8.

CODE:

#include <stdio.h>

#include <string.h>

struct Symbol {

char name[20];

int address;

};

struct Symbol table[10];

int count = 0;

void insert(char \*name, int address) {

strcpy(table[count].name, name);

table[count].address = address;

count++;

}

void display() {

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

printf("%s %d\n", table[i].name, table[i].address);

}

}

int main() {

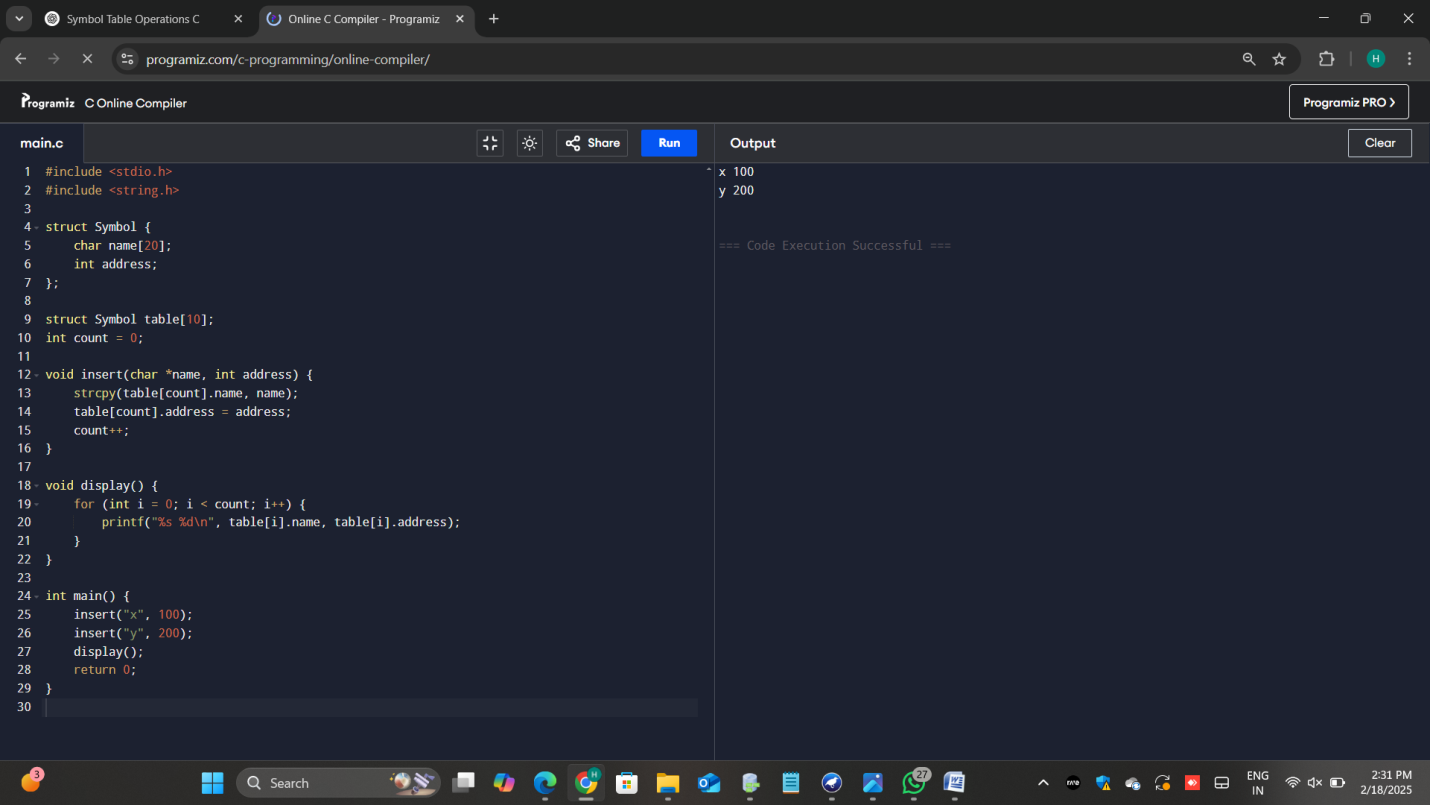
insert("x", 100);

insert("y", 200);

display();

return 0;

}



9.

CODE:

#include <stdio.h>

#include <string.h>

#include <ctype.h>

int isValidSentence(const char \*str) {

int len = strlen(str);

if (!isupper(str[0]) || str[len - 1] != '.') {

return 0;

}

for (int i = 1; i < len - 1; i++) {

if (!isalnum(str[i]) && str[i] != ' ') {

return 0;

}

}

return 1;

}

int main() {

char input[100];

printf("Enter a sentence: ");

fgets(input, sizeof(input), stdin);

input[strcspn(input, "\n")] = 0;

if (isValidSentence(input)) {

printf("Valid Sentence\n");

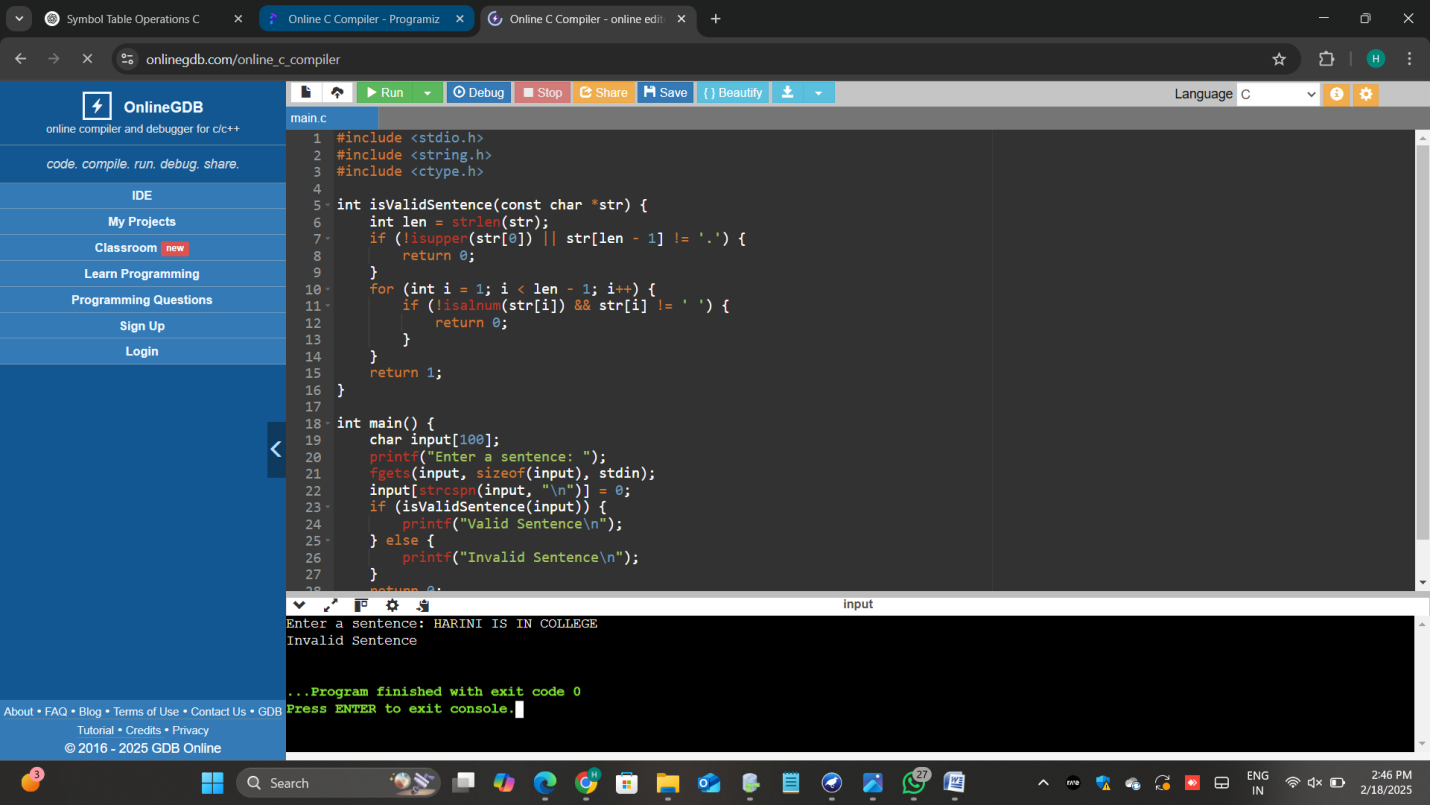
} else {

printf("Invalid Sentence\n");

}

return 0;

}



10.

CODE:

#include <stdio.h>

#include <string.h>

char \*input;

void E();

void T();

void F();

void match(char expected) {

if (\*input == expected) {

input++;

} else {

printf("Error: Unexpected character %c\n", \*input);

exit(1);

}

}

void E() {

T();

while (\*input == '+') {

match('+');

T();

}

}

void T() {

F();

while (\*input == '\*') {

match('\*');

F();

}

}

void F() {

if (\*input == '(') {

match('(');

E();

match(')');

} else if (\*input >= 'a' && \*input <= 'z') {

match(\*input);

} else {

printf("Error: Invalid character %c\n", \*input);

exit(1);

}

}

int main() {

char expr[100];

printf("Enter expression: ");

scanf("%s", expr);

input = expr;

E();

if (\*input == '\0') {

printf("Valid expression\n");

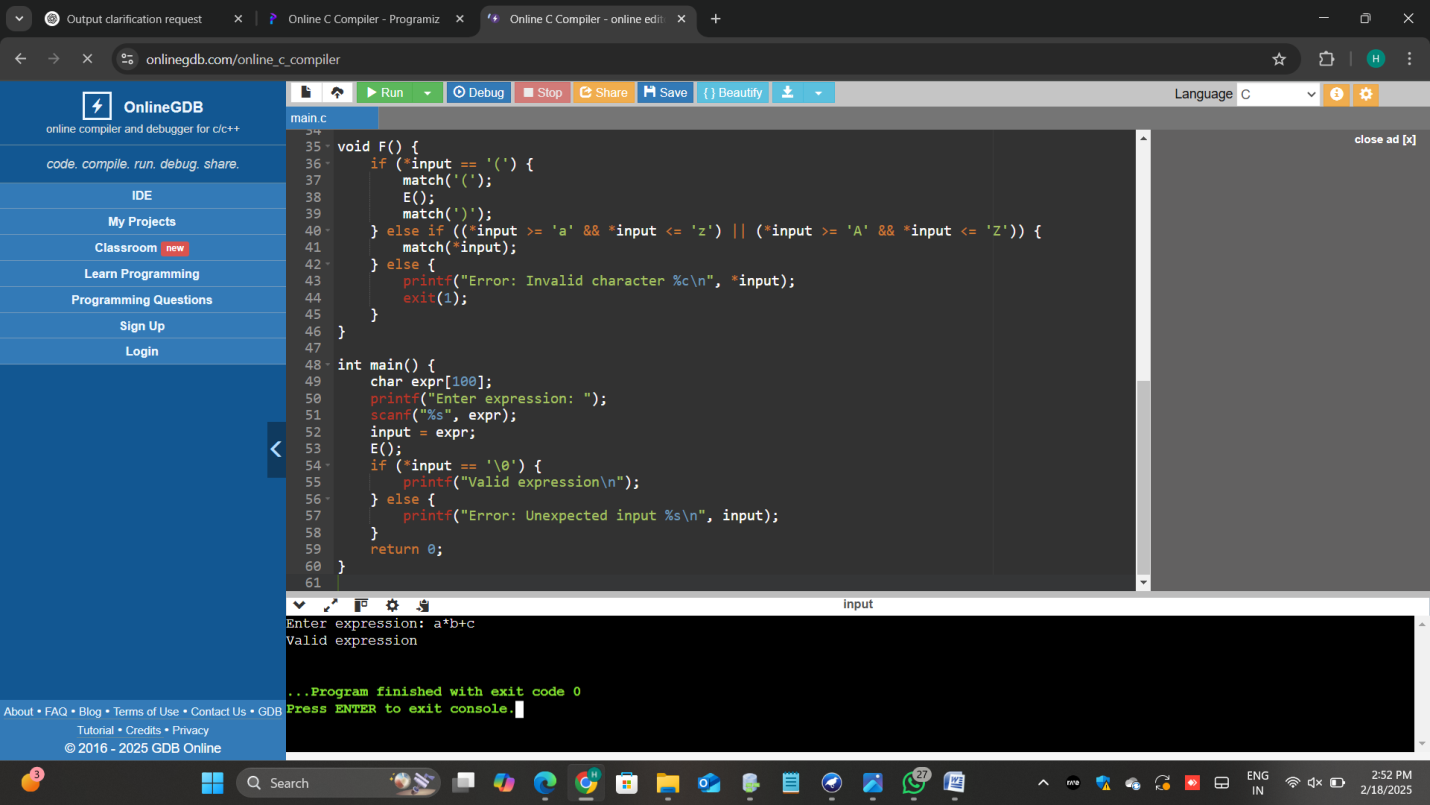
} else {

printf("Error: Unexpected input %s\n", input);

}

return 0;

}



11.

CODE:

#include <stdio.h>

#include <stdlib.h>

#include <math.h>

int precedence(char op) {

if (op == '+' || op == '-') return 1;

if (op == '\*' || op == '/') return 2;

if (op == '^') return 3;

return 0;

}

int evaluate(int a, int b, char op) {

if (op == '+') return a + b;

if (op == '-') return a - b;

if (op == '\*') return a \* b;

if (op == '/') return a / b;

return pow(a, b);

}

int evaluateExpression(char\* expr) {

int operands[100], operandCount = 0;

char operators[100], operatorCount = 0;

int i = 0;

while (expr[i]) {

if (expr[i] == ' ') { i++; continue; }

if (expr[i] >= '0' && expr[i] <= '9') {

int num = 0;

while (expr[i] >= '0' && expr[i] <= '9') num = num \* 10 + (expr[i++] - '0');

operands[operandCount++] = num;

} else if (expr[i] == '(') operators[operatorCount++] = expr[i++];

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

while (operatorCount && operators[operatorCount - 1] != '(')

operands[operandCount++] = evaluate(operands[--operandCount], operands[--operandCount], operators[--operatorCount]);

operatorCount--;

i++;

} else {

while (operatorCount && precedence(operators[operatorCount - 1]) >= precedence(expr[i]))

operands[operandCount++] = evaluate(operands[--operandCount], operands[--operandCount], operators[--operatorCount]);

operators[operatorCount++] = expr[i++];

}

}

while (operatorCount)

operands[operandCount++] = evaluate(operands[--operandCount], operands[--operandCount], operators[--operatorCount]);

return operands[0];

}

int main() {

char expr[100];

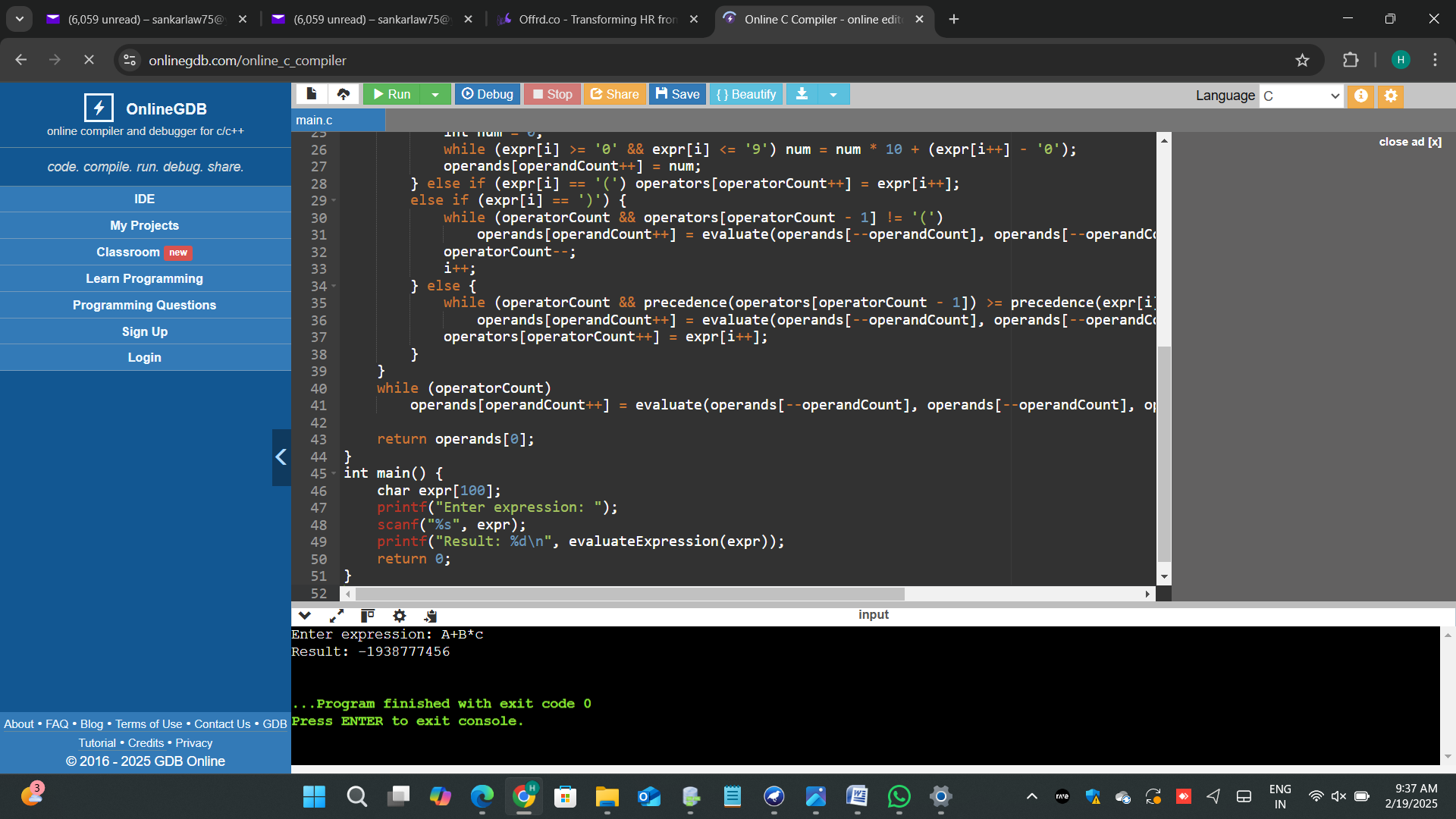
printf("Enter expression: ");

scanf("%s", expr);

printf("Result: %d\n", evaluateExpression(expr));

return 0;

}



12.

CODE:

#include <stdio.h>

#include <stdlib.h>

#include <ctype.h>

int main() {

char sentence[1000];

char ch;

int charCount = 0, wordCount = 0, lineCount = 1;

int inWord = 0;

// Prompt user for sentence

printf("Enter a sentence: ");

fgets(sentence, sizeof(sentence), stdin);

for (int i = 0; sentence[i] != '\0'; i++) {

ch = sentence[i];

charCount++;

// Check for new line

if (ch == '\n') {

lineCount++;

inWord = 0;

}

// Check for word boundaries

if (isspace(ch) || ch == '\n') {

inWord = 0;

} else if (!inWord) {

inWord = 1;

wordCount++;

