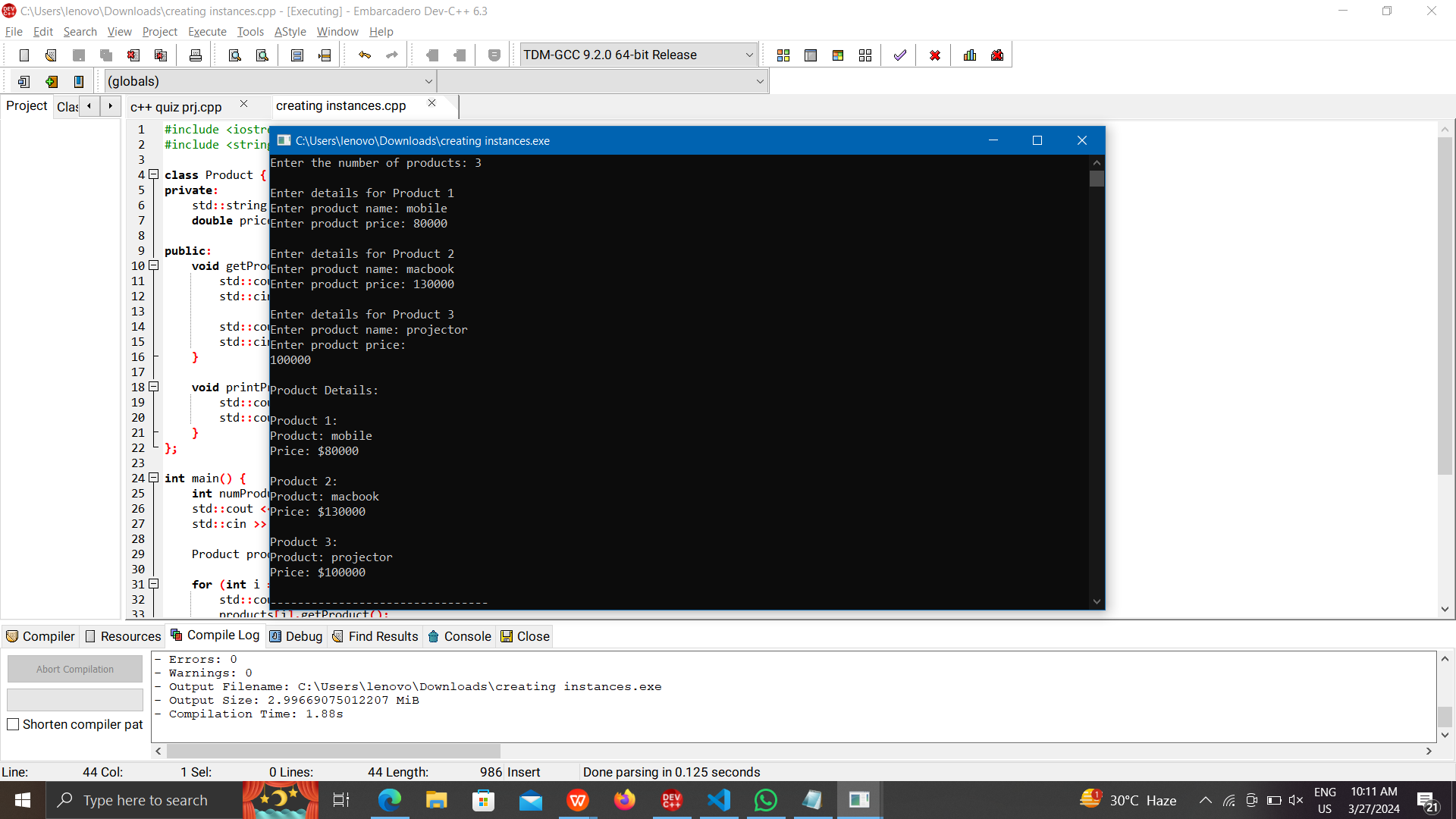
products[i].printProduct();

}

return 0;

}

**Output:**



(4). Write a C++ program to find the maximum of 2 numbers using

a friend function.

\* Each number should be got in 2 different classes.

\* Define a friend function max which is common to both the

class.

**Program:** #include <iostream>

class Number2;

class Number1 {

private:

int num1;

public:

Number1(int n) : num1(n) {}

friend int max(const Number1& n1, const Number2& n2);

};

class Number2 {

private:

int num2;

public:

Number2(int n) : num2(n) {}

friend int max(const Number1& n1, const Number2& n2);

}

int max(const Number1& n1, const Number2& n2) {

return (n1.num1 > n2.num2) ? n1.num1 : n2.num2;

}

int main() {

int num1, num2;

std::cout << "Enter first number: ";

std::cin >> num1;

std::cout << "Enter second number: ";

std::cin >> num2;

Number1 n1(num1);

