EXP NO : 12 **DATE : 28.04.25**

IMPLEMENT CODE OPTIMIZATION TECHNIQUES
COPY PROPAGATION

AIM:

The aim is to implement code optimization techniques like Dead Code Elimination (DCE) and Common Subexpression Elimination (CSE) to improve the efficiency and performance of a program. These techniques are applied to intermediate code (e.g., Three-Address Code or TAC) during the compilation process.

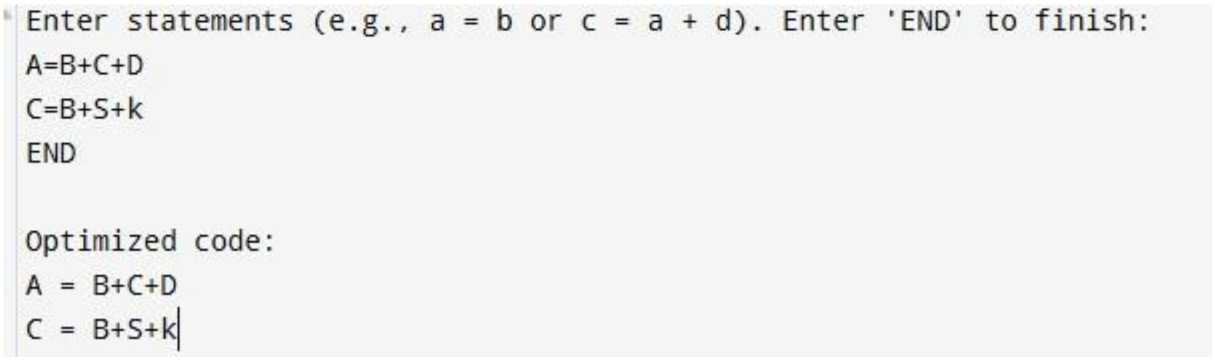
ALGORITHM:

- The desired header files are declared.
- The two file pointers are initialized one for reading the C program from the file and one for writing the converted program with constant folding
- The file is read and checked if there are any digits or operands present.
- If there is, then the evaluations are to be computed in switch case and stored.
- Copy the stored data to another file.
- Print the copied data file.

PROGRAM:

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#define MAX_LINES 100
#define MAX_LENGTH 50
typedef struct {
    char var[MAX_LENGTH];
    char value[MAX_LENGTH];
    int is_direct_assignment;
} Statement;
void apply_copy_propagation(Statement statements[], int count) {
    for (int i = 0; i < count; i++) {
        if (statements[i].is_direct_assignment) {
            char *lhs = statements[i].var;
            char *rhs = statements[i].value;
            for (int j = i + 1; j < count; j++) {
                if (statements[j].is_direct_assignment) {
                    if (strcmp(statements[j].value, lhs) == 0) {
                        strcpy(statements[j].value, rhs);
                    }
                } else {
                    char *pos = strstr(statements[j].value, lhs);
                    if (pos != NULL) {
                        char temp[MAX_LENGTH];
                        strcpy(temp, pos + strlen(lhs));
                    *pos = '\0';
                    strcat(statements[j].value, rhs);
                    strcat(statements[j].value, temp);
                }
            }
        }
    }
  }
}

int main() {
    Statement statements[MAX_LINES];
    int count = 0;

    printf("Enter statements (e.g., a = b or c = a + d). Enter 'END' to finish:\n");
    char line[MAX_LENGTH];
    while (fgets(line, sizeof(line), stdin)) {
        if (strncmp(line, "END", 3) == 0) break;
        line[strcspn(line, "\n")] = 0; // Remove newline character

        char *equals = strchr(line, '=');
        if (equals != NULL) {
            *equals = '\0';
            strcpy(statements[count].var, line);
            strcpy(statements[count].value, equals + 1);
            statements[count].is_direct_assignment = (strchr(equals + 1, '+') == NULL &&
                                                     strchr(equals + 1, '-') == NULL &&
                                                     strchr(equals + 1, '*') == NULL &&
                                                     strchr(equals + 1, '/') == NULL);
            count++;
        }
    }

    apply_copy_propagation(statements, count);

    printf("\nOptimized code:\n");
    for (int i = 0; i < count; i++) {
        if (!(statements[i].is_direct_assignment && statements[i].value[0] == '\0')) {
            printf("%s = %s\n", statements[i].var, statements[i].value);
        }
    }

    return 0;
}
```

OUTPUT :



```
Enter statements (e.g., a = b or c = a + d). Enter 'END' to finish:
A=B+C+D
C=B+S+k
END

Optimized code:
A = B+C+D
C = B+S+k
```

RESULT:

Thus the above to implement code optimization techniques for copy propagation is executed successfully.