}

}

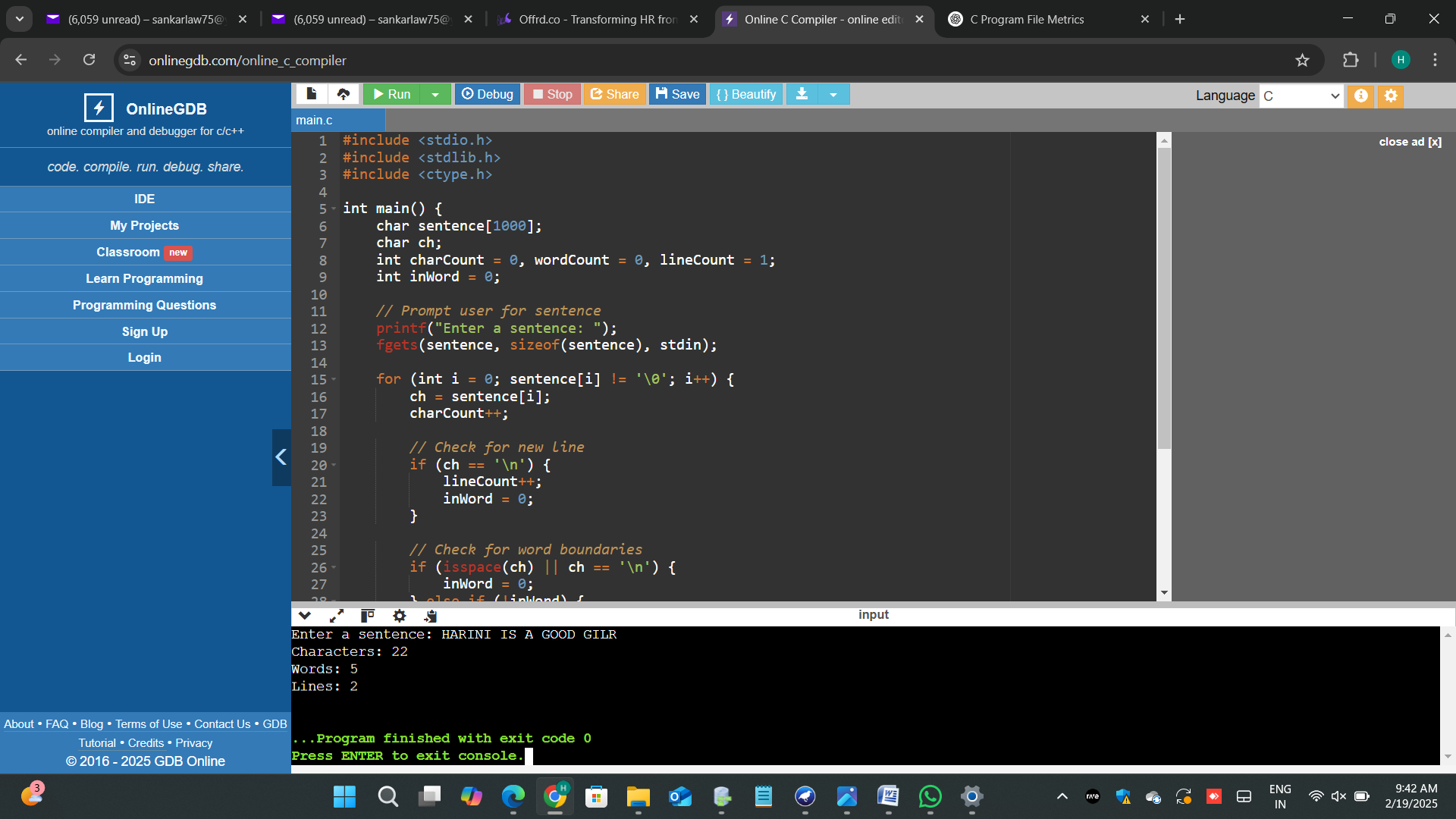
printf("Characters: %d\n", charCount);

printf("Words: %d\n", wordCount);

printf("Lines: %d\n", lineCount);

return 0;

}



13.

