Practical 10

Name: Rajatkumar Patel
Roll No.: 18BCE191

Aim

To implement Code Optimization techniques.

Code

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

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

void 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("Enter expression: ");
   char * exp;
   scanf("%s",exp);
   op[i].l = exp[0];
   strcpy(op[i].r,(exp+2));
 }
 printf("Intermediate Code\n");
 for (i = 0; i < n; i++) {
   printf("%c=", op[i].l);
   printf("%s\n", 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 (k = 0; k < z; k++) {
  printf("%ct=", pr[k].l);
  printf("%s\n", pr[k].r);
}
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;
          printf("pos: %d", a);
          pr[i].r[a] = pr[m].l;
        }
      }
    }
  }
}
printf("\nEliminate Common Expression\n");
for (i = 0; i < z; i++) {
  printf("%c\t=", pr[i].l);
  printf("%s\n", pr[i].r);
}
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[i].l = '\0';
      strcpy(pr[i].r, '\0');
    }
```

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

Input-Output

```
(base) rajat@rajat-VivoBook-S14-X430UA:/Rajat1/Books/Compiler Construction/Practicals/Practical10$ ./Prac10
Enter the Number of Values:3
Enter expression: a=b+c
Enter expression: d=b+c
Enter expression: e=a*d
Intermediate Code
a=b+c
d=b+c
e=a*d

After Dead Code Elimination
at=b+c
dt=b+c
et=a*d
pos: 2
Eliminate Common Expression
a       =b+c
a       =b+c
e       =a*a
```

```
(base) rajat@rajat-VivoBook-S14-X430UA:/Rajat1/Books/Compiler Construction/Practicals/Practical10$ ./Prac10
Enter the Number of Values:4
Enter expression: a=b+c
Enter expression: d=h-g
Enter expression: e=f/q
Enter expression: z=d-e
Intermediate Code
a=b+c
d=h-g
e=f/q
z=d-e

After Dead Code Elimination
dt=h-g
et=f/q
zt=d-e

Eliminate Common Expression
d       =h-g
e       =f/q
z       =d-e
Optimized Code
d=h-g
e=f/q
z=d-e
```

Conclusion: Implemented code optimization technique with the elimination of common expression and dead code elimination.