**Department of CSE**

**Compiler Lab (CSE 352)**

**Lab Report 03**

Checking Whether a Mathematical Statement is Solvable in C

**Submitted By :**

**Fahima Abida Chowdhury**

ID: 0432220005101135

Semester: Spring-2025

Batch: 52(6B1)

**Submitted To :**

**Md. Ismail**

Lecturer

Department of CSE

University of Information Technology and Sciences

**Experiment No.: 03**  
**Experiment Name:** Checking Whether a Mathematical Statement is Solvable in C

**Problem Statement**

The goal of this lab is to write a C program that checks whether a given mathematical expression is syntactically valid and solvable. The program focuses on identifying:

* Invalid characters that are not part of a standard mathematical expression.
* Obvious runtime errors like division by zero, which render an expression unsolvable.

This is a basic validation approach that simulates preprocessing steps before evaluating mathematical expressions.

**Implementation**

**C Code**

#include <stdio.h>

#include <ctype.h>

#include <string.h>

int isValidChar(char ch) {

return isdigit(ch) || ch == '+' || ch == '-' || ch == '\*' || ch == '/' ||

ch == '%' || ch == ' ' || ch == '(' || ch == ')';

}

int containsSimpleDivisionByZero(const char \*expr) {

for (int i = 0; expr[i]; i++) {

if (expr[i] == '/') {

int j = i + 1;

while (expr[j] == ' ') j++;

if (expr[j] == '0') {

int k = j;

while (expr[k] == '0') k++;

if (!isdigit(expr[k])) {

return 1;

}

}

}

}

return 0;

}

int main() {

char expr[100];

printf("Enter a mathematical expression: ");

fgets(expr, sizeof(expr), stdin);

size\_t len = strlen(expr);

if (len > 0 && expr[len - 1] == '\n') {

expr[len - 1] = '\0';

}

int valid = 1;

for (int i = 0; expr[i]; i++) {

if (!isValidChar(expr[i])) {

valid = 0;

break;

}

}

if (!valid) {

printf("Invalid characters in the expression.\n");

} else if (containsSimpleDivisionByZero(expr)) {

printf("Expression is not solvable (division by zero detected).\n");

} else {

printf("Expression looks solvable.\n");

}

return 0;

}

**Input**

The program takes a single line of input from the user, representing a mathematical expression.

**Example Input 1:**

12 + 4 / 2

**Example Input 2:**

5 + (3 \* 0) / 0

**Example Input 3:**

7 & 5

**Output**

Based on the input, the output can be one of the following:

**Example Output 1:**

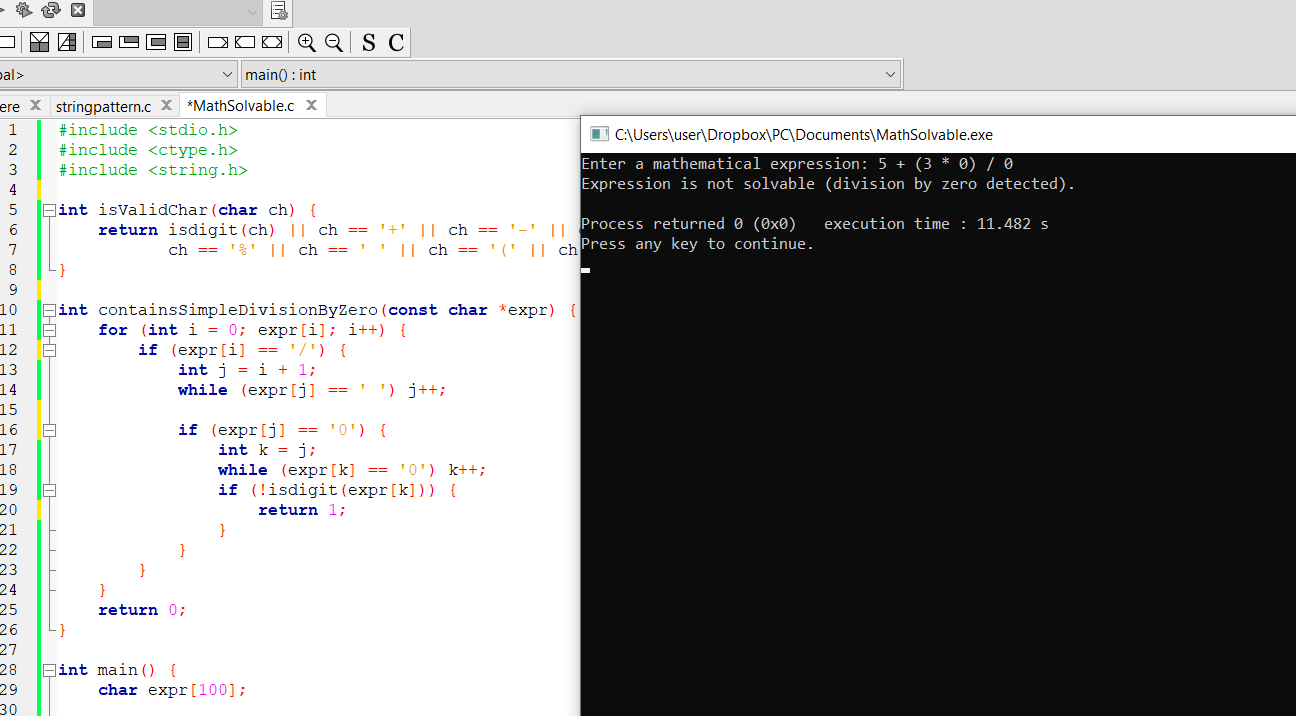
Expression looks solvable.

**Example Output 2:**

Expression is not solvable (division by zero).

**Example Output 3:**

Invalid characters in expression.

****

**Conclusion**

This experiment demonstrates a basic yet effective method to validate mathematical expressions for solvability in C. It checks for illegal characters and obvious mathematical errors like division by zero, enhancing the reliability of expression evaluation systems.

Such pre-evaluation checks are useful in interpreters, compilers, and calculator applications where robust input validation is necessary. While the program does not compute the result of the expression, it serves as a critical preprocessing step in expression analysis.