CODE:

#include <stdio.h>

int main() {

int a, b, c, d, x, y;

printf("Enter values for a, b, c, d: ");

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

int temp = a \* b;

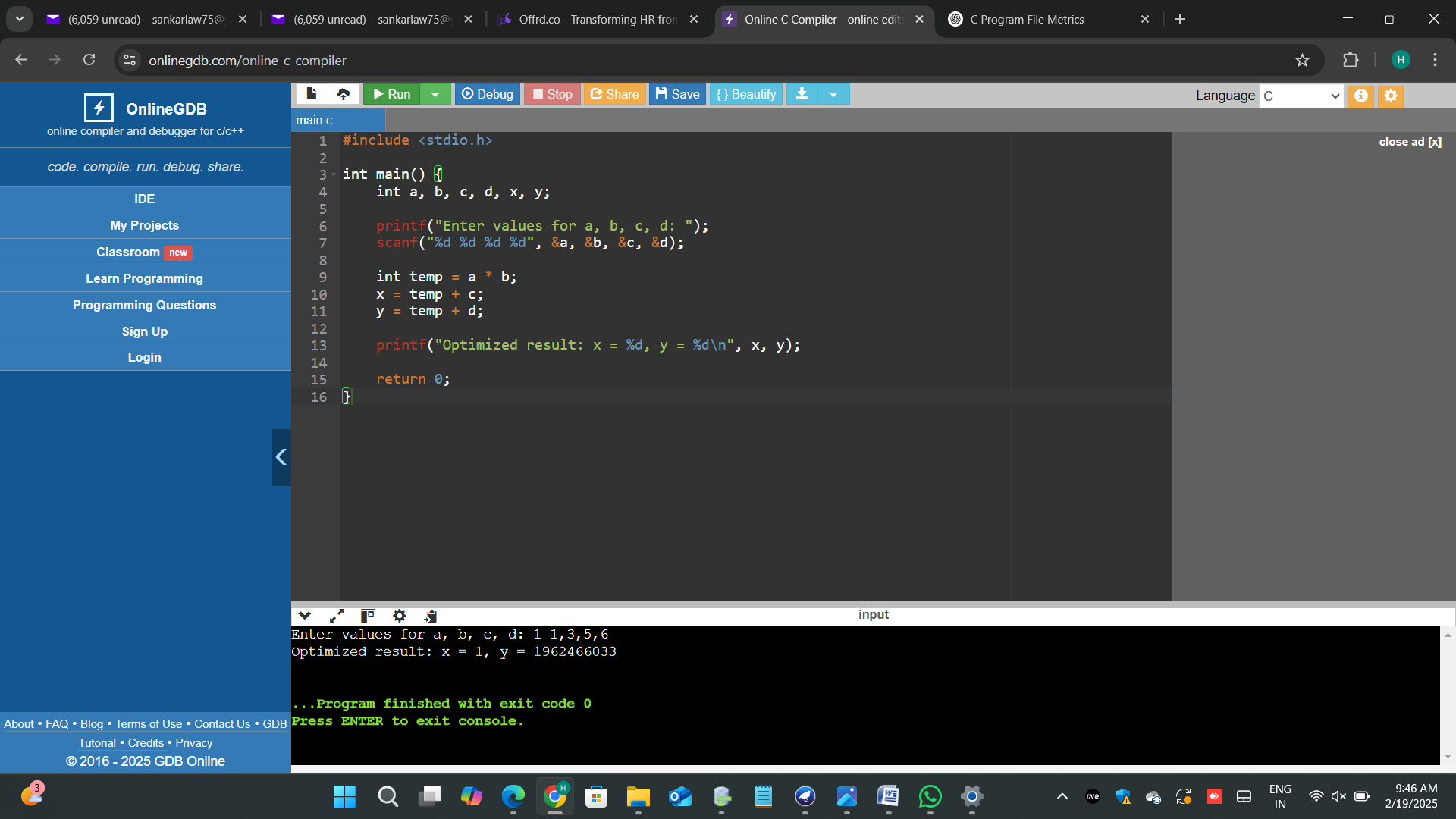
x = temp + c;

y = temp + d;

printf("Optimized result: x = %d, y = %d\n", x, y);

return 0;

}



14.

