12.The main function of the Intermediate code generation is producing three address code statements for a given input expression. The three address codes help in determining the sequence in which operations are actioned by the compiler. The key work of Intermediate code generators is to simplify the process of Code Generator. Write a C Program to Generate the Three address code representation for the given input statement.

**PROGRAM:**

#include <stdio.h>

#include <stdlib.h>

int tempCount = 0;

char\* newTemp() {

char\* temp = (char\*)malloc(5);

sprintf(temp, "t%d", tempCount++);

return temp;

}

int isOperator(char c) {

return c == '+' || c == '-' || c == '\*' || c == '/';

}

const char\* skipWhitespace(const char\* input) {

while (\*input == ' ' || \*input == '\t') {

input++;

}

return input;

}

char\* parseNext(const char\* input, char\* buffer) {

input = skipWhitespace(input);

int i = 0;

while (\*input != '\0' && \*input != ' ' && !isOperator(\*input)) {

buffer[i++] = \*input++;

}

buffer[i] = '\0';

return buffer;

}

void genAdd(char\* op1, char\* op2, char\* result) {

printf("%s = %s + %s\n", result, op1, op2);

}

void genSub(char\* op1, char\* op2, char\* result) {

printf("%s = %s - %s\n", result, op1, op2);

}

void genMul(char\* op1, char\* op2, char\* result) {

printf("%s = %s \* %s\n", result, op1, op2);

}

void genDiv(char\* op1, char\* op2, char\* result) {

printf("%s = %s / %s\n", result, op1, op2);

}

int main() {

char expression[100];

char buffer[100];

printf("Enter an arithmetic expression: ");

fgets(expression, sizeof(expression), stdin);

const char\* input = expression;

char\* op1 = parseNext(input, buffer);

while (\*input != '\0') {

input = skipWhitespace(input);

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

break;

}

char op = \*input++;

char\* op2 = parseNext(input, buffer);

char\* result = newTemp();

if (op == '+') {

genAdd(op1, op2, result);

} else if (op == '-') {

genSub(op1, op2, result);

} else if (op == '\*') {

genMul(op1, op2, result);

} else if (op == '/') {

genDiv(op1, op2, result);

}

op1 = result;

}

return 0;

}

**OUTPUT:**

