Integer Restroring Division:

PROGRAM:

#include <stdio.h>

#include <stdlib.h>

// Function to perform integer restoring division

void restoringDivision(int dividend, int divisor, int \*quotient, int \*remainder) { int A = 0; // Remainder register (A) int Q = abs(dividend); // Dividend (Q register) int M = abs(divisor); // Divisor (M register)

int n = sizeof(int) \* 8; // Number of bits (assume 32 bits)

// Perform restoring division for (int i = 0; i < n; i++) {

A = (A << 1) | ((Q >> (n - 1)) & 1); // Shift left (A and Q)

Q = Q << 1;

A = A - M; // Subtract M from A

if (A < 0) {

Q = Q & ~(1); // Set the LSB of Q to 0

A = A + M; // Restore A

} else {

Q = Q | 1; // Set the LSB of Q to 1

}

}

// Set quotient and remainder

\*quotient = Q;

\*remainder = A;

// Adjust the signs of quotient and remainder

if (dividend < 0 && divisor > 0) {

\*quotient = -\*quotient;

} else if (dividend > 0 && divisor < 0) {

\*quotient = -\*quotient;

}

if (dividend < 0) {

\*remainder = -\*remainder;

}

}

int main() { int dividend, divisor;

int quotient, remainder;

// Input the dividend and divisor printf("Enter dividend: "); scanf("%d", &dividend); printf("Enter divisor: ");

scanf("%d", &divisor);

// Check for division by zero

if (divisor == 0) {

printf("Error: Division by zero is not allowed.\n"); return 1;

}

// Perform restoring division

restoringDivision(dividend, divisor, &quotient, &remainder);

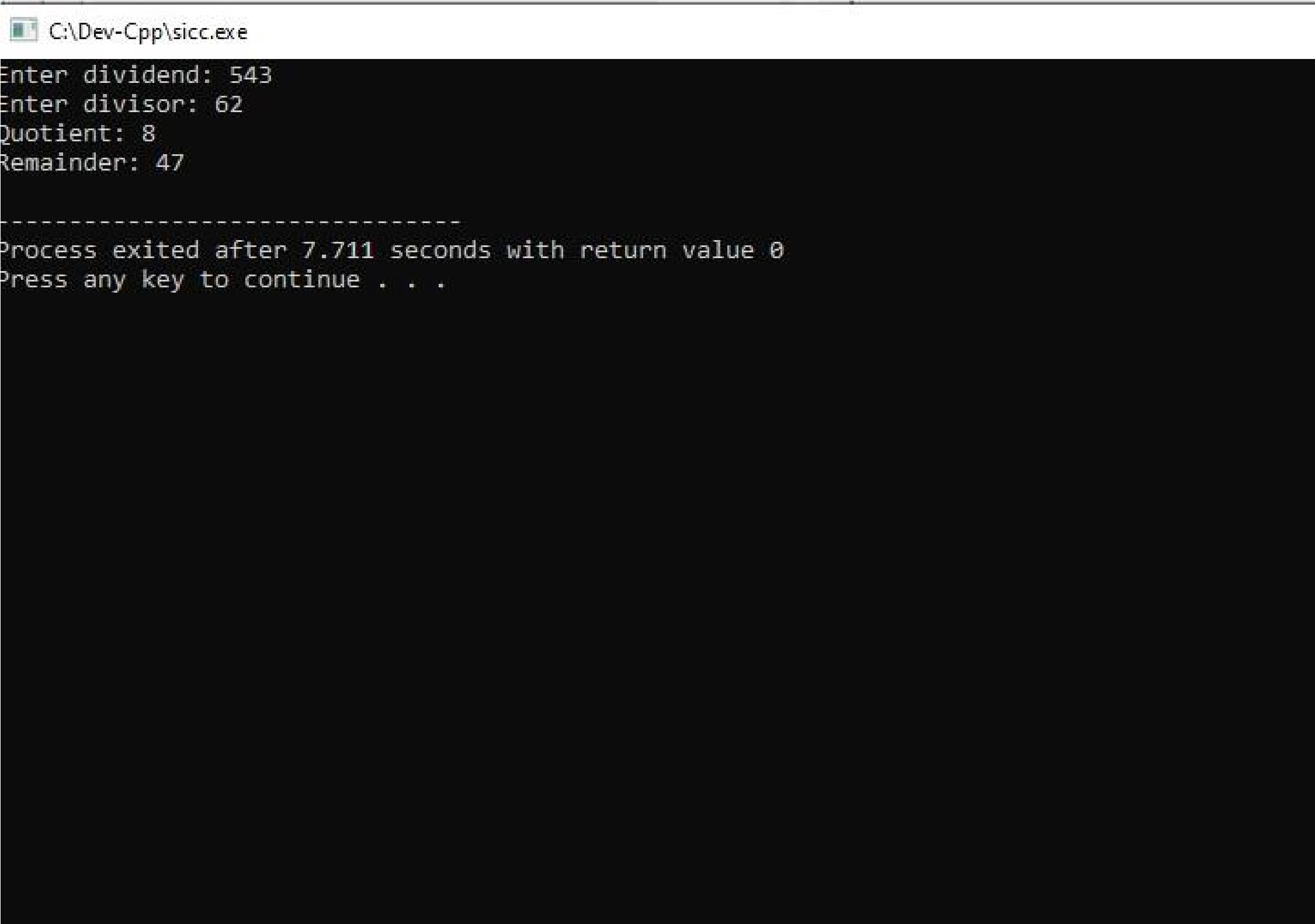
// Output the results printf("Quotient: %d\n", quotient);

printf("Remainder: %d\n", remainder);

return 0;

}

INPUT & OUTPUT:

**RESULT:** Thus the program was executed successfully using DevC++.