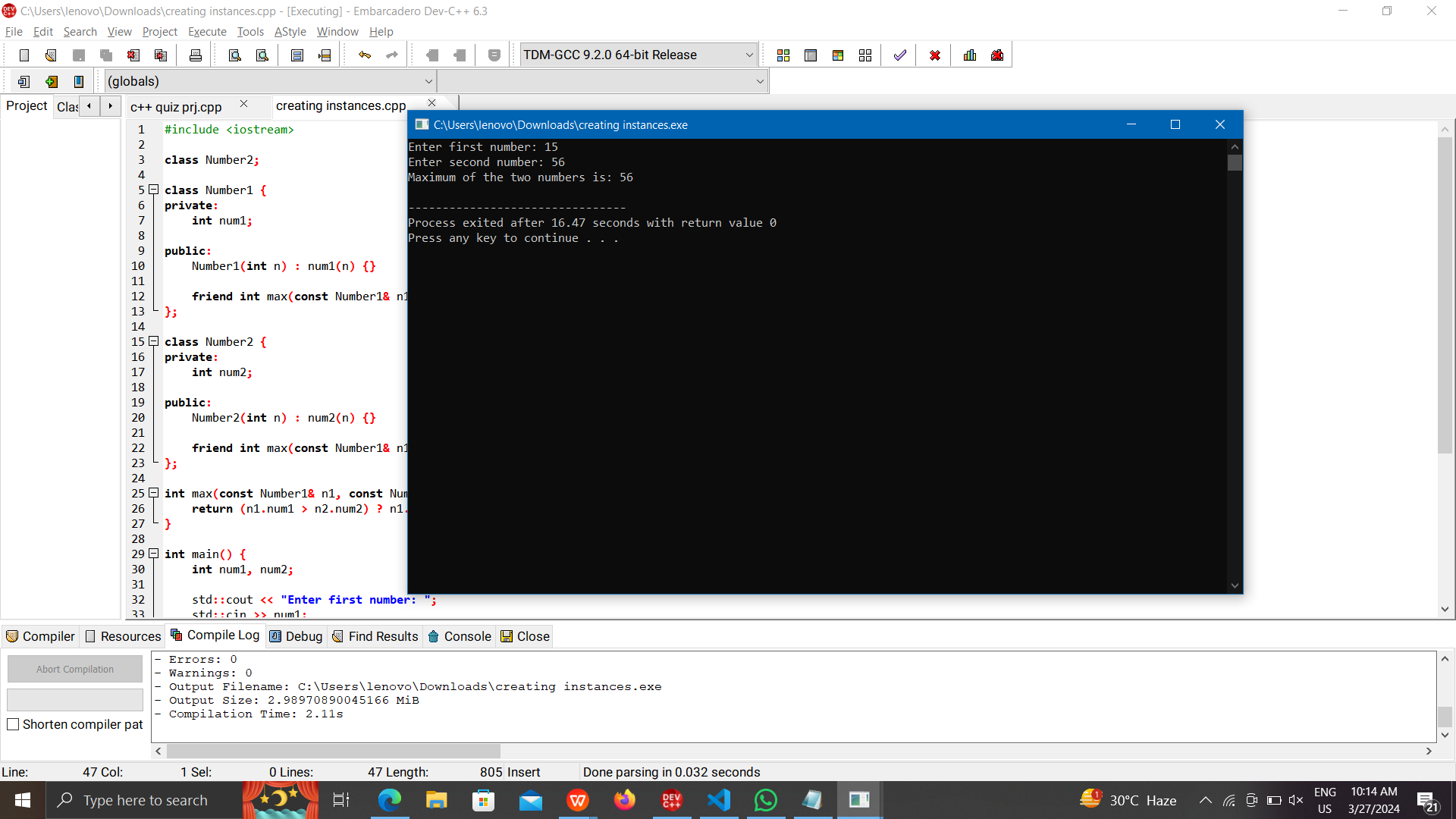
Number2 n2(num2);

int maximum = max(n1, n2);

std::cout << "Maximum of the two numbers is: " << maximum << std::endl;

return 0;

}

**Output:**

(5). Implement a banking system using C++ classes. Create classes

for customers, accounts, and transactions. Apply encapsulation

to protect sensitive information, and demonstrate the use of

friend functions for access control. Discuss how encapsulation

enhances the security and maintainability of the system.

**Program:** #include <iostream>

#include <string>

class Account {

public:

std::string accountNumber;

double balance;

Account(const std::string& accNum, double bal) : accountNumber(accNum), balance(bal) {}

void deposit(double amount) {

balance += amount;

std::cout << "Deposit of $" << amount << " successful. New balance: $" << balance << std::endl;

}

void withdraw(double amount) {

if (amount <= balance) {

balance -= amount;

std::cout << "Withdrawal of $" << amount << " successful. New balance: $" << balance << std::endl;

} else {

std::cout << "Insufficient funds. Withdrawal failed." << std::endl;

}

}

friend void transfer(Account& from, Account& to, double amount);

};

void transfer(Account& from, Account& to, double amount) {

if (amount <= from.balance) {

from.balance -= amount;

to.balance += amount;

std::cout << "Transfer of $" << amount << " successful." << std::endl;

} else {

std::cout << "Transfer failed. Insufficient funds in the source account." << std::endl;

}

}

int main() {

std::string accNum1 = "1001";

std::string accNum2 = "2002";

Account account1(accNum1, 5000.0);

Account account2(accNum2, 3000.0);

double amount;

std::cout << "Account 1 Balance: $" << account1.balance << std::endl;

std::cout << "Account 2 Balance: $" << account2.balance << std::endl;

std::cout << "Enter amount to transfer from Account 1 to Account 2: ";

std::cin >> amount;

transfer(account1, account2, amount);

std::cout << "Updated Account 1 Balance: $" << account1.balance << std::endl;

std::cout << "Updated Account 2 Balance: $" << account2.balance << std::endl;

return 0;

}

**Output:**