CODE:

#include <stdio.h>

#include <string.h>

void generateThreeAddressCode(char op, char arg1, char arg2, char result) {

printf("%c = %c %c %c\n", result, arg1, op, arg2);

}

int main() {

char expr[100];

char arg1, arg2, op, result = 'T';

printf("Enter a simple arithmetic expression (e.g., a+b): ");

scanf("%s", expr);

if (sscanf(expr, "%c%c%c", &arg1, &op, &arg2) == 3) {

generateThreeAddressCode(op, arg1, arg2, result);

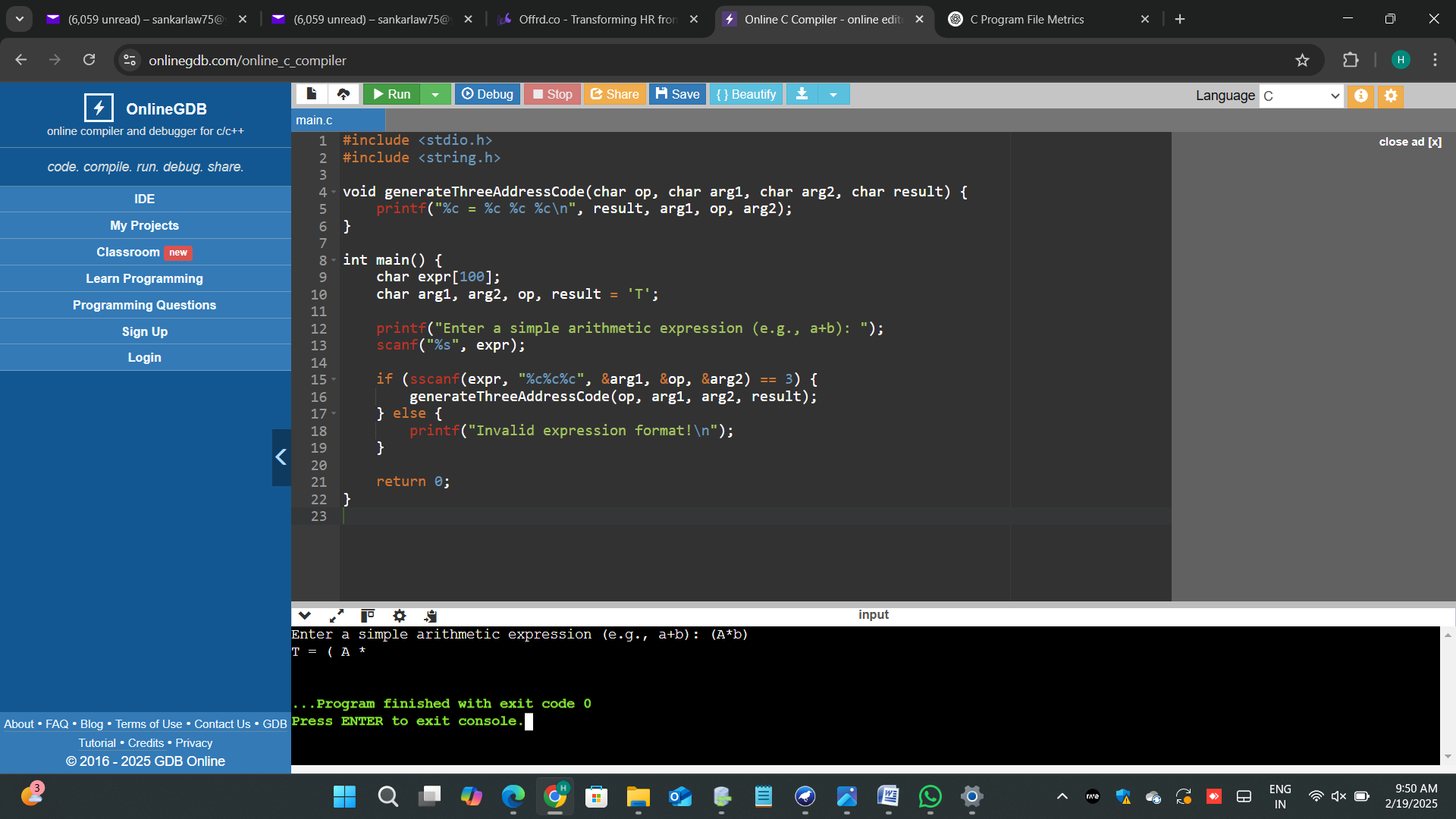
} else {

printf("Invalid expression format!\n");

}

return 0;

}



15.

CODE: