Ex No: 10

Date:

IMPLEMENT CODE OPTIMIZATION TECHNIQUES DEAD CODE AND COMMON SUB EXPRESSION ELIMINATION

AIM:

To write a C program to implement the dead code elimination and common sub expression elimination (code optimization) techniques.

ALGORITHM:

- Start
- Create the input file which contains three address code.
- Open the file in read mode.
- If the file pointer returns NULL, exit the program else go to 5.
- Scan the input symbol from left to right.
- Store the first expression in a string.
- Compare the string with the other expressions in the file.
- If there is a match, remove the expression from the input file.
- Perform these steps 5-8 for all the input symbols in the file.
- Scan the input symbol from the file from left to right.
- Get the operand before the operator from the three address code.
- Check whether the operand is used in any other expression in the three address code.
- If the operand is not used, then eliminate the complete expression from the three-address code else go to 14.
- Perform steps 11 to 13 for all the operands in the three address code till end of the file is reached.
- Stop.

PROGRAM:

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

struct op {
   char l;
   char r[20];
};

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

   printf("Enter number of values: ");
```

```
    scanf("%d", &n);

    struct op op[n], pr[10];

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

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

    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++;
            }
        }
    }
    pr[z].l = op[n - 1].l;
    strcpy(pr[z].r, op[n - 1].r);
    z++;

    printf("\nAfter dead code elimination:\n");
    for (int k = 0; k < z; k++) {
        printf("%c=%s\n", pr[k].l, pr[k].r);
    }

    // Sub-expression elimination and duplicate production elimination code goes here...

    printf("Optimized 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:

```
vimal@KBVIMAL:~$ vi 301deadcode.c
vimal@KBVIMAL:~$ gcc 301deadcode.c
vimal@KBVIMAL:~$ ./a.out
Enter number of values: 3
Enter left and right values:
        left: a
        right: b
        left: c
        right: d
        left: e
        right: f
Intermediate Code:
a=b
c=d
e=f

After dead code elimination:
e=f
Optimized Code:
e=f
vimal@KBVIMAL:~$
```

RESULT: