19CSE401 Compiler Design Lab

Anindita Das Badhan
CH.EN.U4CSE22180
4th Year CSE-B

Lab Exercise- 07

Aim: To implementation of Code Optimization Techniques

Code:

```c
#include <stdio.h>
#include <string.h>

struct op {
    char l;      // Left-hand side (e.g., a = ...)
    char r[20];  // Right-hand side (e.g., ... = b + c)
} op[10], pr[10];

int main() {
    int i, j, n, z = 0, k, m, q, a;
    char temp, t;
    char *p, *l;
    char *tem;

    printf("Enter the number of expressions: ");
    scanf("%d", &n);

    // Input expressions
    for (i = 0; i < n; i++) {
        printf("Left: ");
        scanf(" %c", &op[i].l); // Space before %c to ignore newline
        printf("Right: ");
        scanf(" %s", op[i].r);
    }

    // Show intermediate code
    printf("\nIntermediate Code:\n");
    for (i = 0; i < n; i++) {
        printf("%c = %s\n", op[i].l, op[i].r);
    }
```

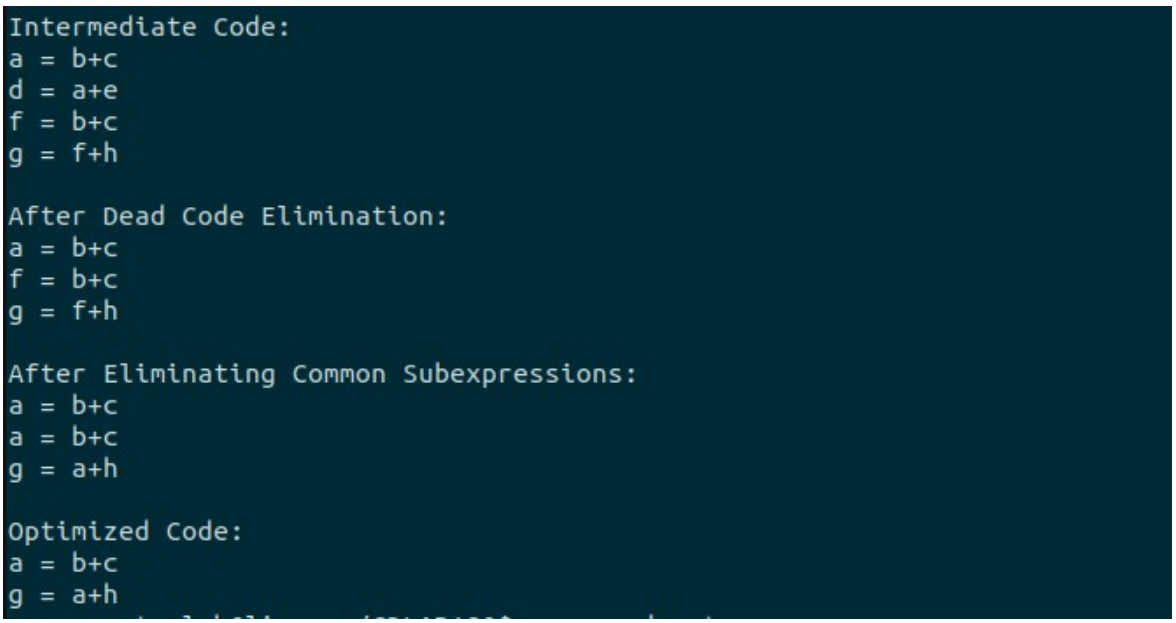
```c
    // DEAD CODE ELIMINATION
    for (i = 0; i < n - 1; i++) {
        temp = op[i].l;
        for (j = 0; j < n; j++) {
            p = strchr(op[j].r, temp); // Check if temp is used in any expression
            if (p) {
                pr[z].l = op[i].l;
                strcpy(pr[z].r, op[i].r);
                z++;
                break;
            }
        }
    }

    // Add the last expression (assumed as output-related)
    pr[z].l = op[n - 1].l;
    strcpy(pr[z].r, op[n - 1].r);
    z++;

    printf("\nAfter Dead Code Elimination:\n");
    for (k = 0; k < z; k++) {
        printf("%c = %s\n", pr[k].l, pr[k].r);
    }
```

```
    // COMMON SUBEXPRESSION ELIMINATION
    for (m = 0; m < z; m++) {
        tem = pr[m].r;
        for (j = m + 1; j < z; j++) {
            p = strstr(tem, pr[j].r);
            if (p) {
                t = pr[j].l;
                pr[j].l = pr[m].l;

                for (i = 0; i < z; i++) {
                    l = strchr(pr[i].r, t);
                    if (l) {
                        a = l - pr[i].r;
                        pr[i].r[a] = pr[m].l;
                    }
                }
            }
        }
    }

    printf("\nAfter Eliminating Common Subexpressions:\n");
    for (i = 0; i < z; i++) {
        printf("%c = %s\n", pr[i].l, pr[i].r);
    }

    // REMOVE DUPLICATE ASSIGNMENTS (e.g., a=b+c and again a=b+c)
    for (i = 0; i < z; i++) {
        for (j = i + 1; j < z; j++) {
            q = strcmp(pr[i].r, pr[j].r);
            if ((pr[i].l == pr[j].l) && !q) {
                pr[j].l = '\0'; // Mark for removal
            }
        }
    }
```

```
    // FINAL OUTPUT
    printf("\nOptimized Code:\n");
    for (i = 0; i < z; i++) {
        if (pr[i].l != '\0') {
            printf("%c = %s\n", pr[i].l, pr[i].r);
        }
    }

    return 0;
}
```

Output:

```
asecomputerlab@linux:~/CDLAB180$ nano codeopt.c
asecomputerlab@linux:~/CDLAB180$ gcc codeopt.c -o codeopt
asecomputerlab@linux:~/CDLAB180$ ./codeopt
Enter the number of expressions: 4
Left: a
Right: b+c
Left: d
Right: a+e
Left: f
Right: b+c
Left: g
Right: f+h
```



Result: Thus, the program to implement code optimization has been executed successfully