**C++ ASSIGNMENT 1.2**

1. WAP for printing all natural numbers till 20.

#include <stdio.h>

#include <iostream>

using namespace std;

int main() {

// Iterate from 1 to 20 and print each number

for (int i = 1; i <= 20; ++i) {

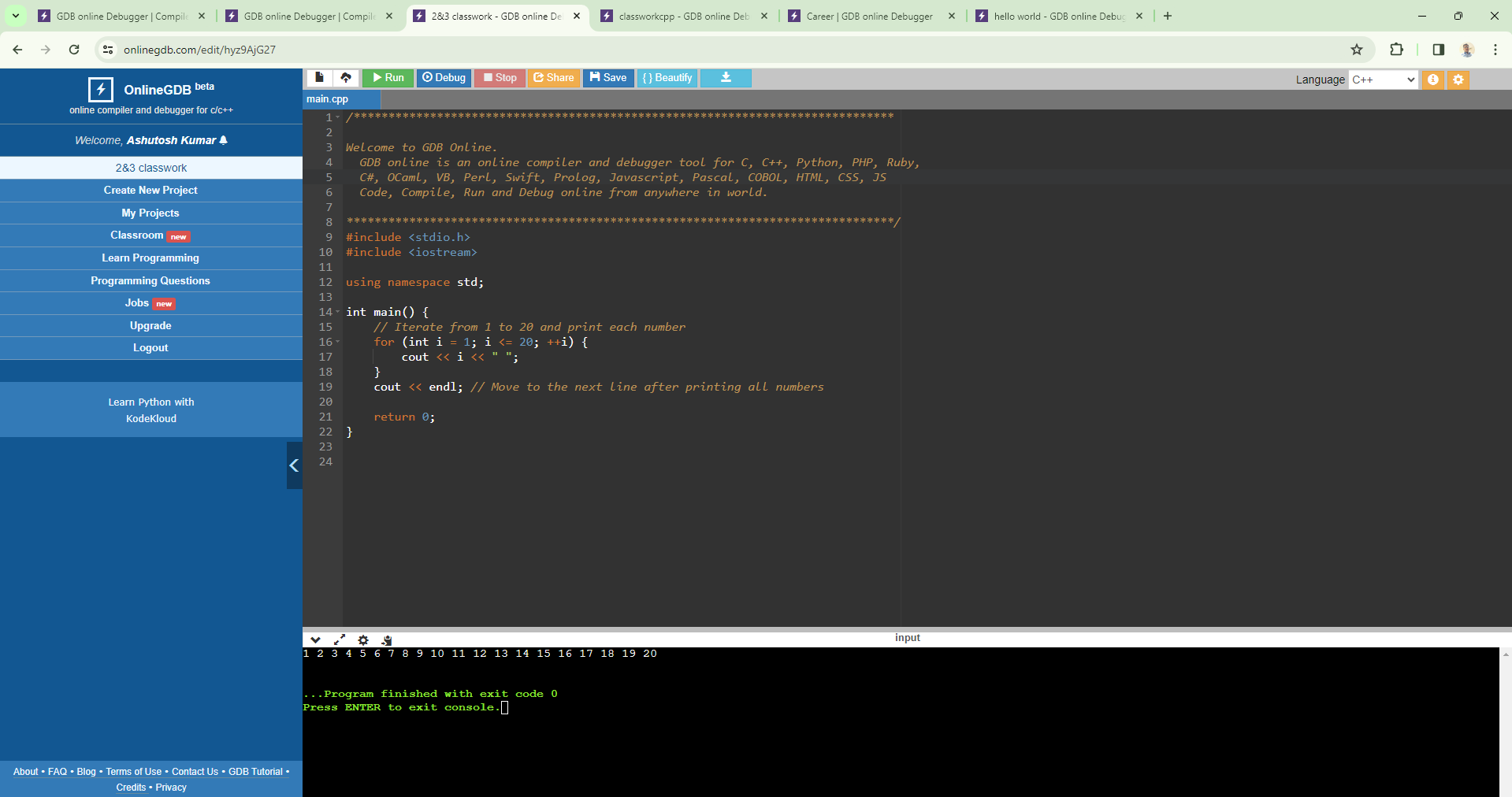
cout << i << " ";

}

cout << endl; // Move to the next line after printing all numbers

return 0;

}



1. WAP for printing all natural numbers in reverse order starting from 20.

#include <iostream>

using namespace std;

int main() {

// Iterate from 20 down to 1 and print each number

for (int i = 20; i >= 1; --i) {

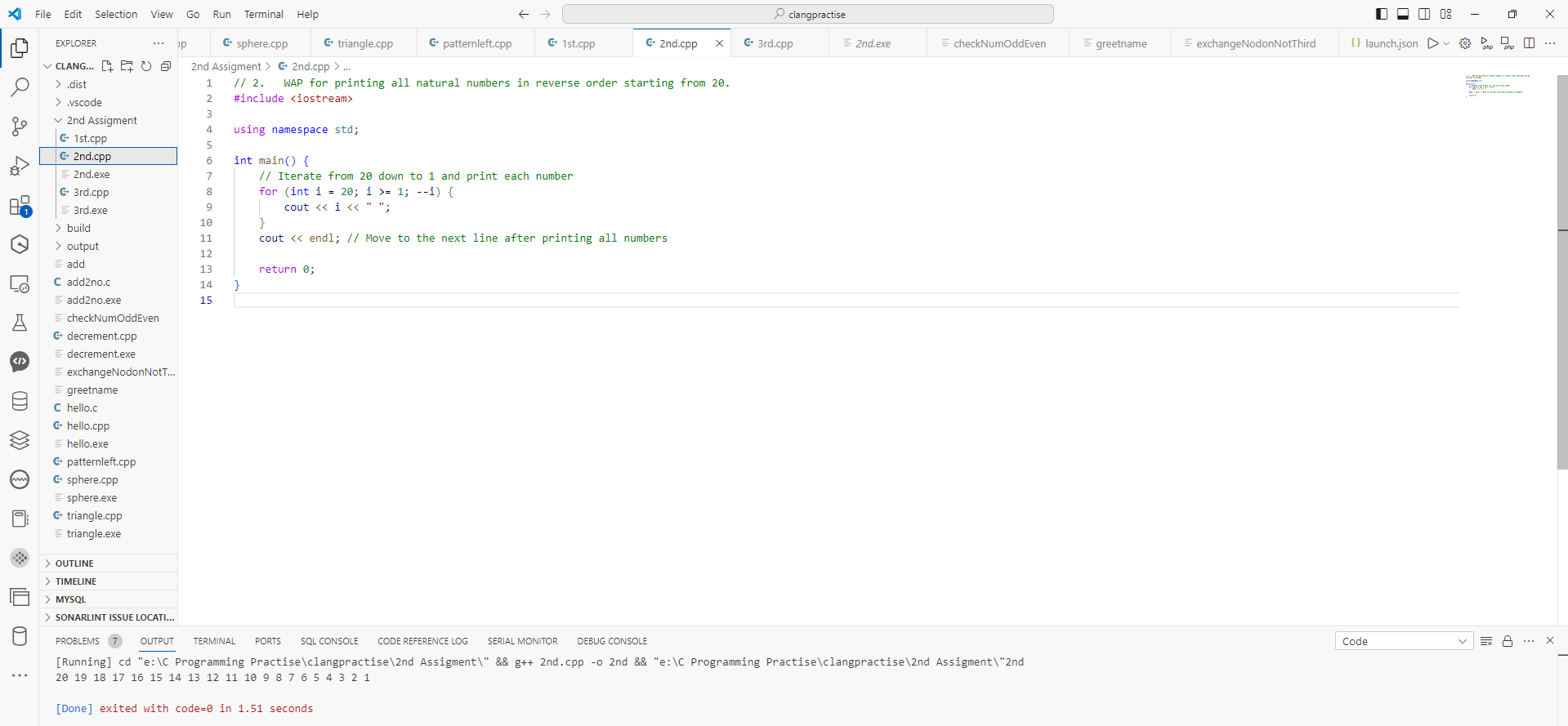
cout << i << " ";

}

cout << endl; // Move to the next line after printing all numbers

return 0;

}



1. WAP for printing all even numbers from 1 to 20.

#include <iostream>

using namespace std;

int main() {

// Iterate from 1 to 20 and print each even number

for (int i = 1; i <= 20; ++i) {

// Check if the number is even

if (i % 2 == 0) {

cout << i << " ";

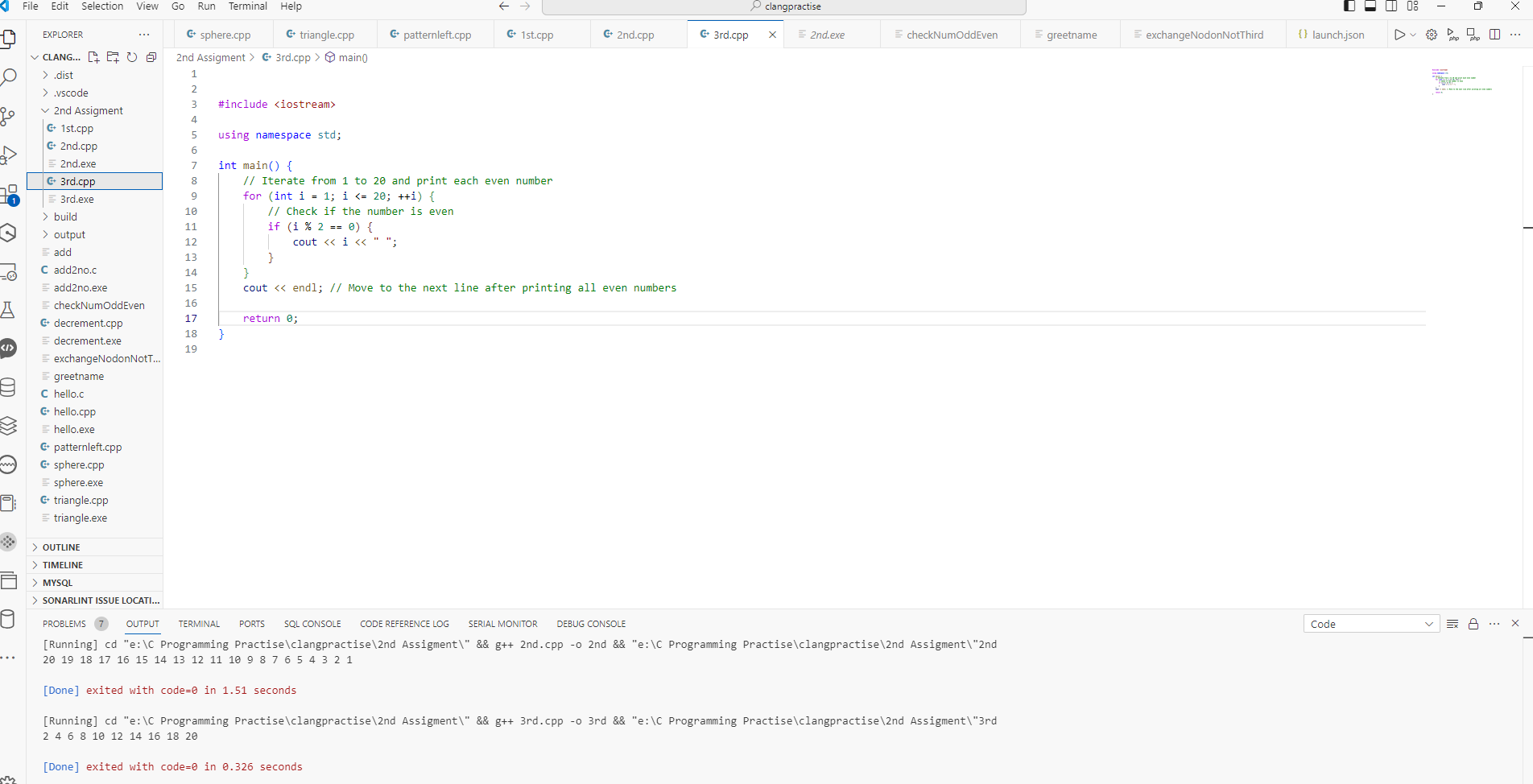
}

}

cout << endl; // Move to the next line after printing all even numbers

return 0;

}



1. WAP for printing all odd numbers from 1 to 20.

#include <iostream>

using namespace std;

int main() {

// Iterate from 1 to 20 and print each odd number

for (int i = 1; i <= 20; ++i) {

// Check if the number is odd

if (i % 2 != 0) {

cout << i << " ";

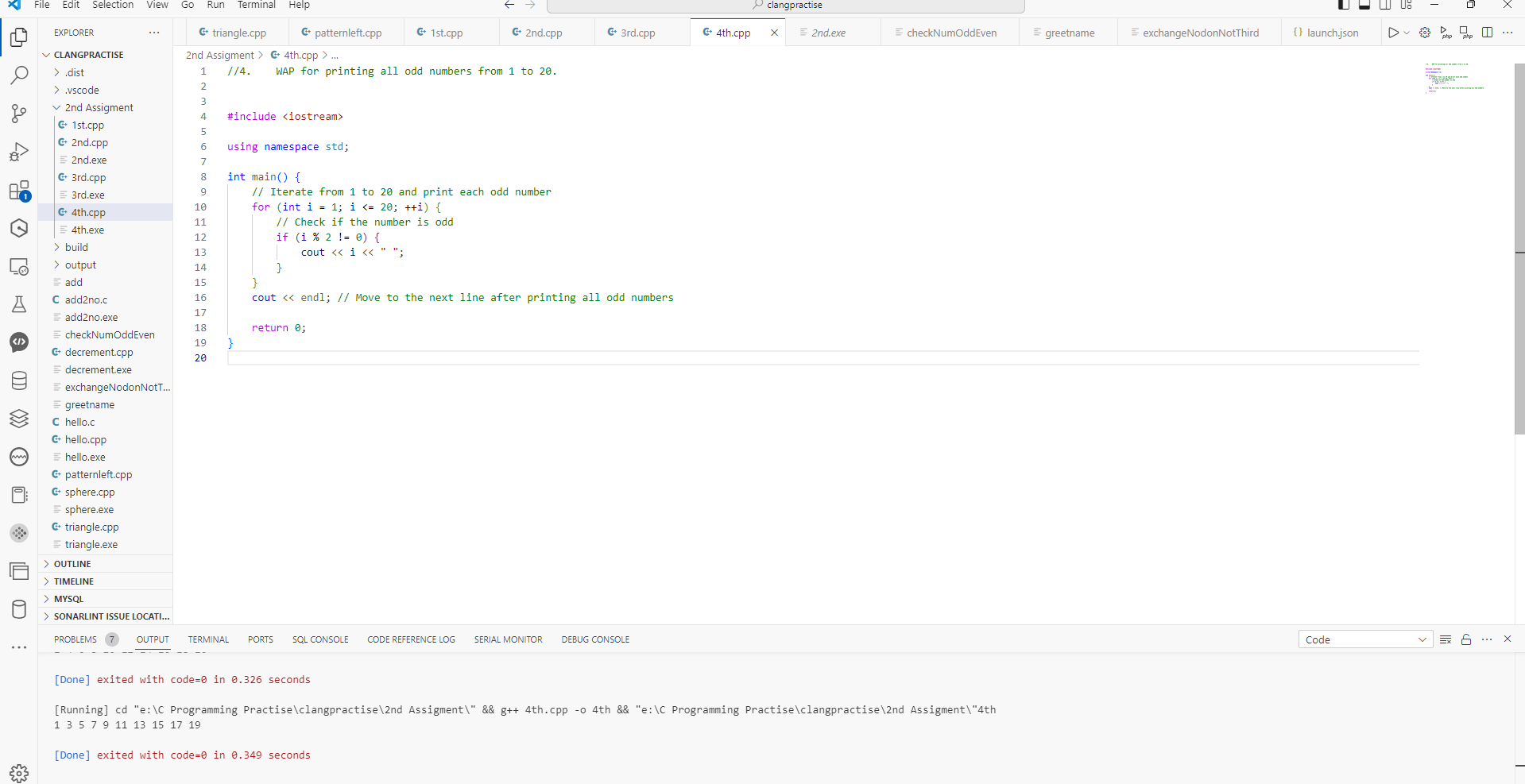
}

}

cout << endl; // Move to the next line after printing all odd numbers

return 0;

}



1. WAP for adding all numbers from 1 to 20.

#include <iostream>

using namespace std;

int main() {

int sum = 0; // Variable to store the sum

// Iterate from 1 to 20 and add each number to the sum

for (int i = 1; i <= 20; ++i) {

sum += i;

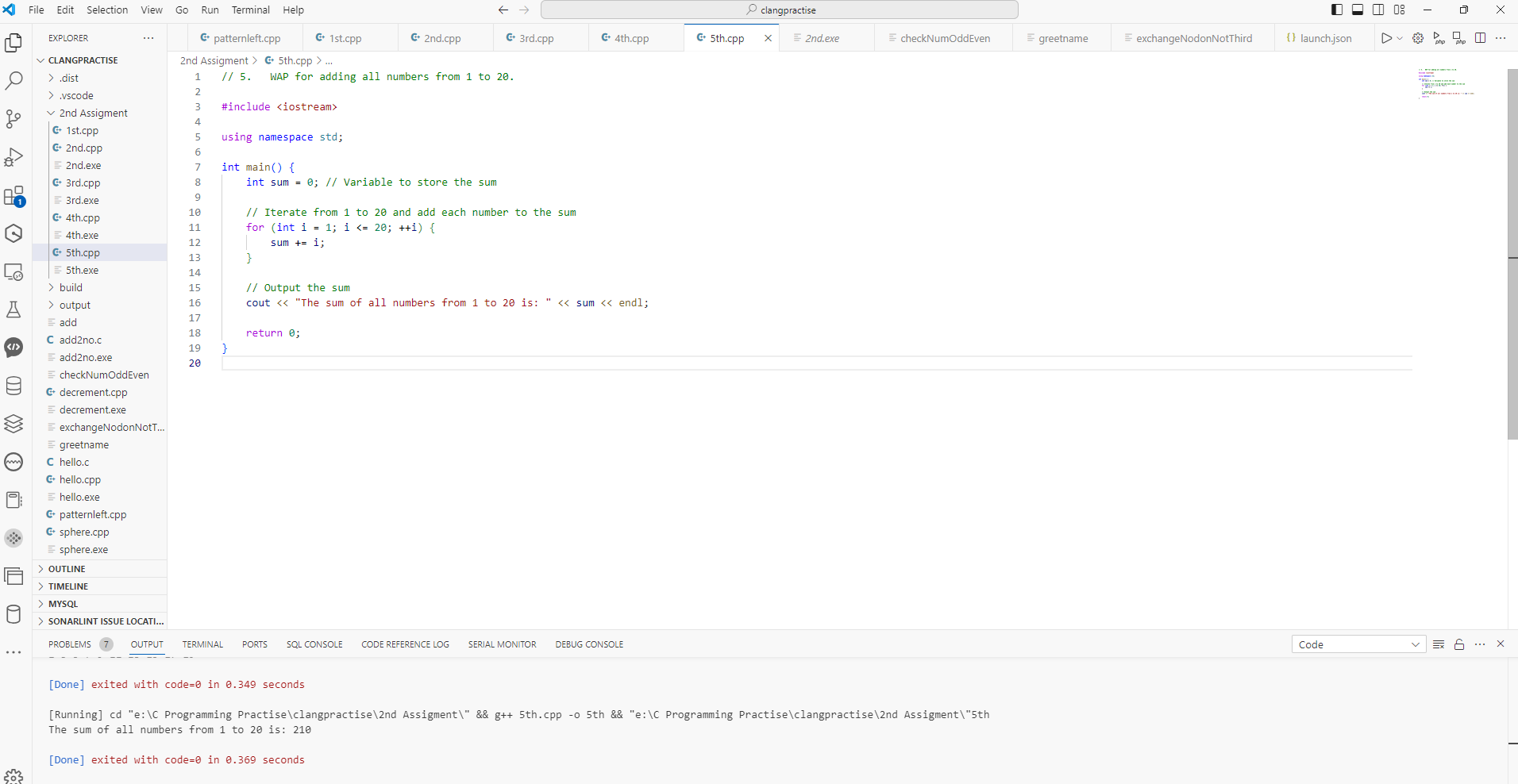
}

// Output the sum

cout << "The sum of all numbers from 1 to 20 is: " << sum << endl;

return 0;

}



1. WAP for finding sum of all even numbers till 20.

#include <iostream>

using namespace std;

int main() {

int sum = 0; // Variable to store the sum

// Iterate from 2 to 20 (even numbers) and add each number to the sum

for (int i = 2; i <= 20; i += 2) {

sum += i;

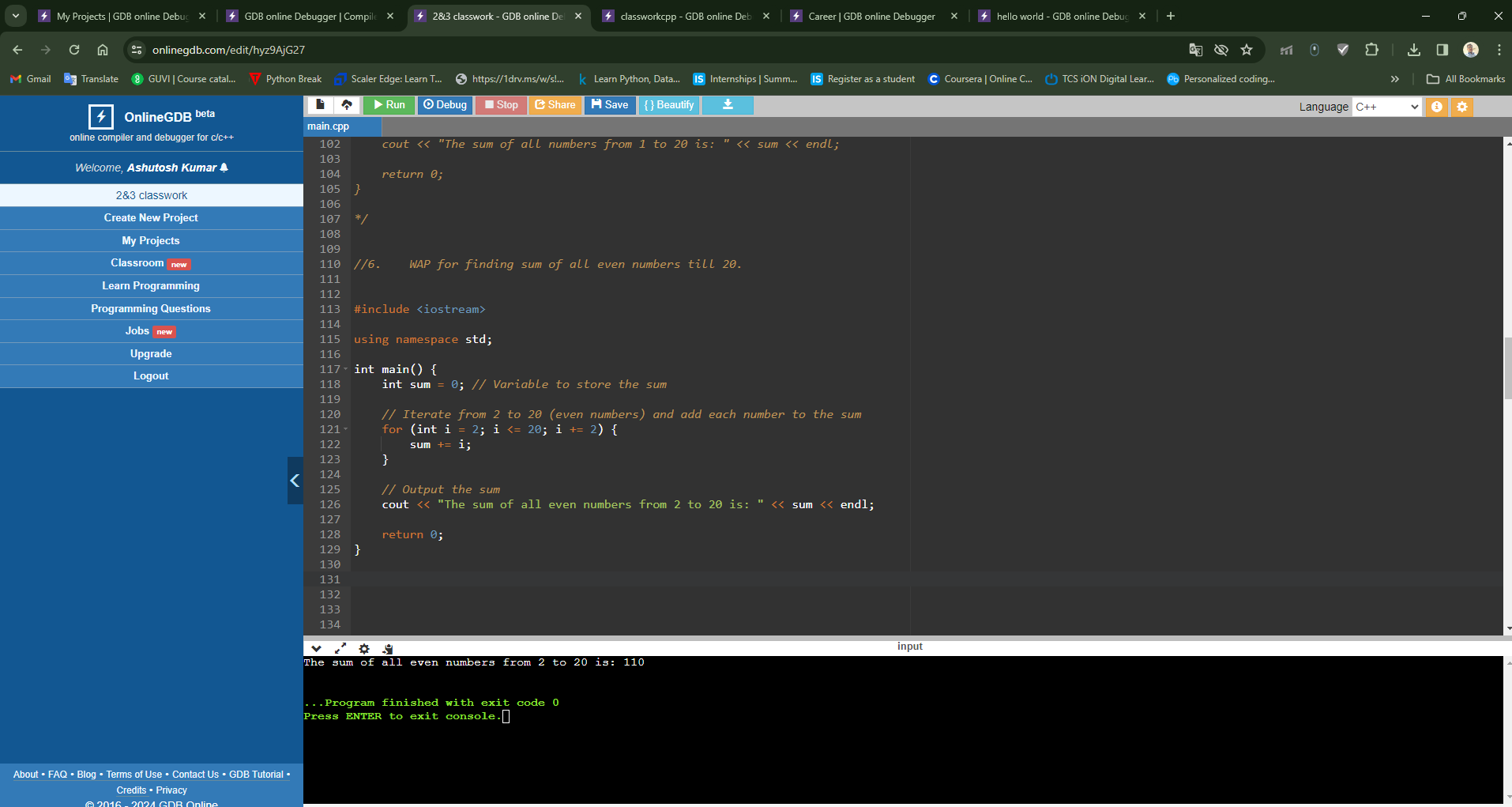
}

// Output the sum

cout << "The sum of all even numbers from 2 to 20 is: " << sum << endl;

return 0;

}



1. WAP for finding sum of all odd numbers till 20.

#include <iostream>

using namespace std;

int main() {

int sum = 0; // Variable to store the sum

// Iterate from 1 to 20 and add each odd number to the sum

for (int i = 1; i <= 20; i += 2) {

sum += i;

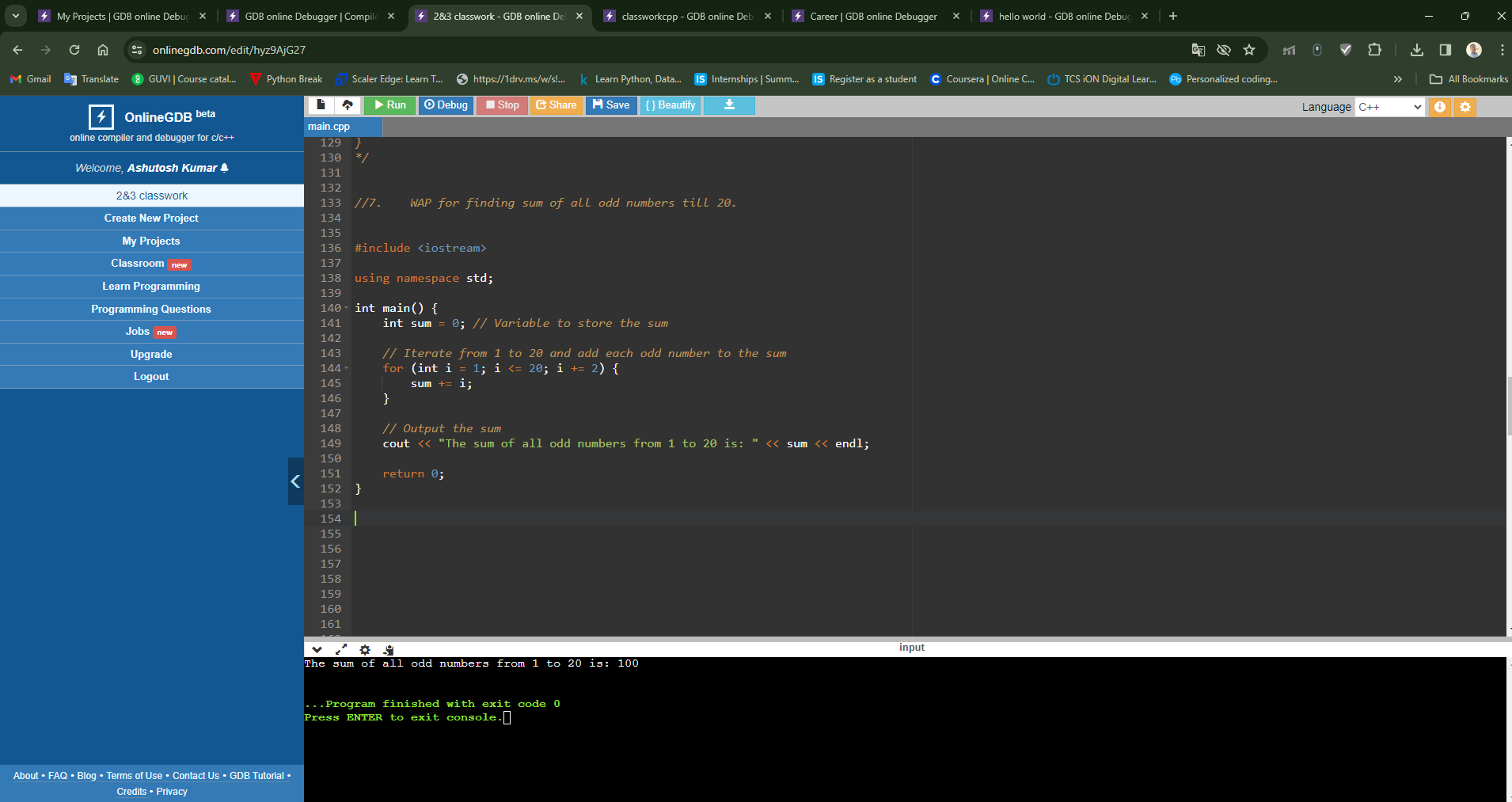
}

// Output the sum

cout << "The sum of all odd numbers from 1 to 20 is: " << sum << endl;

return 0;

}



1. WAP for printing multiplication table of a number. For eg. Display should be “ 2 X 1 = 2”

#include <iostream>

using namespace std;

int main() {

int number;

// Input the number for which multiplication table is to be printed

cout << "Enter the number: ";

cin >> number;

// Print the multiplication table

cout << "Multiplication table of " << number << ":" << endl;

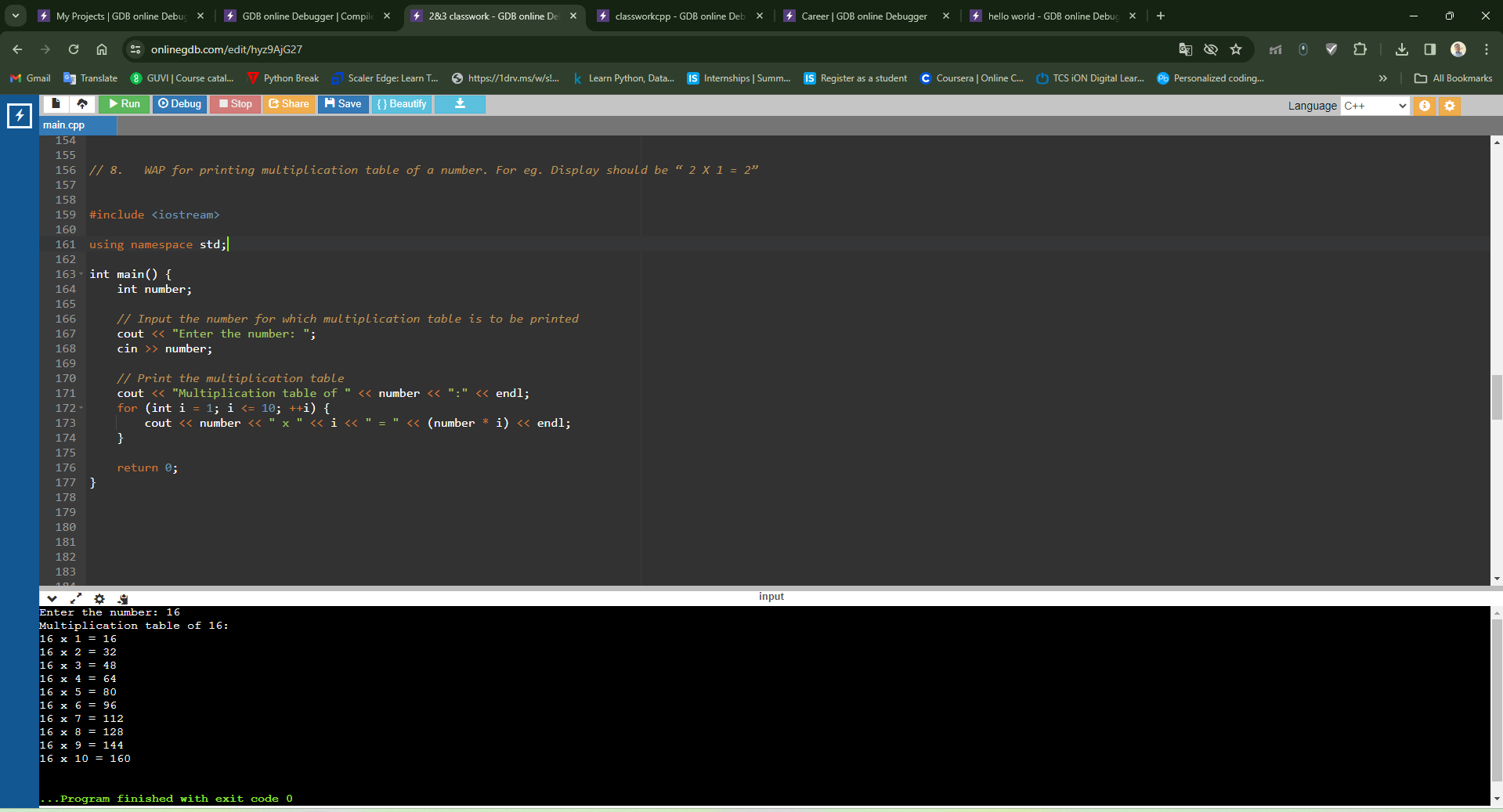
for (int i = 1; i <= 10; ++i) {

cout << number << " x " << i << " = " << (number \* i) << endl;

}

return 0;

}



1. WAP to calculate factorial of a number.

#include <iostream>

using namespace std;

// Function to calculate factorial

int factorial(int n) {

// Base case: Factorial of 0 is 1

if (n == 0) {

return 1;

}

// Recursive case: Factorial of n is n \* factorial(n - 1)

else {

return n \* factorial(n - 1);

}

}

int main() {

int number;

// Input the number for which factorial is to be calculated

cout << "Enter the number: ";

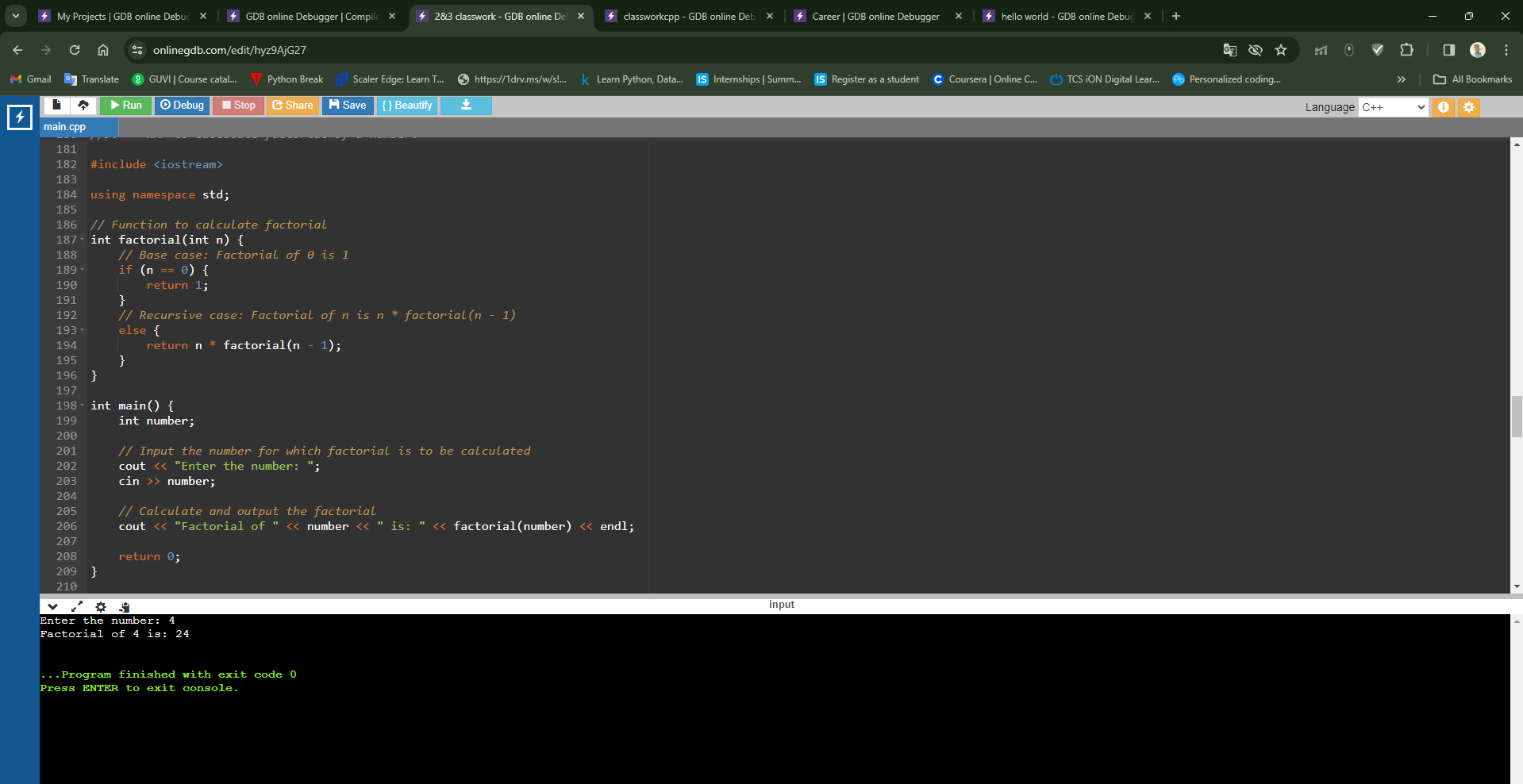
cin >> number;

// Calculate and output the factorial

cout << "Factorial of " << number << " is: " << factorial(number) << endl;

return 0;

}



1. WAP to check whether a number is prime or not.

#include <iostream>

using namespace std;

int main() {

int number;

bool isPrime = true;

// Input the number

cout << "Enter a number: ";

cin >> number;

// Check if the number is less than 2

if (number < 2) {

isPrime = false;

} else {

// Check for divisibility from 2 to the square root of the number

for (int i = 2; i \* i <= number; ++i) {

if (number % i == 0) {

isPrime = false; // If the number is divisible, it's not prime

break;

}

}

}

// Output the result

if (isPrime) {

cout << number << " is a prime number." << endl;

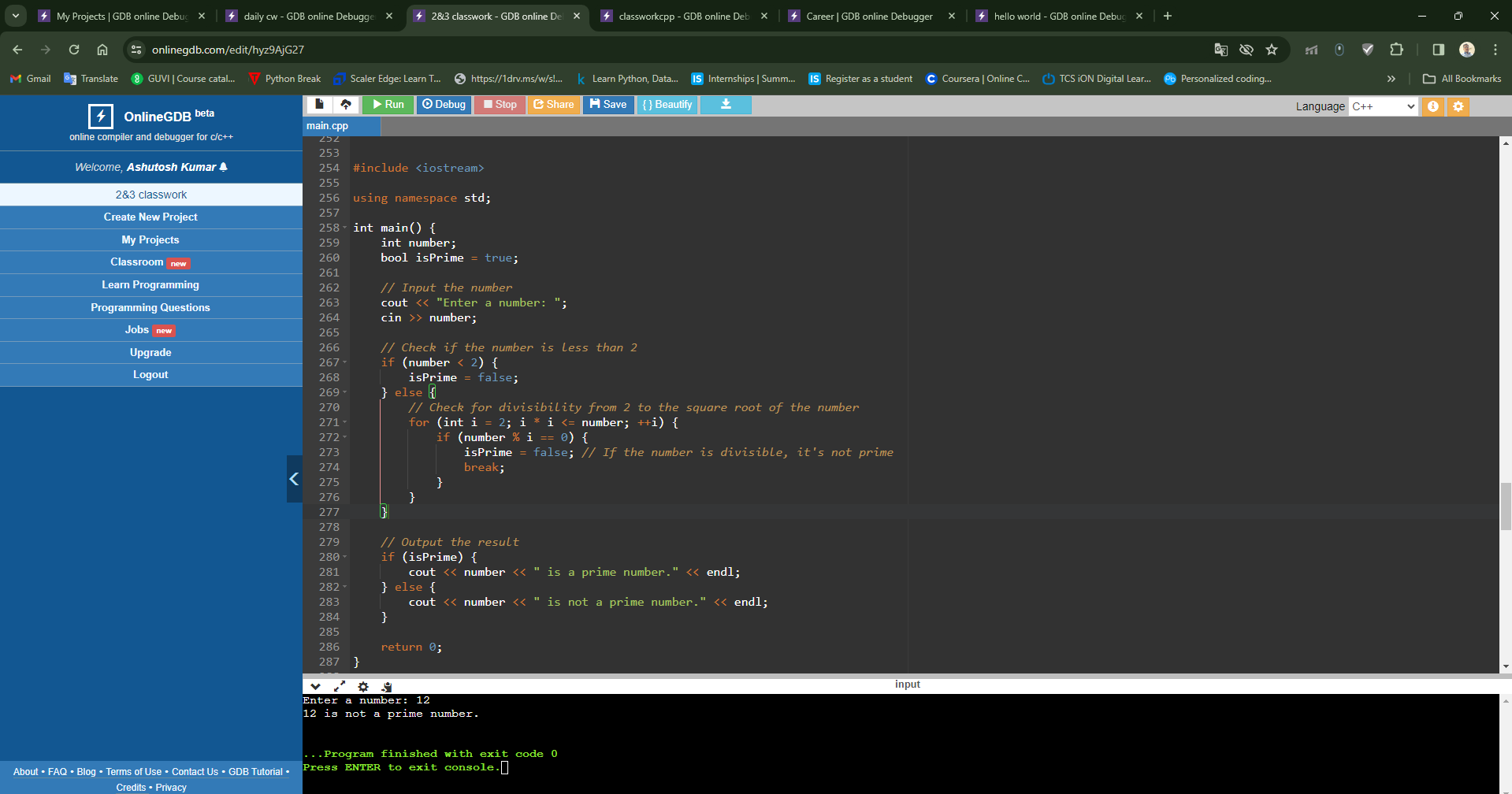
} else {

cout << number << " is not a prime number." << endl;

}

return 0;

}



1. WAP to print all digits of a number and their sum.

#include <iostream>

using namespace std;

int main() {

int number, digit, sum = 0;

// Input the number

cout << "Enter a number: ";

cin >> number;

// Temporary variable to store the original number

int originalNumber = number;

// Loop to extract each digit and calculate their sum

while (number > 0) {

// Extract the last digit

digit = number % 10;

// Add the digit to the sum

sum += digit;

// Output the digit

cout << "Digit: " << digit << endl;

// Remove the last digit

number /= 10;

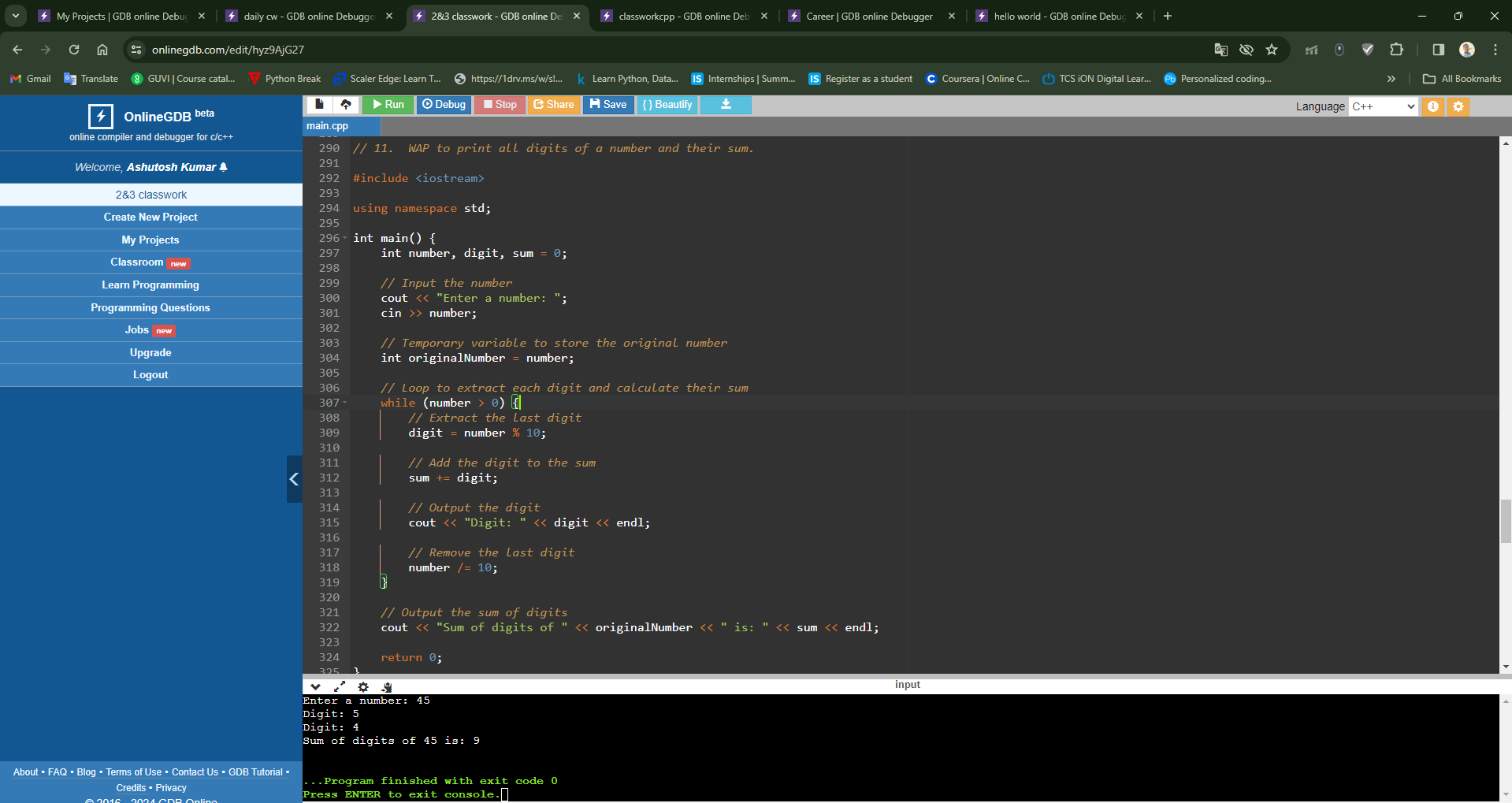
}

// Output the sum of digits

cout << "Sum of digits of " << originalNumber << " is: " << sum << endl;

return 0;

}



1. WAP to print reverse of a number.

#include <iostream>

using namespace std;

int main() {

int number, reversedNumber = 0;

// Input the number

cout << "Enter a number: ";

cin >> number;

// Reverse the number

while (number > 0) {

// Append the last digit of the number to the reversed number

reversedNumber = reversedNumber \* 10 + number % 10;

// Remove the last digit from the number

number /= 10;

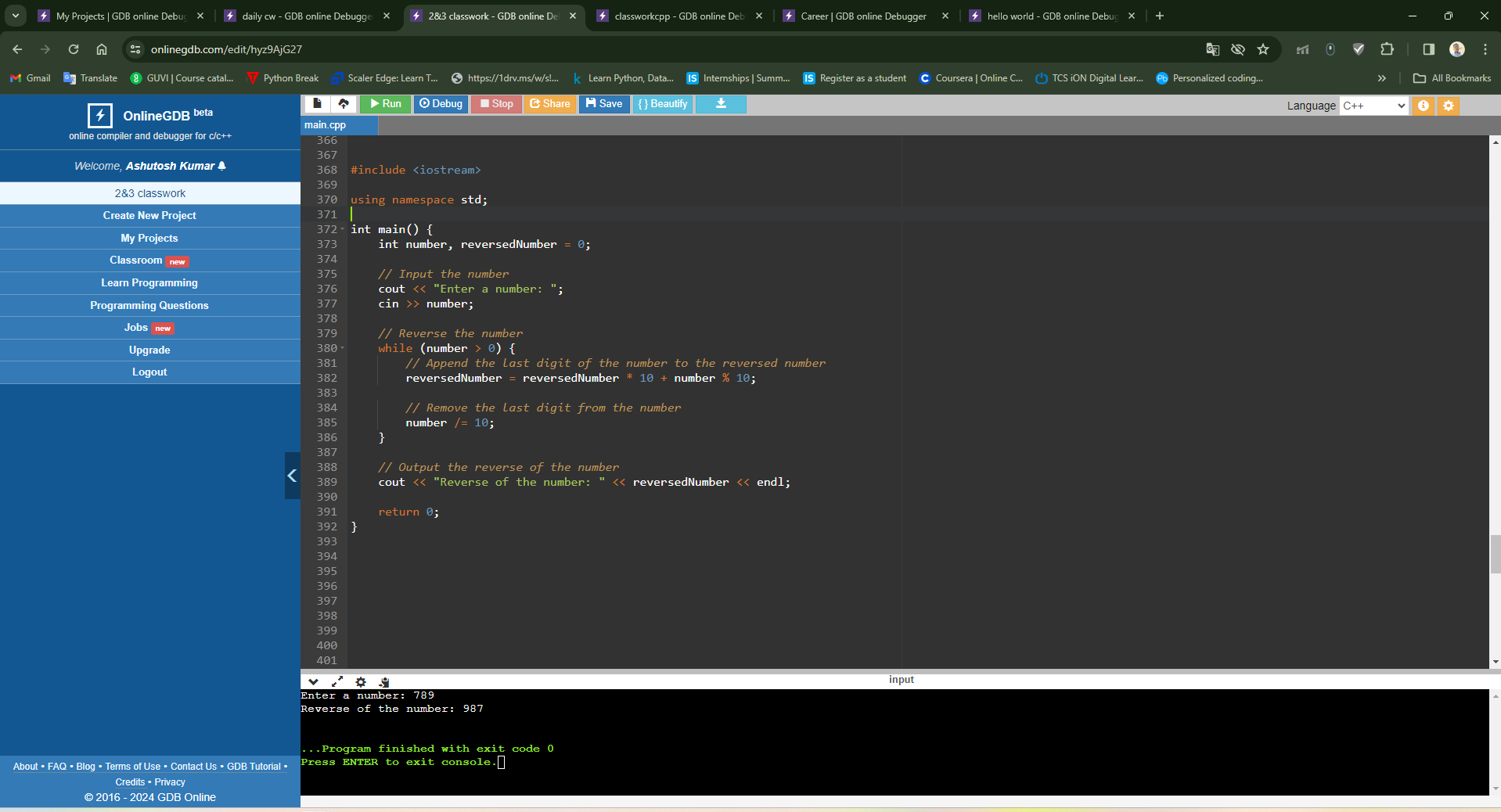
}

// Output the reverse of the number

cout << "Reverse of the number: " << reversedNumber << endl;

return 0;

}



1. WAP to check whether the number is Armstrong or not.

#include <iostream>

#include <cmath>

using namespace std;

int main() {

int number, originalNumber, numDigits = 0, sum = 0;

// Input the number

cout << "Enter a number: ";

cin >> number;

// Store the original number

originalNumber = number;

// Count the number of digits

while (originalNumber != 0) {

originalNumber /= 10;

++numDigits;

}

// Reset originalNumber to the actual number

originalNumber = number;

// Calculate the sum of digits raised to the power of numDigits

while (originalNumber != 0) {

int digit = originalNumber % 10;

sum += pow(digit, numDigits);

originalNumber /= 10;

}

// Check if the number is an Armstrong number

if (sum == number) {

cout << number << " is an Armstrong number." << endl;

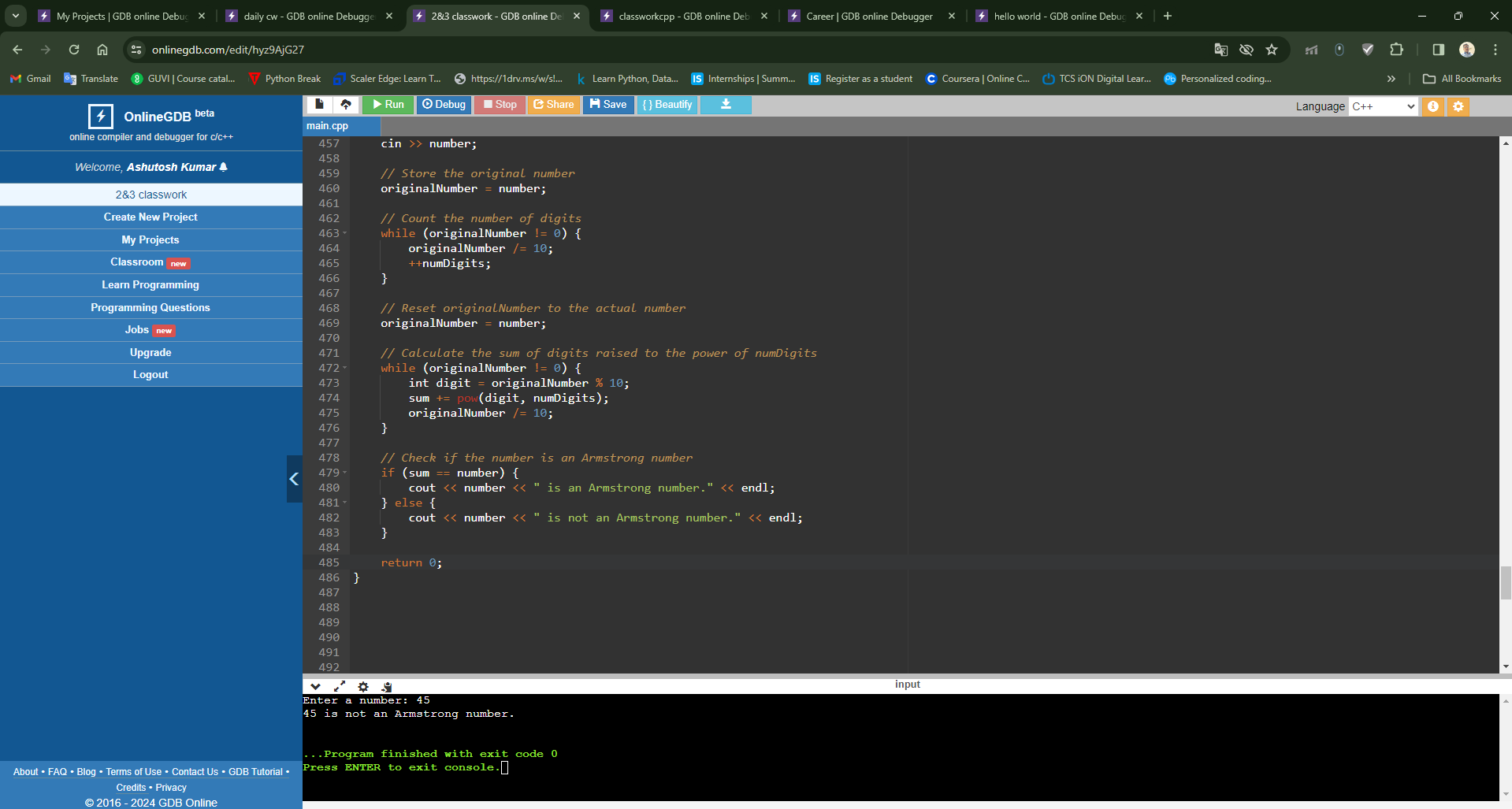
} else {

cout << number << " is not an Armstrong number." << endl;

}

return 0;

}



1. WAP to print the Fibonacci series in a given range.

#include <iostream>

using namespace std;

int main() {

int n, firstTerm = 0, secondTerm = 1, nextTerm;

// Input the range

cout << "Enter the range (number of terms): ";

cin >> n;

cout << "Fibonacci Series up to " << n << " terms: ";

// Print the first two terms of the Fibonacci series

cout << firstTerm << " " << secondTerm << " ";

// Generate and print the rest of the Fibonacci series

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

nextTerm = firstTerm + secondTerm;

cout << nextTerm << " ";

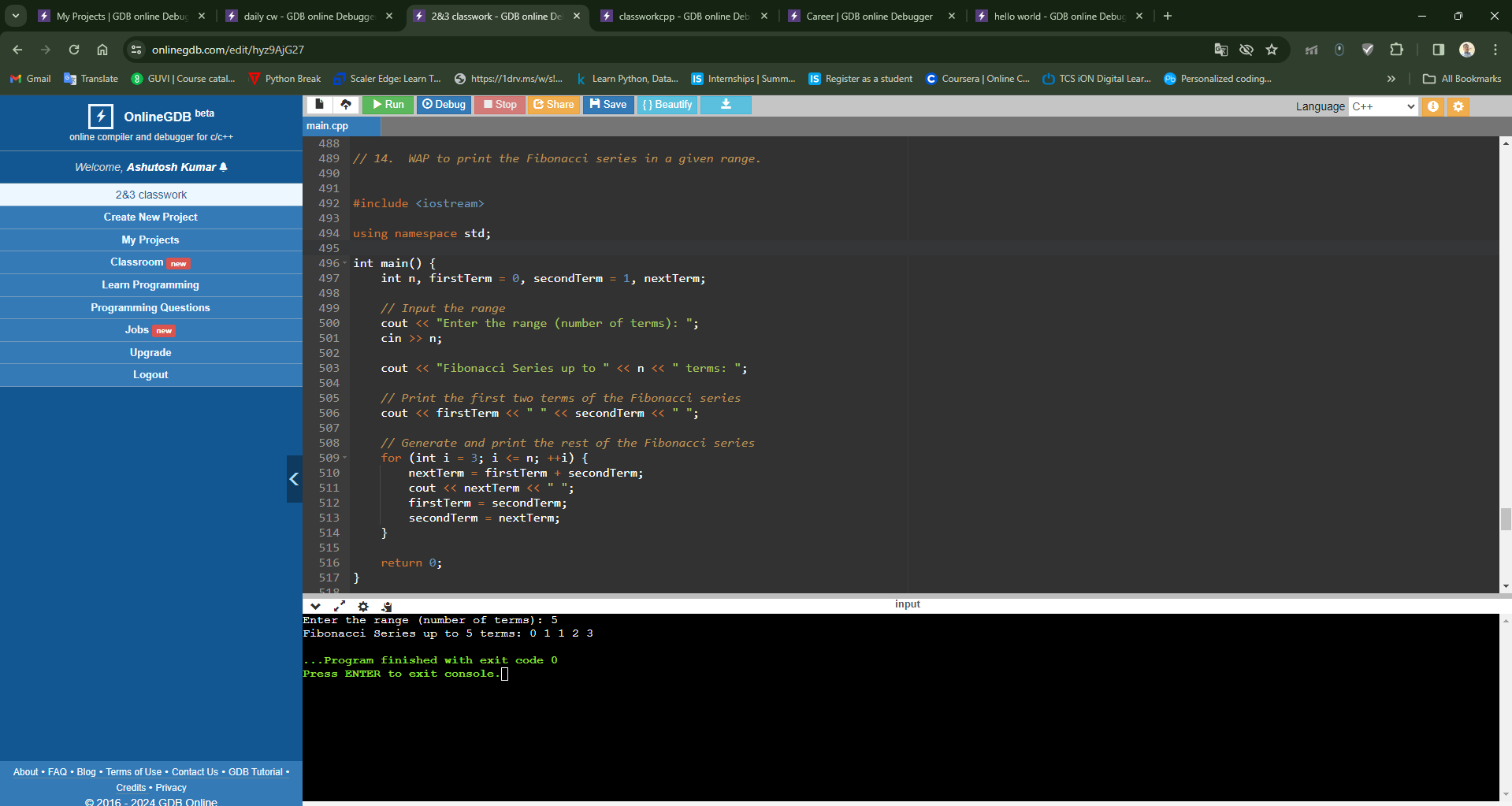
firstTerm = secondTerm;

secondTerm = nextTerm;

}

return 0;

}



1. WAP to check whether the number entered is palindrome or not.

#include <iostream>

using namespace std;

int main() {

int number, originalNumber, reversedNumber = 0, remainder;

// Input the number

cout << "Enter a number: ";

cin >> number;

// Store the original number

originalNumber = number;

// Reverse the number

while (number != 0) {

remainder = number % 10;

reversedNumber = reversedNumber \* 10 + remainder;

number /= 10;

}

// Check if the original number is equal to its reverse

if (originalNumber == reversedNumber) {

cout << "The number " << originalNumber << " is a palindrome." << endl;

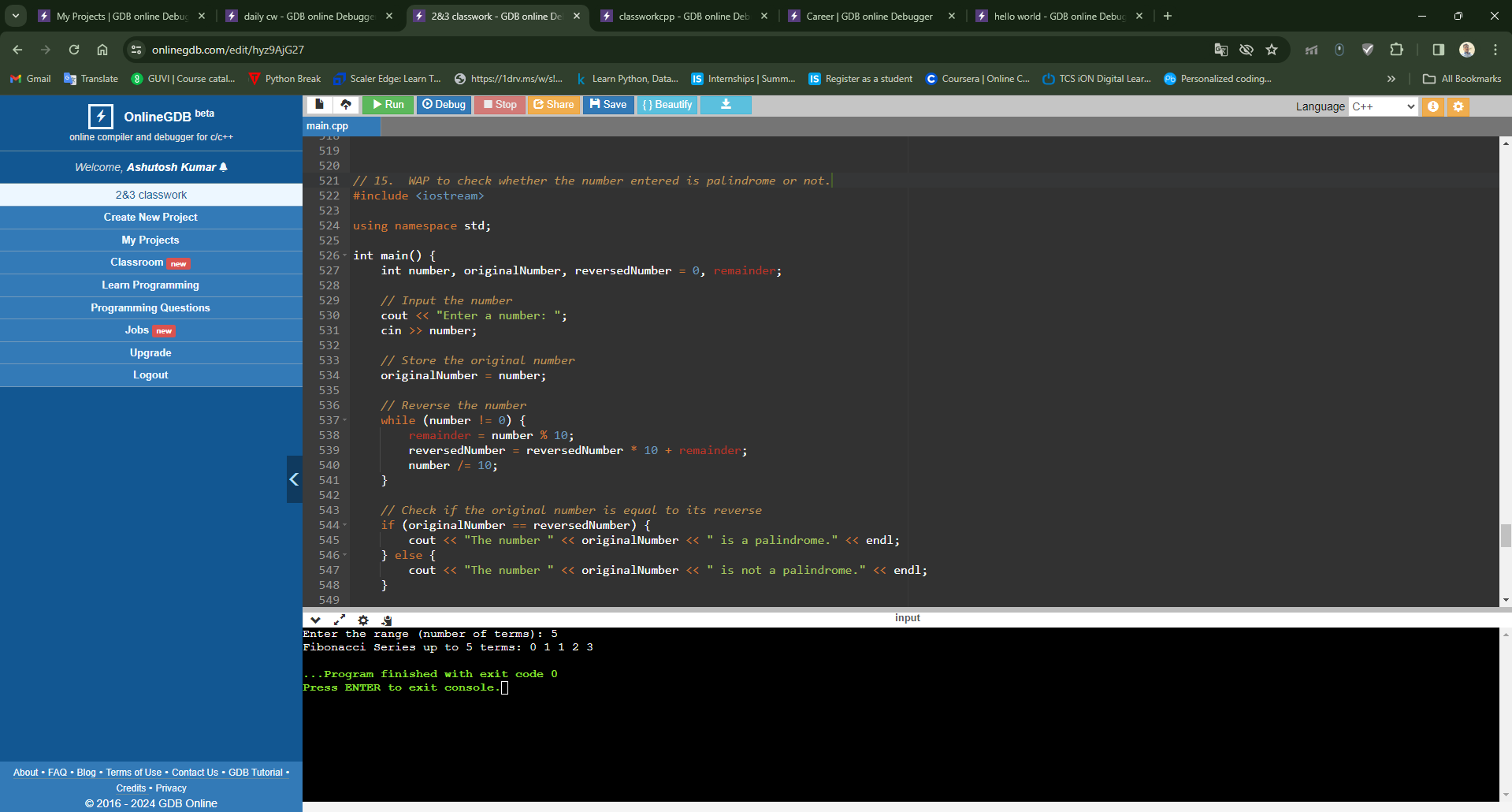
} else {

cout << "The number " << originalNumber << " is not a palindrome." << endl;

}

return 0;

}



**C++ ASSIGNMENT 1.3**

WAP to print following kind of patterns:

1. \*

\*\*

\*\*\*

\*\*\*\*

\*\*\*\*\*

#include <iostream>

using namespace std;

int main() {

int rows;

// Input the number of rows

cout << "Enter the number of rows: ";

cin >> rows;

// Outer loop for rows

for (int i = 1; i <= rows; ++i) {

// Inner loop for columns

for (int j = 1; j <= i; ++j) {

cout << "\*";

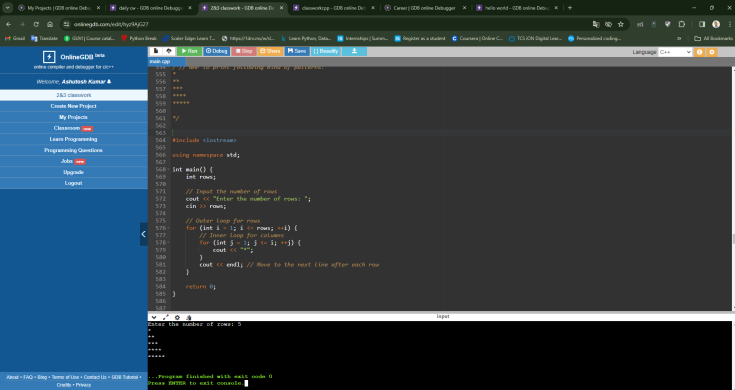
}

cout << endl; // Move to the next line after each row

}

return 0;

}



**Next Pdf Solution of Patterns**

1. \*

\*\*

\*\*\*

\*\*\*\*

\*\*\*\*\*\*

1. \*

\*\*\*

\*\*\*\*\*

\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*

1. \*\*\*\*\*\*\*\*\*

\*\*\*\* \*\*\*\*

\*\*\* \*\*\*

\*\* \*\*

\* \*

1. ABCD

ABC

AB

A

1. 1

12

123

1234

12345

123456

1. ABCDEDCBA

ABCD DCBA

ABC CBA

AB BA

A A