Name: Aman Kumar Rauniyar
Roll No: CH.EN.U4CSE22174

Lab-8

Title: To implementation of Code Optimization Techniques.

Code:

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

struct op {
    char l;
    char r[20];
} op[10], pr[10];

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

    printf("Enter the Number of Values: ");
    scanf("%d", &n);

    for (i = 0; i < n; i++) {
        printf("left: ");
        scanf(" %c", &op[i].l);
        printf("right: ");
        scanf(" %s", op[i].r);
    }

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

```
// Dead code elimination part
for (i = 0; i < n - 1; i++) {
    temp = op[i].l;
    for (j = 0; j < n; j++) {
        p = strchr(op[j].r, temp);
        if (p) {
            pr[z].l = op[i].l;
            strcpy(pr[z].r, op[i].r);
            z++;
            break; // Once found, no need to add duplicates
        }
    }
}
// Add last statement as it is
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 Expressions\n");
    for (i = 0; i < z; i++) {
        printf("%c = %s\n", pr[i].l, pr[i].r);
    }

    // Remove duplicates by marking them '\0'
    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 == 0)) {
                pr[i].l = '\0';
            }
        }
    }

    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@asecomputerlab-hp-prodesk-400-g7-micrtower-pc:~/Desktop/lab$ nano dead_code_elimination.c
asecomputerlab@asecomputerlab-hp-prodesk-400-g7-micrtower-pc:~/Desktop/lab$ gcc -o dead_code dead_code_elimination.c
asecomputerlab@asecomputerlab-hp-prodesk-400-g7-micrtower-pc:~/Desktop/lab$ ./dead_code
Enter the Number of Values: 4
left: a
right: b+c
left: d
right: a+e
left: f
right: d+g
left: h
right: a+i

Intermediate Code
a = b+c
d = a+e
f = d+g
h = a+i

After Dead Code Elimination
a = b+c
d = a+e
h = a+i

After Eliminating Common Expressions
a = b+c
d = a+e
h = a+i

Optimized Code
a = b+c
d = a+e
h = a+i
```