15. TO ELIMINATE COMMON SUBEXPRESSION

CODE:

#include <stdio.h>

#include <string.h>

struct Expression {

char id;

char operation;

char operand1;

char operand2;

char result;

int isComputed;

};

int main() {

struct Expression expressions[100]; // Array to store expressions

int exprCount = 0; // Counter for the number of expressions

// Example expressions

expressions[exprCount++] = {'t', '+', 'a', 'b', ' ', 0}; // t = a + b

expressions[exprCount++] = {'u', '\*', 'c', 'd', ' ', 0}; // u = c \* d

expressions[exprCount++] = {'v', '-', 'a', 'b', ' ', 0}; // v = a - b

expressions[exprCount++] = {'w', '\*', 'c', 'd', ' ', 0}; // w = c \* d

// Eliminate common subexpressions

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

for (int j = i + 1; j < exprCount; j++) {

if (expressions[i].operation == expressions[j].operation &&

expressions[i].operand1 == expressions[j].operand1 &&

expressions[i].operand2 == expressions[j].operand2) {

expressions[j].isComputed = 1;

expressions[j].result = expressions[i].result;

}

}

}

// Display optimized expressions

printf("Optimized Expressions:\n");

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

if (!expressions[i].isComputed) {

printf("%c = %c %c %c\n", expressions[i].result,

expressions[i].operand1, expressions[i].operation,

expressions[i].operand2);

}

}

return 0;

}

OUTPUT:

Optimized Expressions:

= a + b

= c \* d

= a - b

--------------------------------

Process exited after 0.05733 seconds with return value 0

Press any key to continue . . .

