1**.Student details :**

#include <stdio.h>

struct student {

char name[50];

int roll\_number;

int marks[3];

};

float calculateAverage(struct student s) {

float sum = 0;

for (int i = 0; i < 3; i++) {

sum += s.marks[i];

}

return sum / 3.0;

}

int main() {

struct student s;

printf("Enter student details:\n");

printf("Name: ");

scanf("%s", s.name);

printf("Roll Number: ");

scanf("%d", &s.roll\_number);

printf("Enter marks in three subjects: ");

for (int i = 0; i < 3; i++) {

scanf("%d", &s.marks[i]);

}

printf("\nStudent Details:\n");

printf("Name: %s\n", s.name);

printf("Roll Number: %d\n", s.roll\_number);

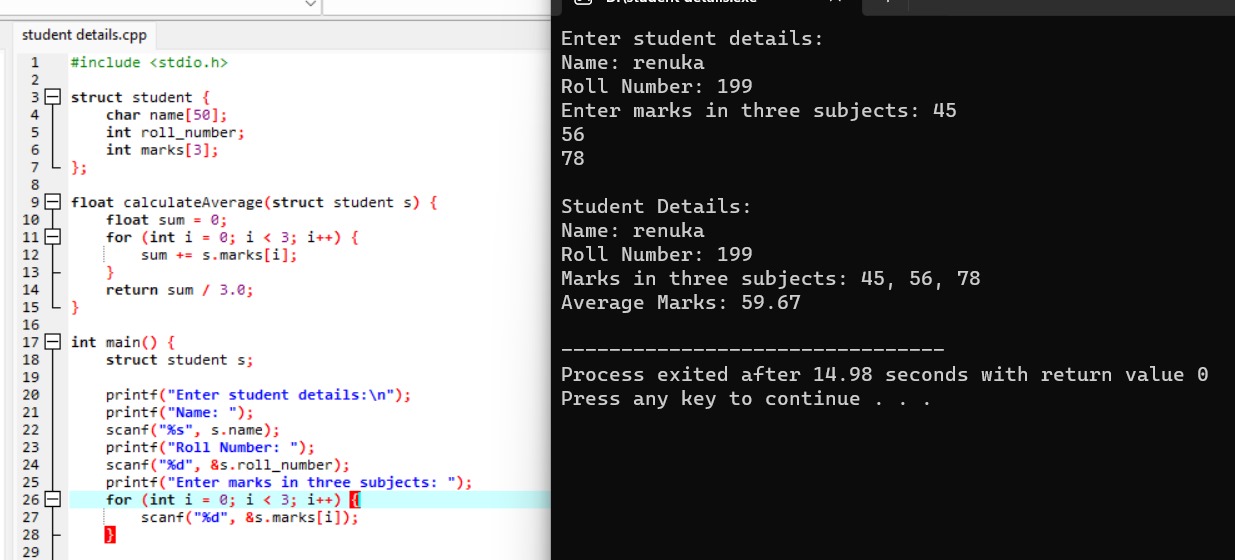
printf("Marks in three subjects: %d, %d, %d\n", s.marks[0], s.marks[1], s.marks[2]);

float average = calculateAverage(s);

printf("Average Marks: %.2f\n", average);

return 0;

}



2.**0’s and 1’s:**

#include <stdio.h>

#include <stdlib.h>

int max(int a, int b) {

return (a > b) ? a : b;

}

int findMaxLengthSubarray(int arr[], int n) {

int maxLength = 0;

int sum = 0;

int maxLenEndingIndex[n \* 2 + 1];

for (int i = 0; i < n \* 2 + 1; i++) {

maxLenEndingIndex[i] = -2;

}

maxLenEndingIndex[n] = -1;

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

sum += (arr[i] == 0) ? -1 : 1;

if (maxLenEndingIndex[sum + n] == -2) {

maxLenEndingIndex[sum + n] = i;

} else {

maxLength = max(maxLength, i - maxLenEndingIndex[sum + n]);

}

}

return maxLength;

}

int main() {

int arr[] = {0, 1, 0, 1, 1, 0, 0};

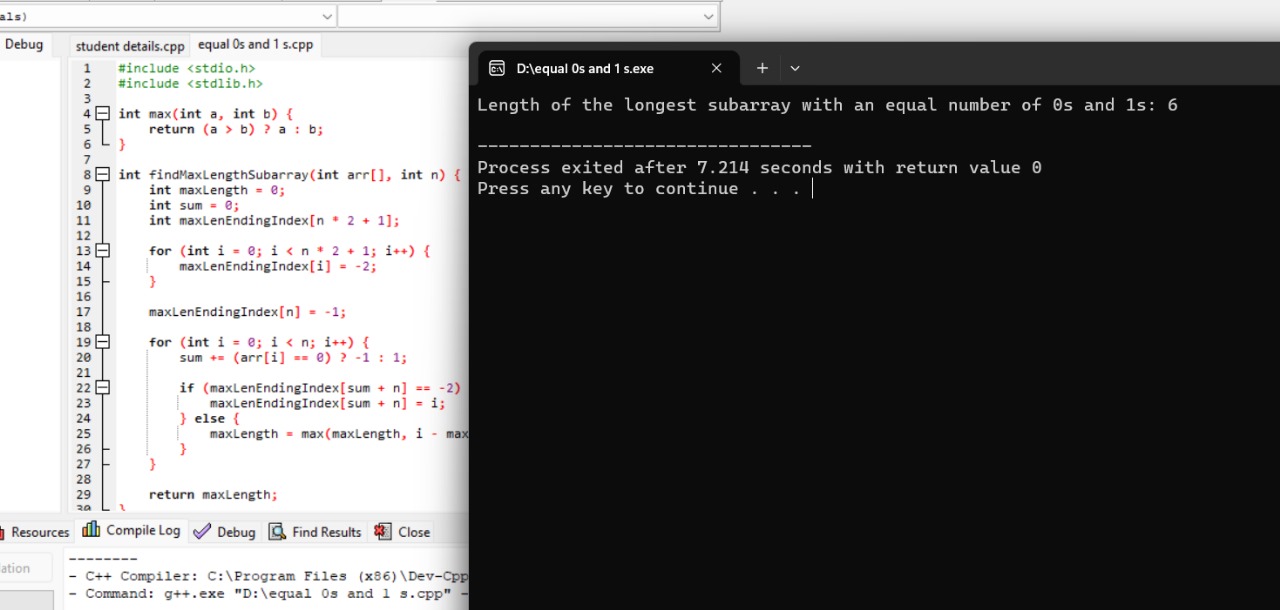
int n = sizeof(arr) / sizeof(arr[0]);

int maxLength = findMaxLengthSubarray(arr, n);

printf("Length of the longest subarray with an equal number of 0s and 1s: %d\n", maxLength);

return 0;

}



3.**Employee:**

#include <stdio.h>

struct Employee {

char name[50];

int employeeID;

float salary;

};

int main() {

struct Employee emp;

printf("Enter employee name: ");

scanf("%s", emp.name);

printf("Enter employee ID: ");

scanf("%d", &emp.employeeID);

printf("Enter employee salary: ");

scanf("%f", &emp.salary);

printf("\nEmployee Details:\n");

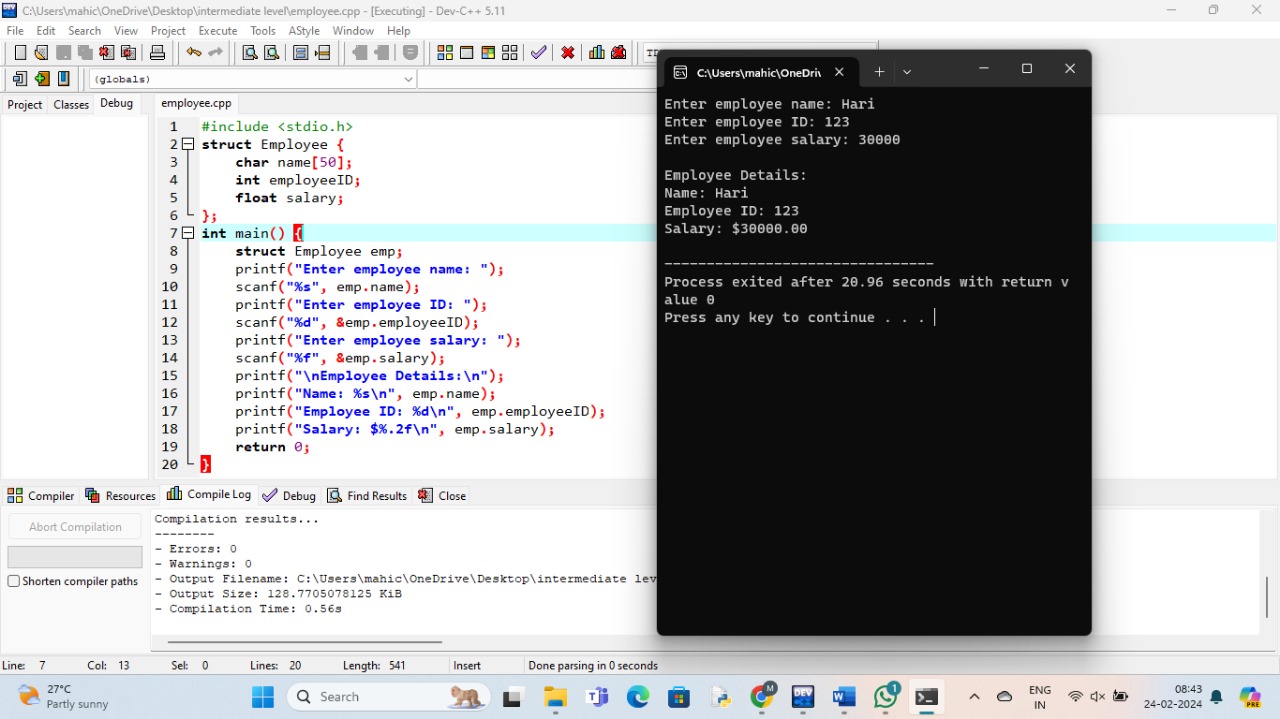
printf("Name: %s\n", emp.name);

printf("Employee ID: %d\n", emp.employeeID);

printf("Salary: $%.2f\n", emp.salary);

return 0;

}



4**.Date:**

#include <stdio.h>

struct Date {

int day;

int month;

int year;

};

int compareDates(struct Date date1, struct Date date2) {

if (date1.year < date2.year)

return -1;

else if (date1.year > date2.year)

return 1;

else {

if (date1.month < date2.month)

return -1;

else if (date1.month > date2.month)

return 1;

else {

if (date1.day < date2.day)

return -1;

else if (date1.day > date2.day)

return 1;

else

return 0;

}

}

}

int main() {

struct Date date1, date2;

printf("Enter first date (day month year): ");

scanf("%d %d %d", &date1.day, &date1.month, &date1.year);

printf("Enter second date (day month year): ");

scanf("%d %d %d", &date2.day, &date2.month, &date2.year);

int result = compareDates(date1, date2);

if (result < 0)

printf("First date comes first.\n");

else if (result > 0)

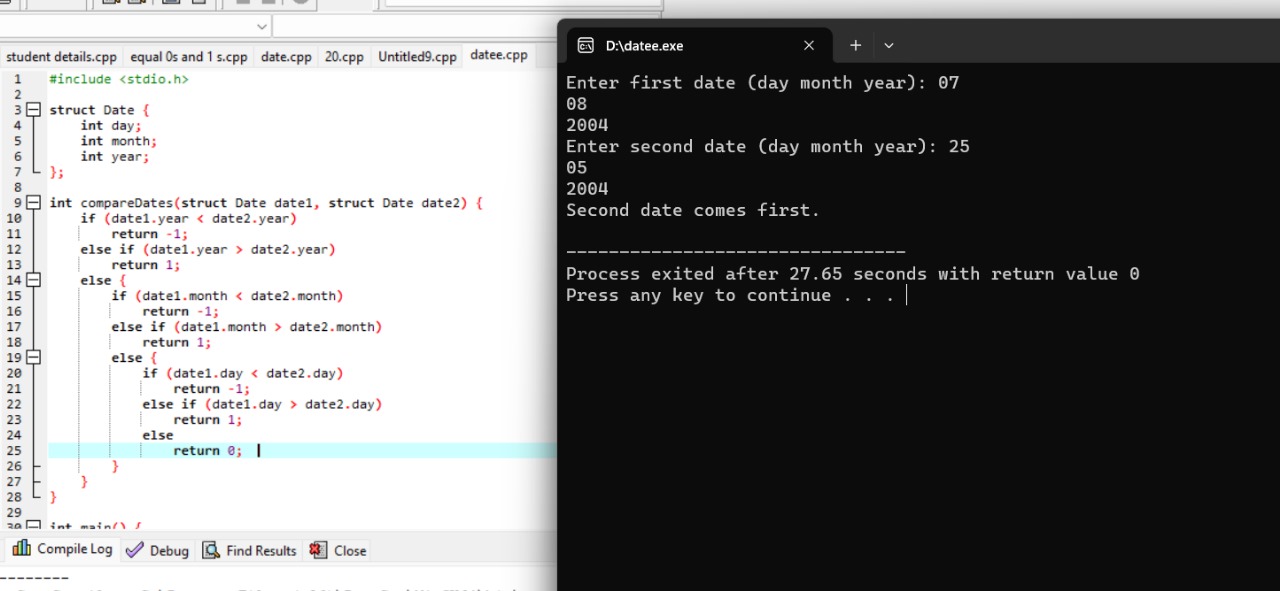
printf("Second date comes first.\n");

else

printf("Both dates are equal.\n");

return 0;

}



5.**Book:**

#include <stdio.h>

// Define the structure for Book

struct Book {

char title[100];

char author[100];

float price;

};

int main() {

// Initialize an array of books

struct Book books[3] = {

{"The Great Gatsby", "F. Scott Fitzgerald", 10.99},

{"To Kill a Mockingbird", "Harper Lee", 12.50},

{"1984", "George Orwell", 8.75}

};

// Display details of each book

printf("Book Details:\n");

for (int i = 0; i < 3; i++) {

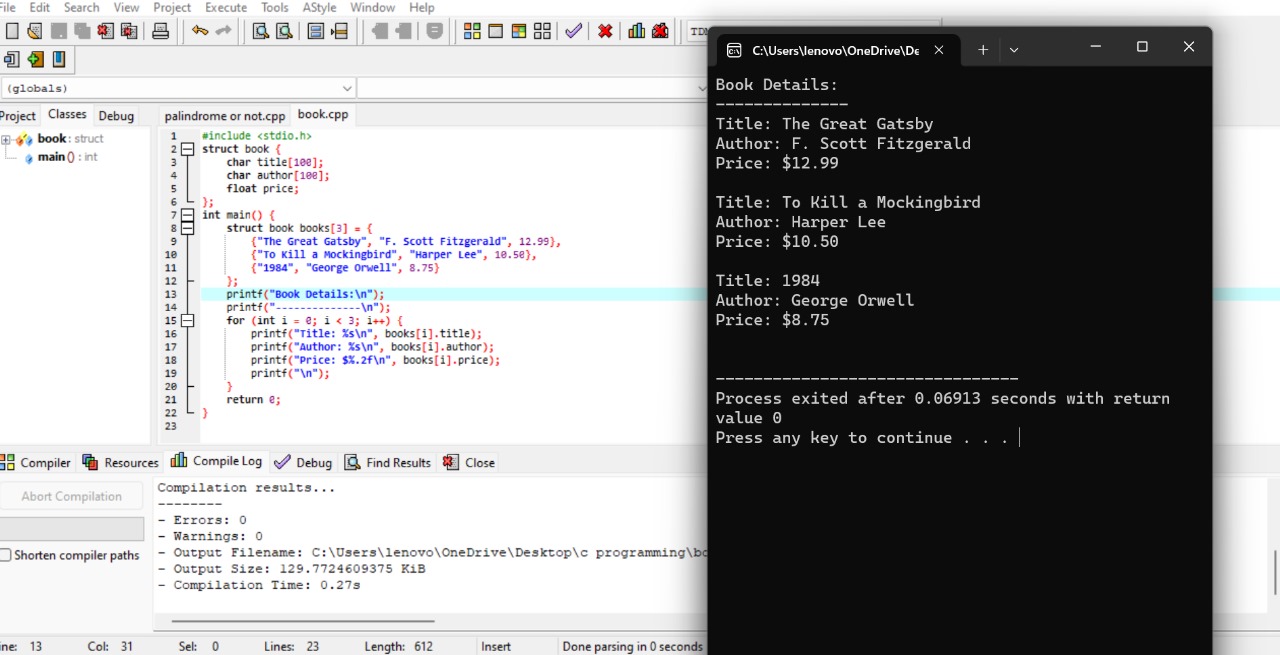
printf("Title: %s\n", books[i].title);

printf("Author: %s\n", books[i].author);

printf("Price: $%.2f\n\n", books[i].price);

}

return 0;

} 

6.**Distance between two points:**

#include <stdio.h>

#include <math.h>

// Structure to represent a point in a 2D coordinate system

struct Point {

double x;

double y;

};

// Function to calculate the distance between two points

double calculateDistance(struct Point p1, struct Point p2) {

double distance = sqrt((p2.x - p1.x) \* (p2.x - p1.x) + (p2.y - p1.y) \* (p2.y - p1.y));

return distance;

}

int main() {

struct Point point1 = {4.0, 3.0}; // Coordinates of point 1

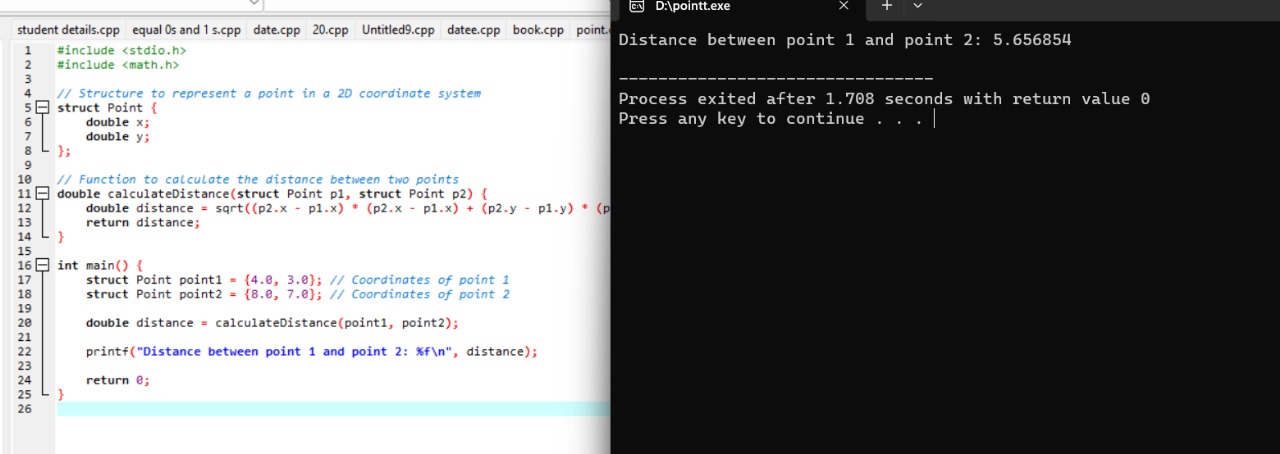
struct Point point2 = {8.0, 7.0}; // Coordinates of point 2

double distance = calculateDistance(point1, point2);

printf("Distance between point 1 and point 2: %f\n", distance);

return 0;

}



7.**Rearrange array:**

#include <stdio.h>

void rearrange(int arr[], int size) {

int j = 0;

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

if (arr[i] < 0) {

if (i != j) {

int temp = arr[i];

arr[i] = arr[j];

arr[j] = temp;

}

j++;

}

}

}

int main() {

int arr[] = {4, -3, 2, 1, -5, 0, -7, 9, -2};

int size = sizeof(arr) / sizeof(arr[0]);

int i; // Declare i before the loop

rearrange(arr, size);

printf("Rearranged Array: ");

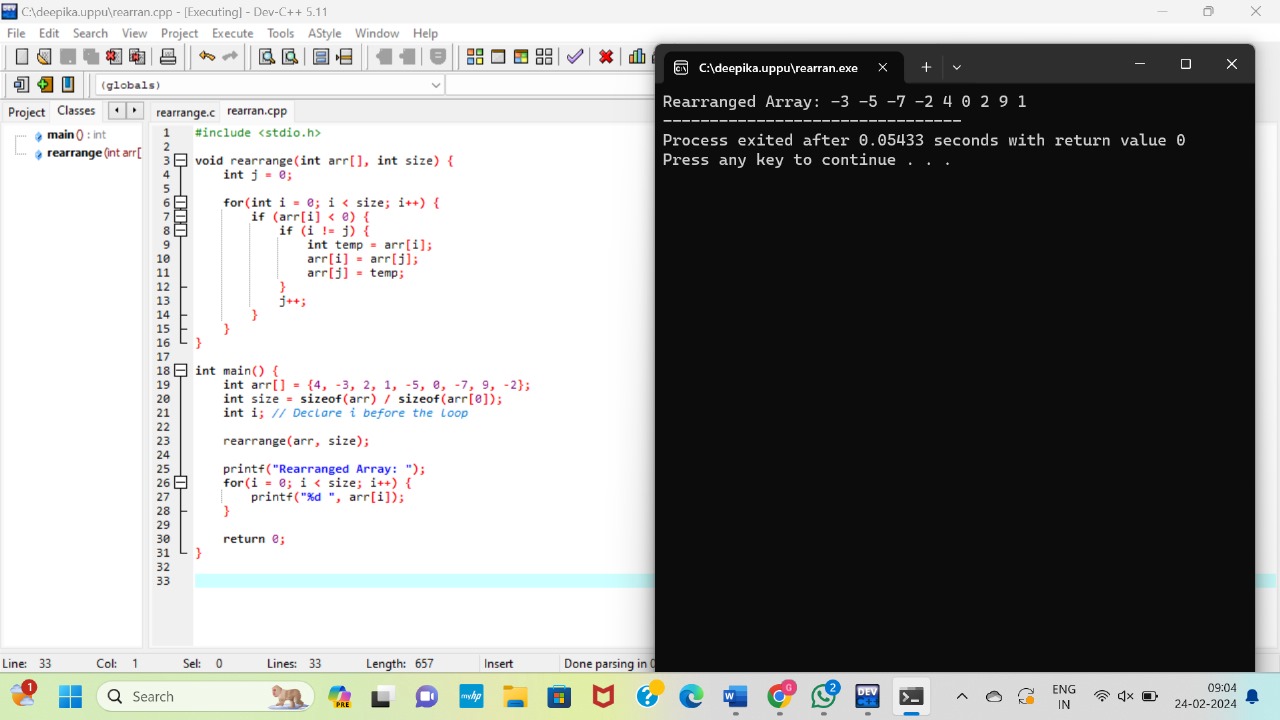
for(i = 0; i < size; i++) {

printf("%d ", arr[i]);

}

return 0;

}



8.**Recursive B to D:**

#include <stdio.h>

void decimalToBinary(int n) {

if (n > 1) {

decimalToBinary(n / 2);

}

printf("%d", n % 2);

}

int main() {

int decimalNumber;

printf("Enter a decimal number: ");

scanf("%d", &decimalNumber);

if (decimalNumber < 0) {

printf("Decimal number should be non-negative.\n");

return 1;

}

printf("Binary representation: ");

if (decimalNumber == 0) {

printf("0");

} else {

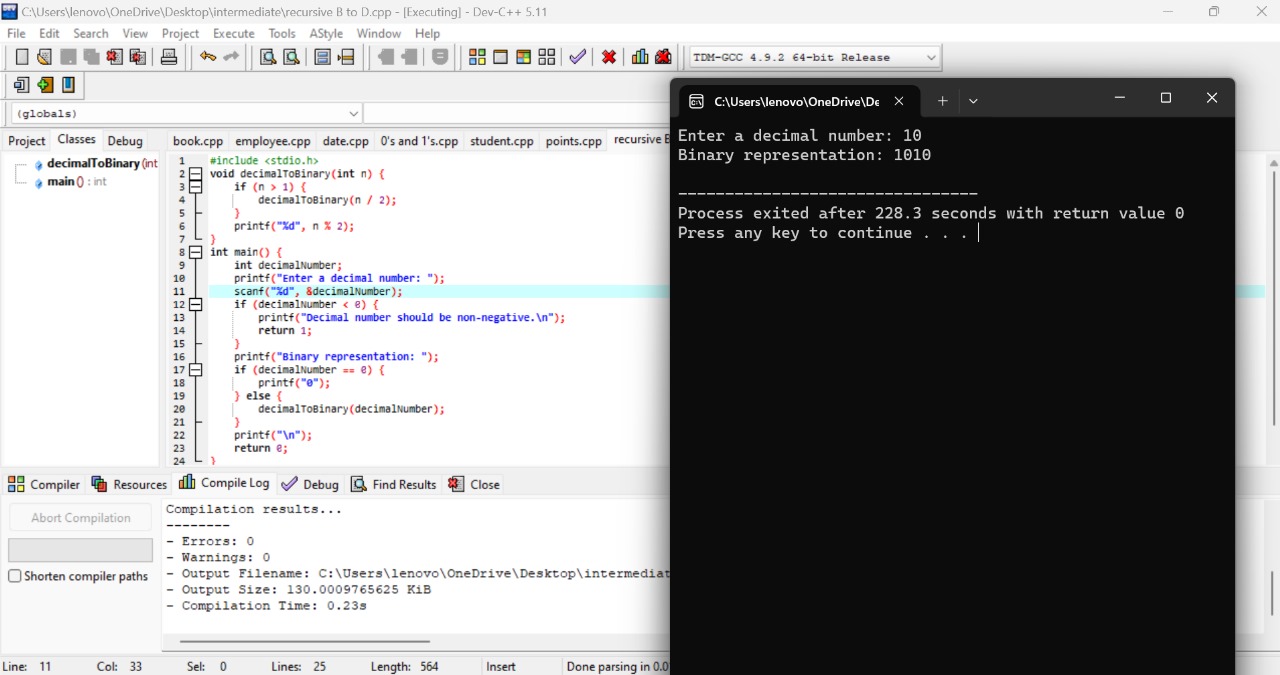
decimalToBinary(decimalNumber);

}

printf("\n");

return 0;

}



9.**Recursive permutation:**

#include <stdio.h>

#include <string.h>

void swap(char \*x, char \*y) {

char temp = \*x;

\*x = \*y;

\*y = temp;

}

void permute(char \*str, int left, int right) {

if (left == right) {

printf("%s\n", str);

} else {

for (int i = left; i <= right; i++) {

swap((str + left), (str + i));

permute(str, left + 1, right);

swap((str + left), (str + i));

}

}

}

int main() {

char str[100];

printf("Enter a string: ");

scanf("%s", str);

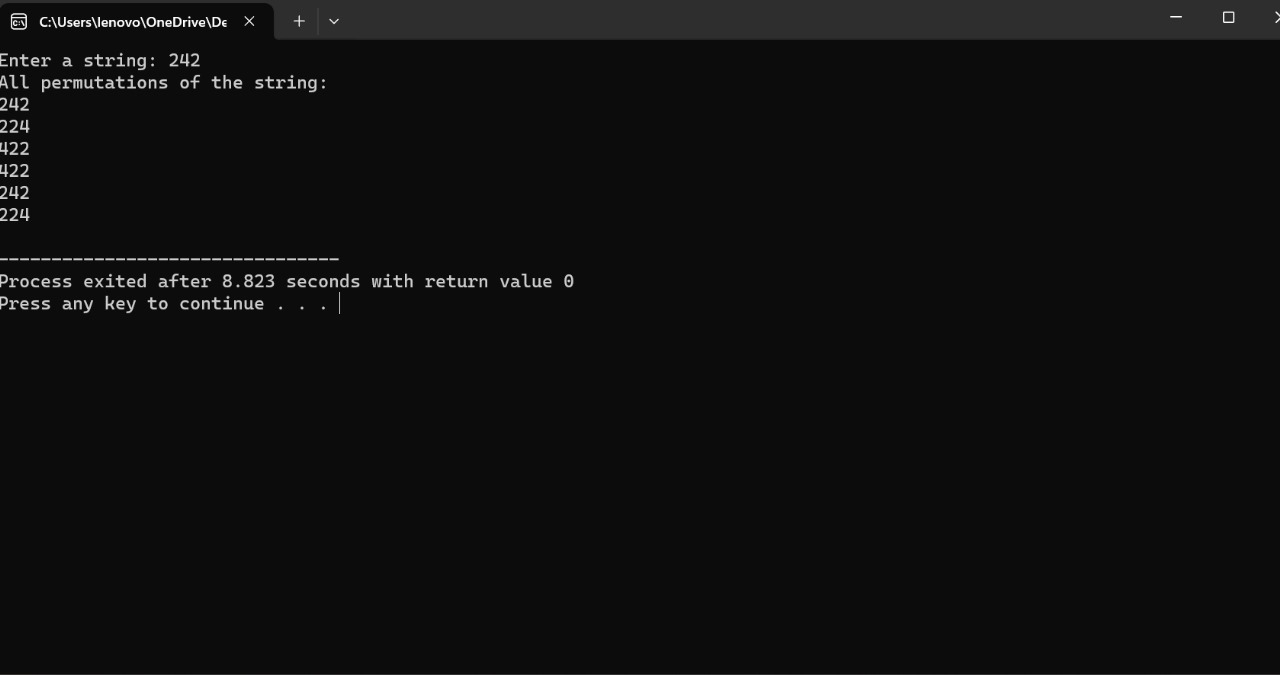
int n = strlen(str);

printf("All permutations of the string:\n");

permute(str, 0, n - 1);

return 0;

}



10.**Find min max:**

#include <stdio.h>

void findMaxMin(int arr[], int length, int \*max, int \*min) {

\*max = \*min = arr[0];

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

if (arr[i] > \*max) {

\*max = arr[i];

} else if (arr[i] < \*min) {

\*min = arr[i];

}

}

}

int main() {

int arr[] = {5, 12, -3, 8, 2, 10};

int length = sizeof(arr) / sizeof(arr[0]);

int max, min;

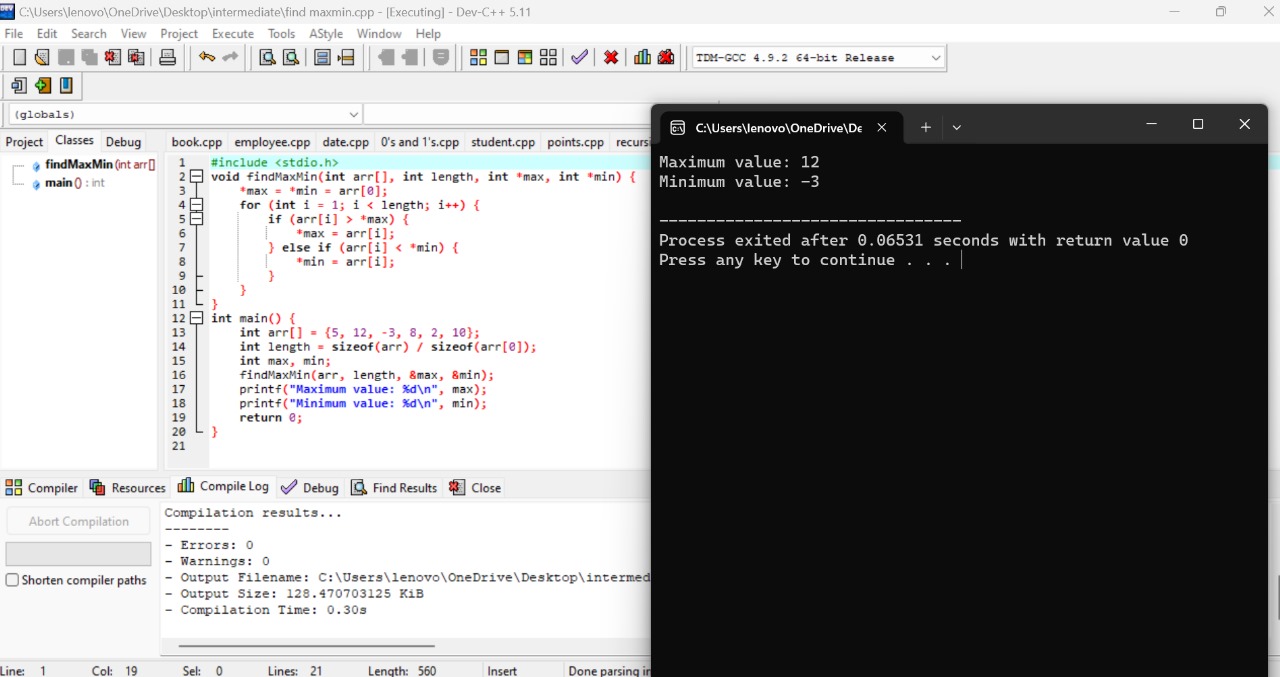
findMaxMin(arr, length, &max, &min);

printf("Maximum value: %d\n", max);

printf("Minimum value: %d\n", min);

return 0;

}



**Schedule problems:**

1.**Write a C program to implement a function that calculates the area of a triangle given its three sides.**

#include<stdio.h>

#include <math.h>

double areaOfTriangle(double a, double b, double c) {

double s = (a + b + c) / 2;

double area = sqrt(s \* (s - a) \* (s - b) \* (s - c));

return area;

}

int main() {

double side1, side2, side3;

printf("Enter the length of side 1: ");

scanf("%lf", &side1);

printf("Enter the length of side 2: ");

scanf("%lf", &side2);

printf("Enter the length of side 3: ");

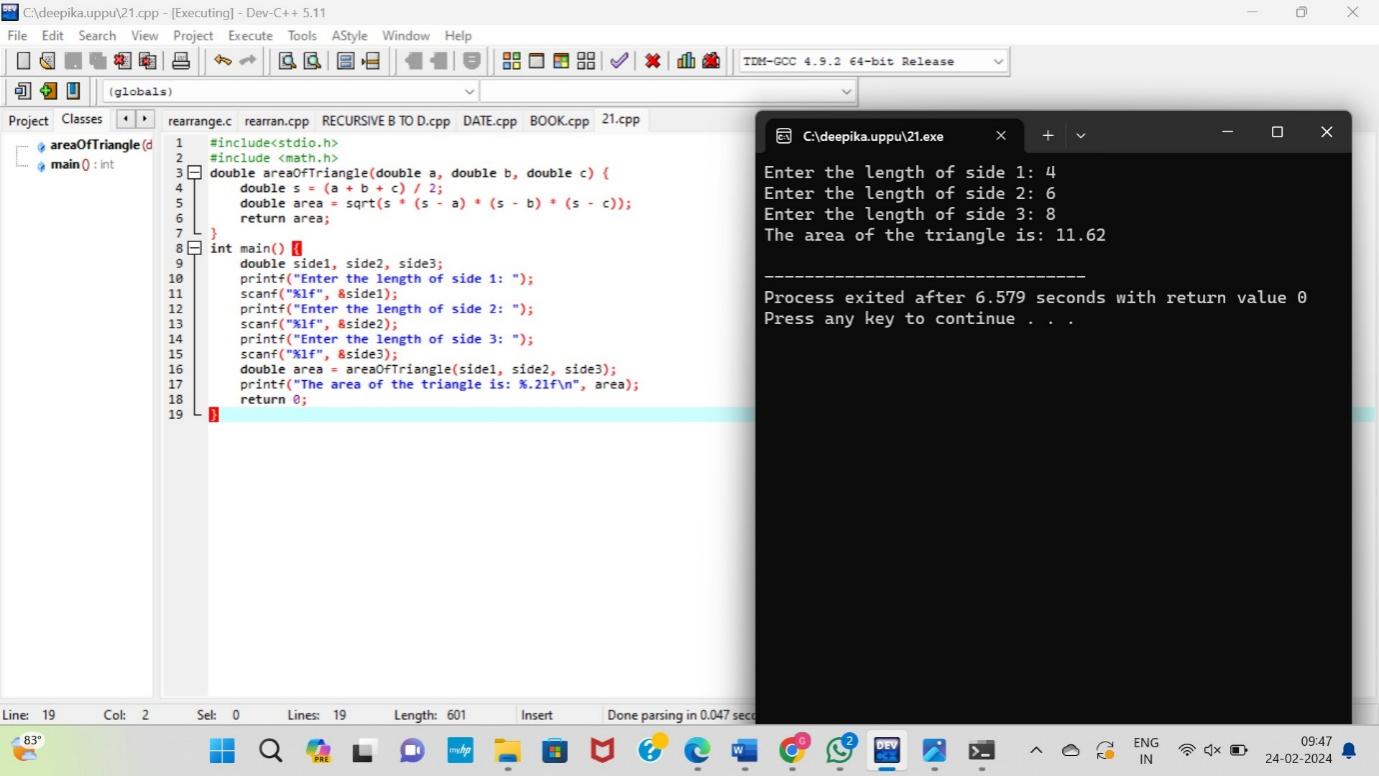
scanf("%lf", &side3);

double area = areaOfTriangle(side1, side2, side3);

printf("The area of the triangle is: %.2lf\n", area);

return 0;

}



2. **Write a C program to implement a function that checks whether a given year is a leap year or not.**

#include <stdio.h>

#include <string.h>

#include <limits.h>

#define MAX\_CHARS 256

int allCharactersFound(int arr[]) {

for (int i = 0; i < MAX\_CHARS; i++) {

if (arr[i] > 0) {

return 0;

}

}

return 1;

}

char\* minWindowSubstring(char\* str, char\* pattern) {

int patternCount[MAX\_CHARS] = {0};

int strCount[MAX\_CHARS] = {0};

int patternLength = strlen(pattern);

int strLength = strlen(str);

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

patternCount[pattern[i]]++;

}

int minWindowSize = INT\_MAX;

char\* minWindowStart = NULL;

int left = 0, right = 0;

int count = 0;

int startPos = -1;

while (right < strLength) {

if (patternCount[str[right]] > 0) {

strCount[str[right]]++;

if (strCount[str[right]] <= patternCount[str[right]]) {

count++;

}

}

if (count == patternLength) {

while (patternCount[str[left]] == 0 || strCount[str[left]] > patternCount[str[left]]) {

if (strCount[str[left]] > patternCount[str[left]]) {

strCount[str[left]]--;

}

left++;

}

if (right - left + 1 < minWindowSize) {

minWindowSize = right - left + 1;

minWindowStart = &str[left];

startPos = left;

}

}

right++;

}

if (startPos != -1) {

minWindowStart[minWindowSize] = '\0';

}

return minWindowStart;

}

int main() {

char str[] = "ADOBECODEBANC";

char pattern[] = "ABC";

char\* minWindow = minWindowSubstring(str, pattern);

if (minWindow != NULL) {

printf("Minimum window substring containing all characters of pattern: %s\n", minWindow);

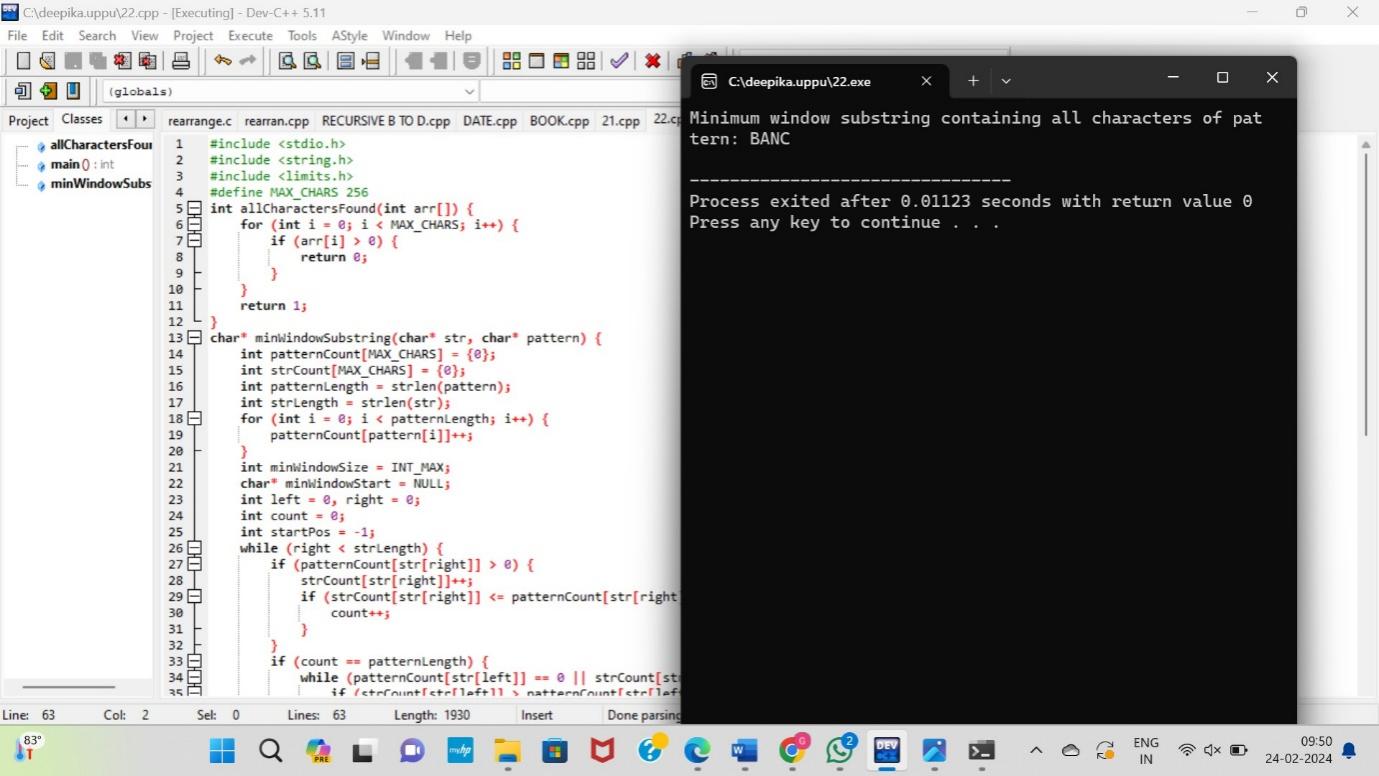
} else {

printf("No window found\n");

}

return 0;

}



**3. Write a C program to implement a function that converts a temperature from Celsius to Fahrenheit and vice versa.**

#include <stdio.h>

double celsiusToFahrenheit(double celsius) {

return (celsius \* 9 / 5) + 32;

}

int main() {

double celsius, fahrenheit;

printf("Enter temperature in Celsius: ");

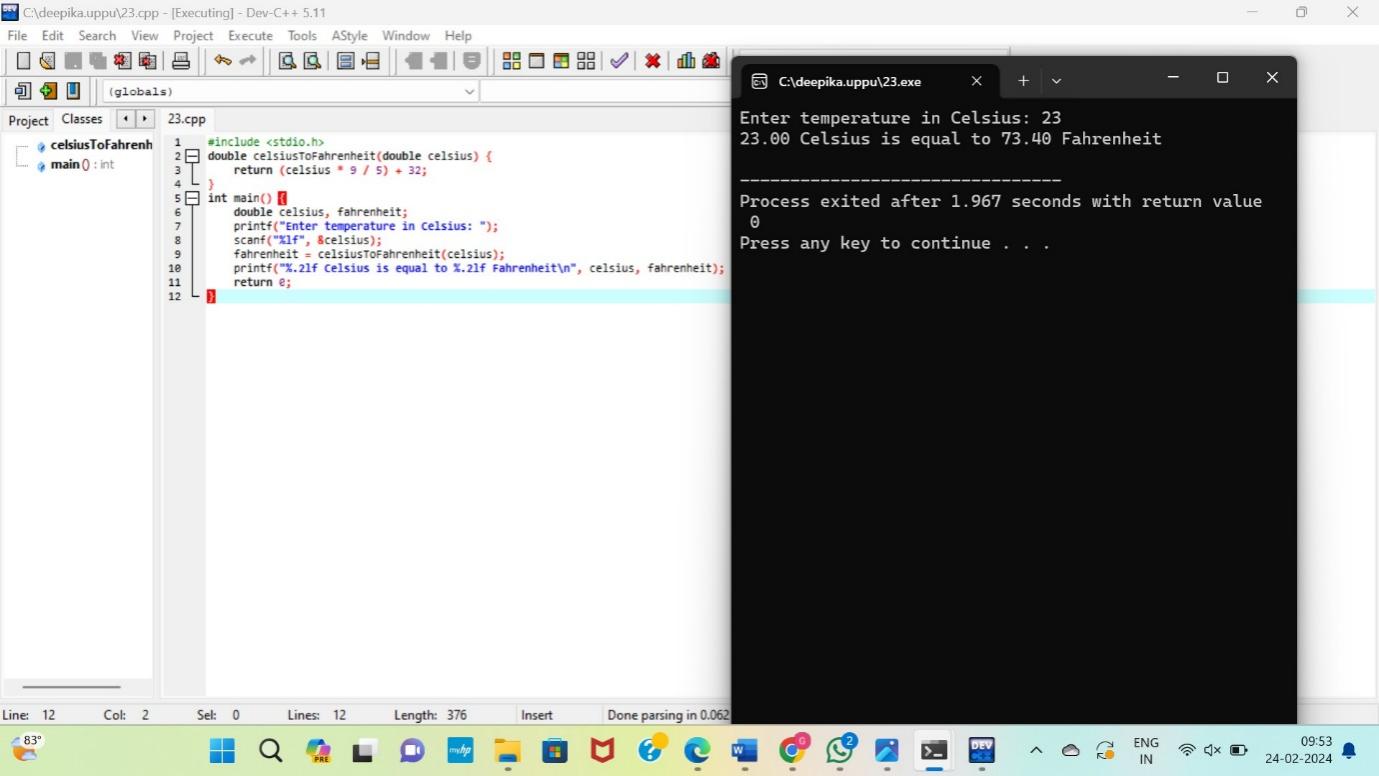
scanf("%lf", &celsius);

fahrenheit = celsiusToFahrenheit(celsius);

printf("%.2lf Celsius is equal to %.2lf Fahrenheit\n", celsius, fahrenheit);

return 0;

}



**4. Write a C program to implement a function that calculates the factorial of a given number using recursion.**

#include <stdio.h>

int factorial(int n) {

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

return 1;

} else {

return n \* factorial(n - 1);

}

}

int main() {

int num;

printf("Enter a number: ");

scanf("%d", &num);

if (num < 0) {

printf("Factorial is not defined for negative numbers.\n");

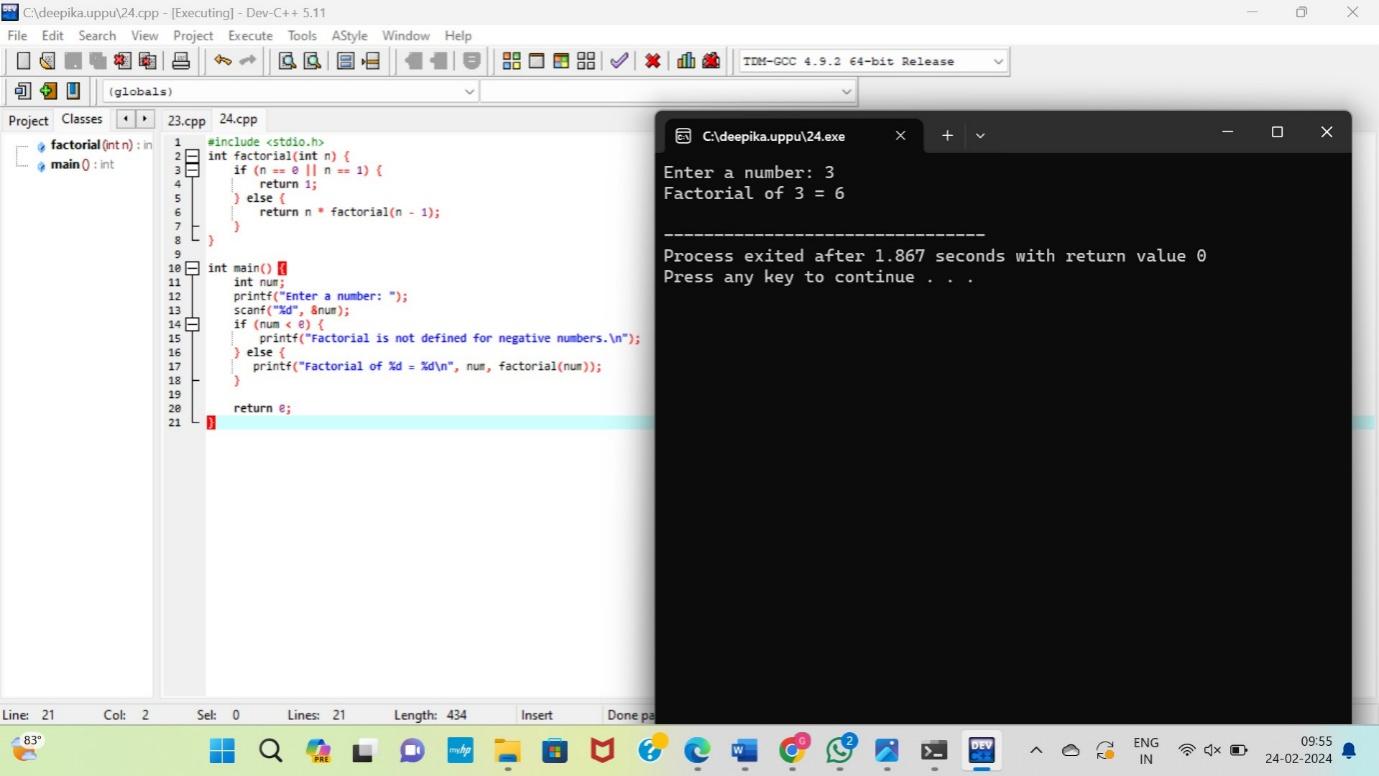
} else {

printf("Factorial of %d = %d\n", num, factorial(num));

}

return 0;

}



**5. Write a C program to implement a function that finds the sum of the first n natural numbers.**

#include <stdio.h>

int sumOfNaturals(int n) {

int sum = 0;

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

sum += i;

}

return sum;

}

int main() {

int n;

printf("Enter a positive integer n: ");

scanf("%d", &n);

if (n <= 0) {

printf("Please enter a positive integer.\n");

return 1;

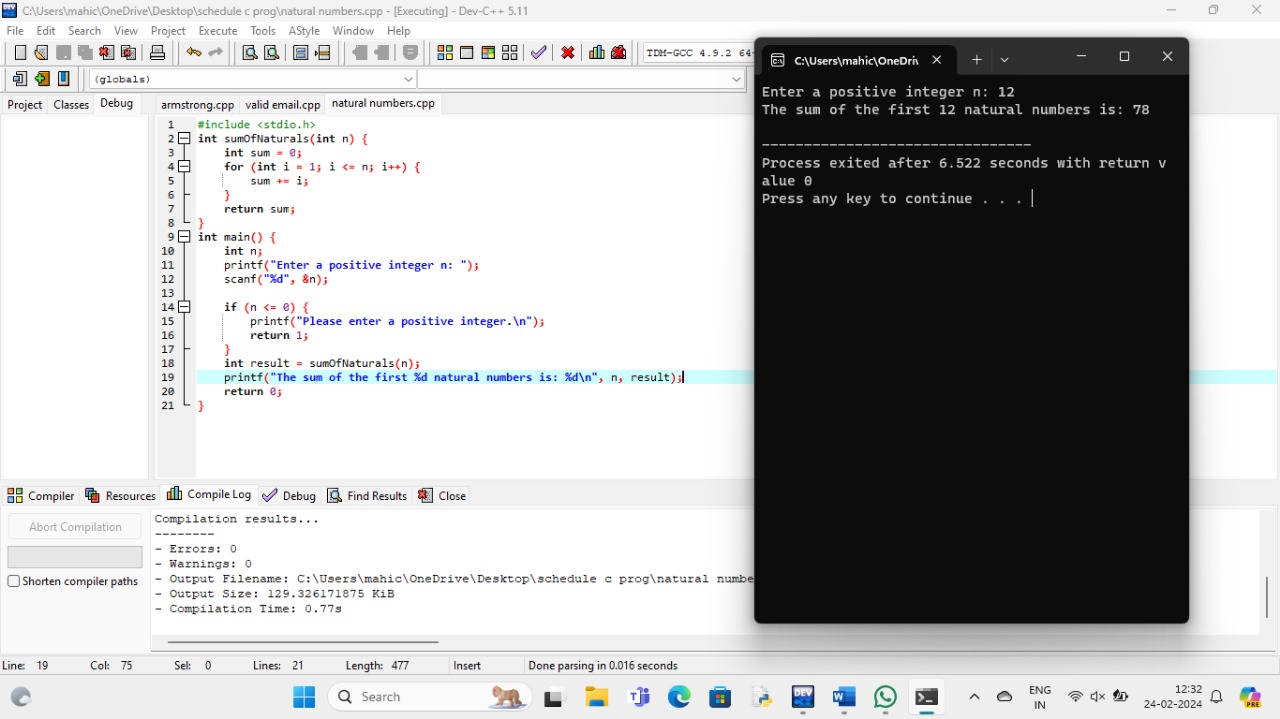
}

int result = sumOfNaturals(n);

printf("The sum of the first %d natural numbers is: %d\n", n, result);

return 0;

}



1. **Write a C program to implement a function that checks whether a given number is Armstrong number or not**

#include <stdio.h>

void findMaxMin(int arr[], int length, int \*max, int \*min) {

\*max = \*min = arr[0];

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

if (arr[i] > \*max) {

\*max = arr[i];

} else if (arr[i] < \*min) {

\*min = arr[i];

}

}

}

int main() {

int arr[] = {5, 12, -3, 8, 2, 10};

int length = sizeof(arr) / sizeof(arr[0]);

int max, min;

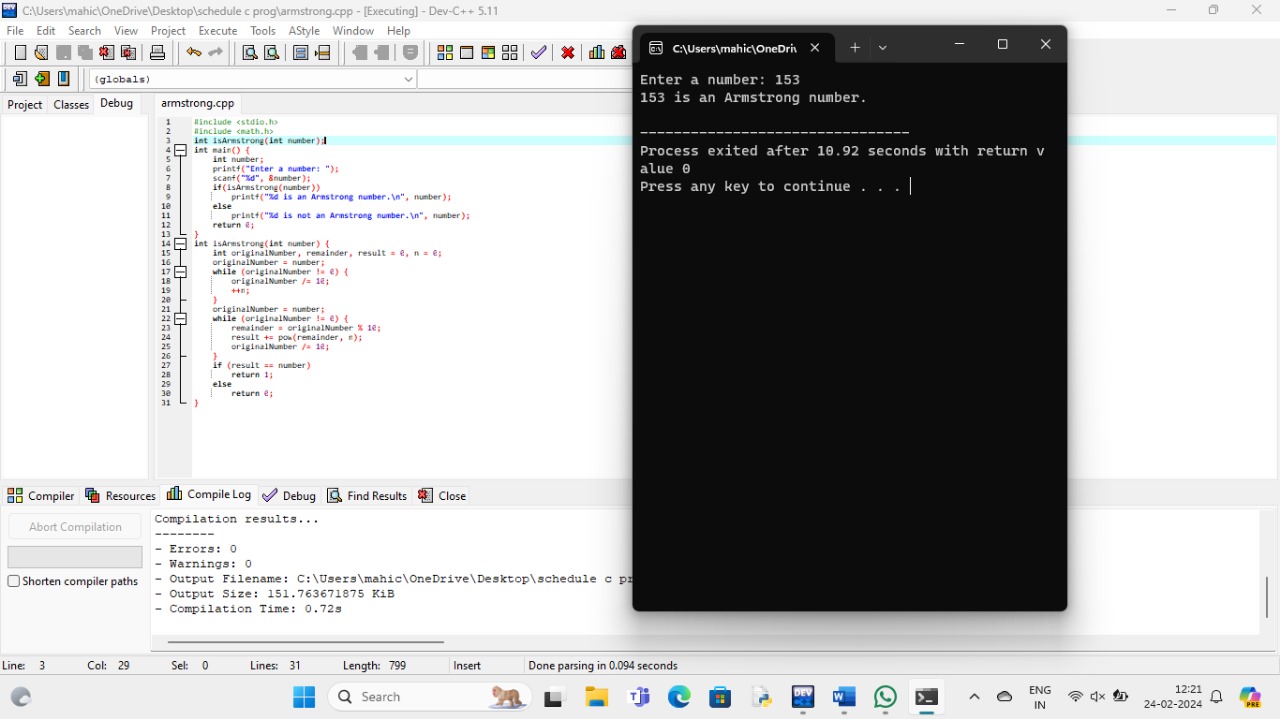
findMaxMin(arr, length, &max, &min);

printf("Maximum value: %d\n", max);

printf("Minimum value: %d\n", min);

return 0;

}



**7. Write a C program to implement a function that finds the roots of a quadratic equation ax^2+bx+c=0**.

#include <stdio.h>

#include <math.h>

void findRoots(float a, float b, float c) {

float discriminant, realPart, imaginaryPart;

discriminant = b \* b - 4 \* a \* c;

if (discriminant > 0) {

float root1 = (-b + sqrt(discriminant)) / (2 \* a);

float root2 = (-b - sqrt(discriminant)) / (2 \* a);

printf("Root1 = %.2f and Root2 = %.2f\n", root1, root2);

} else if (discriminant == 0) {

float root = -b / (2 \* a);

printf("Both roots are real and Equal. Root = %.2f\n", root);

} else {

realPart = -b / (2 \* a);

imaginaryPart = sqrt(-discriminant) / (2 \* a);

printf("Both roots are complex and different.\n");

printf("Root1 = %.2f + %.2fi and Root2 = %.2f - %.2fi\n", realPart, imaginaryPart, realPart, imaginaryPart);

}

}

int main() {

float a, b, c;

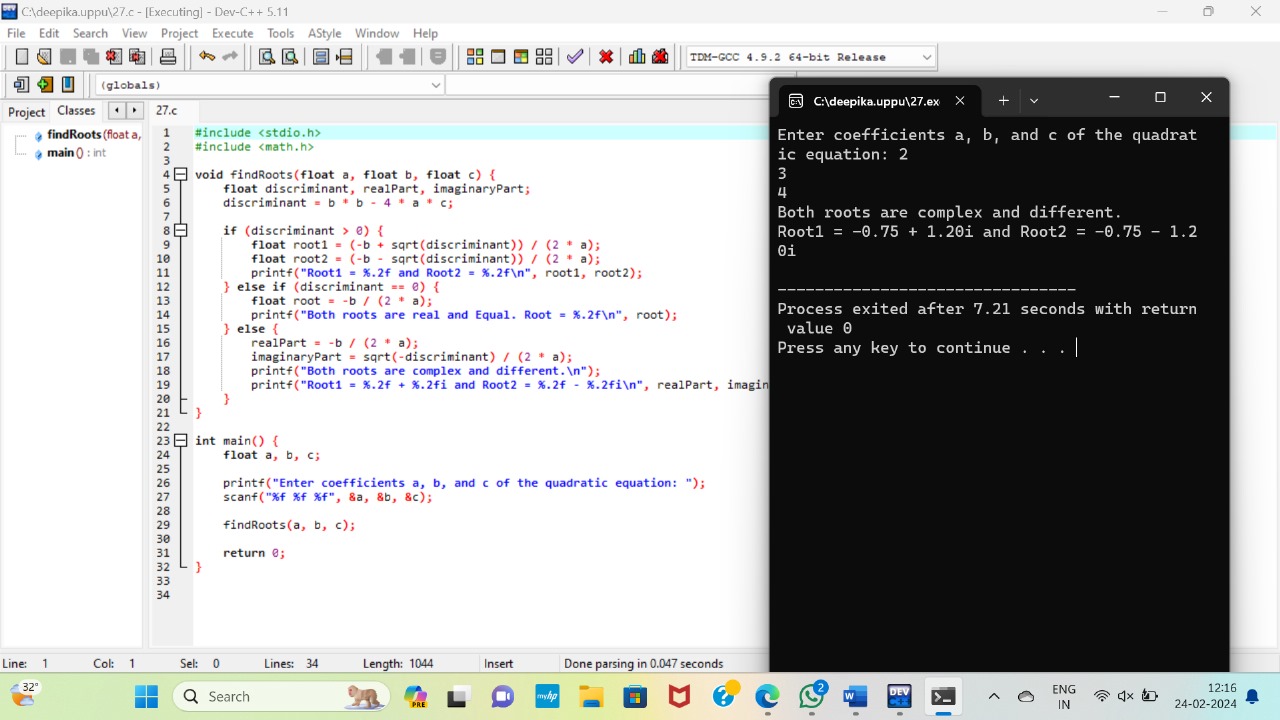
printf("Enter coefficients a, b, and c of the quadratic equation: ");

scanf("%f %f %f", &a, &b, &c);

findRoots(a, b, c);

return 0;

}



**8. Write a C program to implement a function that checks whether a given string is a valid email address or not**

#include <stdio.h>

#include <stdbool.h>

#include <string.h>

#include <ctype.h>

bool isValidEmail(const char \*email) {

int length = strlen(email);

bool atFound = false;

bool dotFound = false;

int atPosition = -1;

// Check for '@' and '.'

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

if (email[i] == '@') {

if (atFound || i == 0 || i == length - 1) {

return false; // More than one '@' or '@' at start or end

}

atFound = true;

atPosition = i;

} else if (email[i] == '.') {

if (i == 0 || i == length - 1 || email[i - 1] == '.') {

return false; // Dot at start, end, or consecutive dots

}

dotFound = true;

} else if (!(isalpha(email[i]) || isdigit(email[i]) || email[i] == '.' || email[i] == '-' || email[i] == '\_')) {

return false; // Invalid character

}

}

// Check if '@' and '.' are appropriately positioned

if (!atFound || !dotFound || atPosition > length - 5) {

return false;

}

return true;

}

int main() {

char email[100];

printf("Enter an email address: ");

scanf("%s", email);

if (isValidEmail(email)) {

printf("Valid email address\n");

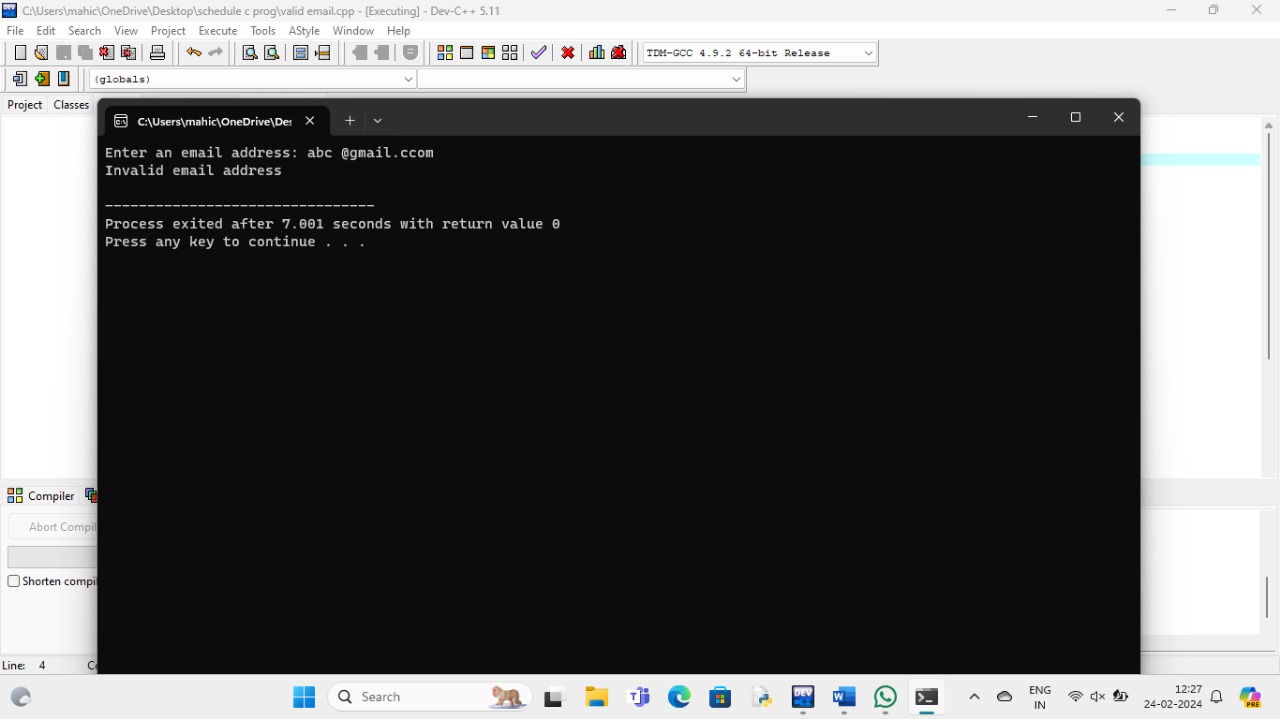
} else {

printf("Invalid email address\n");

}

return 0;

}



**9. Write a C program to implement a function that calculates the sum of digits of a given number until it becomes a single digit.**

include <stdio.h>

int sumOfDigitsUntilSingleDigit(int num) {

while (num > 9) {

int sum = 0;

while (num != 0) {

sum += num % 10;

num /= 10;

}

num = sum;

}

return num;

}

int main() {

int inputNum;

printf("Enter a number: ");

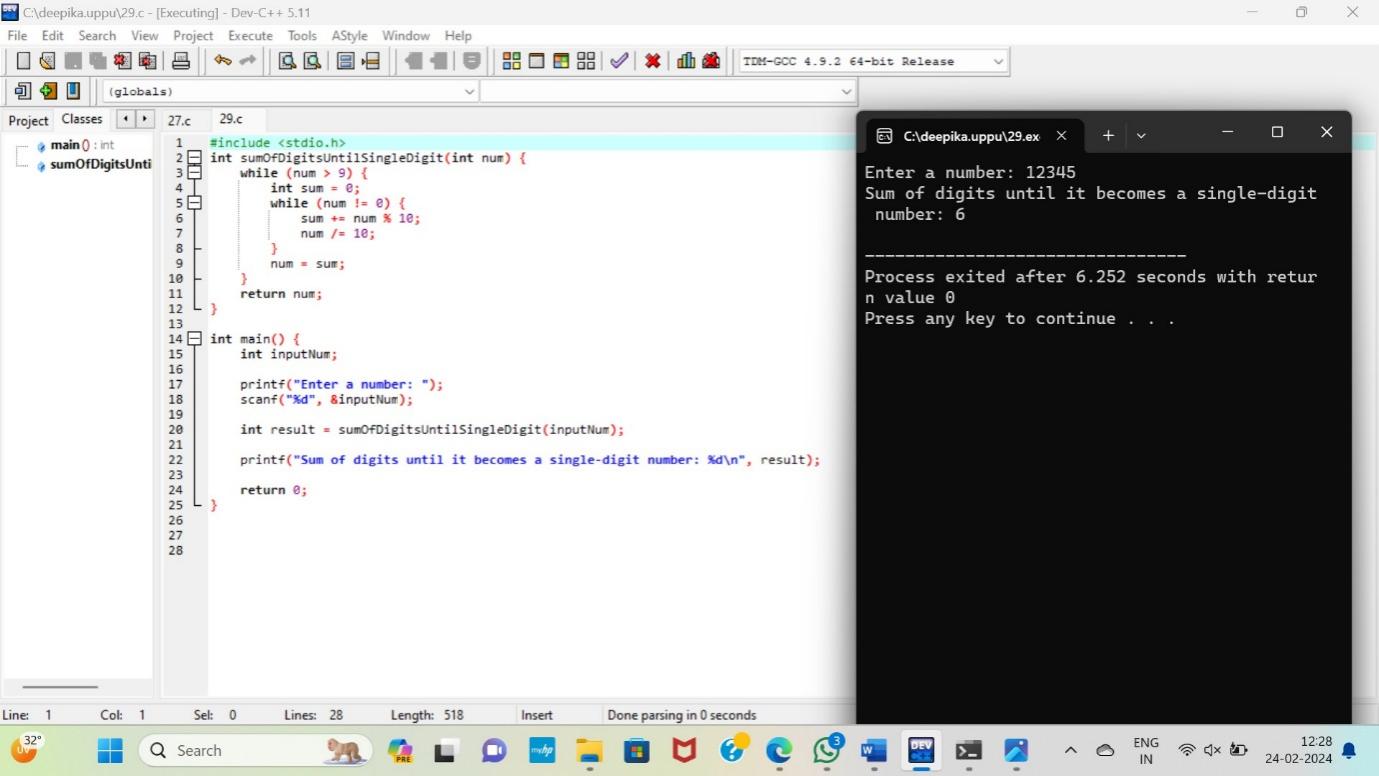
scanf("%d", &inputNum);

int result = sumOfDigitsUntilSingleDigit(inputNum);

printf("Sum of digits until it becomes a single-digit number: %d\n", result);

return 0;

}



**10. Write a C program to implement a function that checks whether a given number is a palindrome or not without converting it to a string.**

#include <stdio.h>

#include <stdbool.h>

bool isPalindrome(int num) {

int originalNum = num;

int reverse = 0;

while (num > 0) {

int digit = num % 10;

reverse = reverse \* 10 + digit;

num /= 10;

}

return originalNum == reverse;

}

int main() {

int inputNum;

printf("Enter a number: ");

scanf("%d", &inputNum);

if (isPalindrome(inputNum)) {

printf("%d is a palindrome number\n", inputNum);

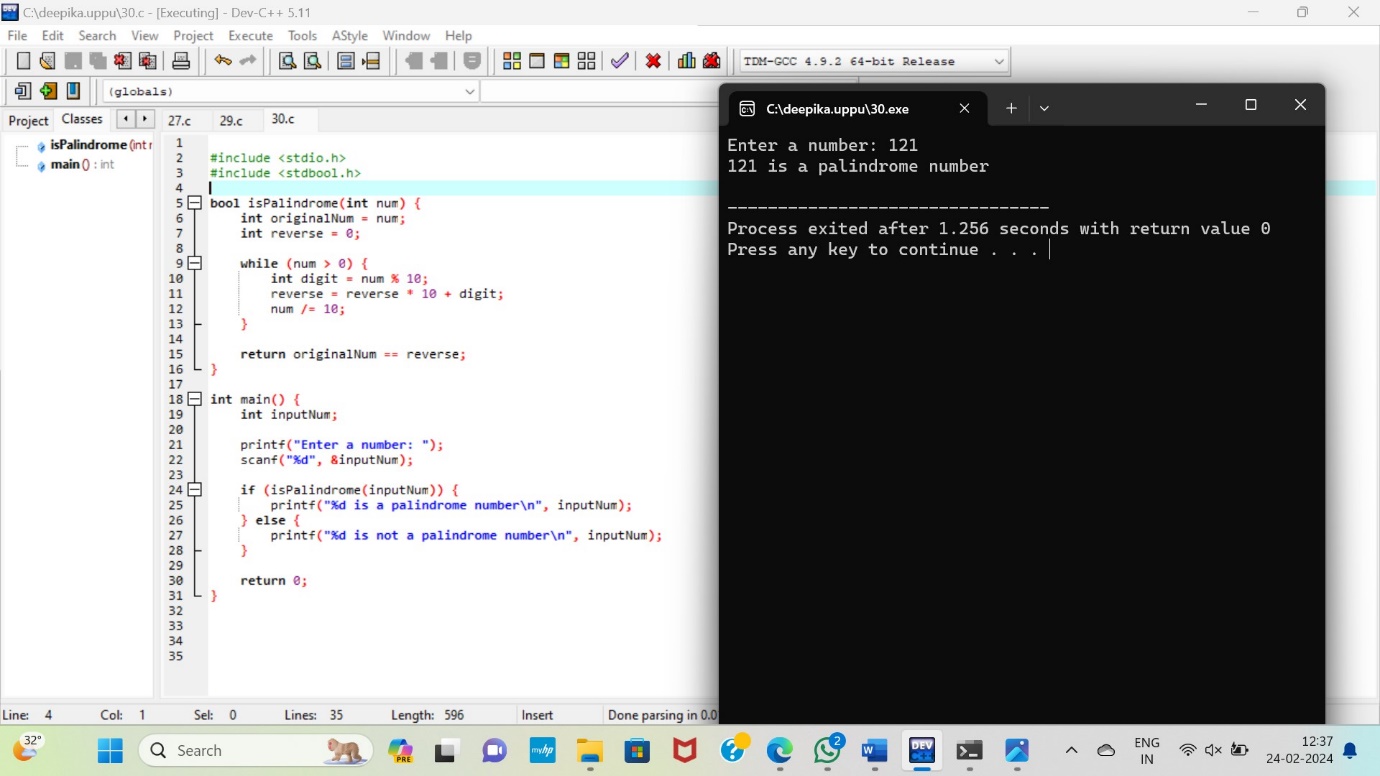
} else {

printf("%d is not a palindrome number\n", inputNum);

}

return 0;

}



1. **Write a C program to implement a function that calculates the sum of two matrices of dimensions m x n .**

#include <stdio.h>

#define MAX\_ROWS 10

#define MAX\_COLS 10

void addMatrices(int m, int n, int matrix1[MAX\_ROWS][MAX\_COLS], int matrix2[MAX\_ROWS][MAX\_COLS], int result[MAX\_ROWS][MAX\_COLS]) {

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

for (int j = 0; j < n; j++) {

result[i][j] = matrix1[i][j] + matrix2[i][j];

}

}

}

void displayMatrix(int m, int n, int matrix[MAX\_ROWS][MAX\_COLS]) {

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

for (int j = 0; j < n; j++) {

printf("%d\t", matrix[i][j]);

}

printf("\n");

}

}

int main() {

int m, n;

printf("Enter the number of rows and columns for the matrices (max %d x %d): ", MAX\_ROWS, MAX\_COLS);

scanf("%d %d", &m, &n);

if (m > MAX\_ROWS || n > MAX\_COLS) {

printf("Error: Dimensions exceed maximum size.\n");

return 1;

}

int matrix1[MAX\_ROWS][MAX\_COLS], matrix2[MAX\_ROWS][MAX\_COLS], result[MAX\_ROWS][MAX\_COLS];

printf("Enter elements of the first matrix:\n");

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

for (int j = 0; j < n; j++) {

scanf("%d", &matrix1[i][j]);

}

}

printf("Enter elements of the second matrix:\n");

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

for (int j = 0; j < n; j++) {

scanf("%d", &matrix2[i][j]);

}

}

addMatrices(m, n, matrix1, matrix2, result);

printf("Sum of the matrices:\n");

displayMatrix(m, n, result);

return 0;

}

