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<conio.h
>
#include<string.h
> struct op
{
  char l;
  char r[20];
 }
```

ROLL NO:210701115

125

```c
op[10], pr[10];

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

  temp, t; char * tem;
  clrscr(); printf("enter
  no of values");
  scanf("%d", & n);  for (i
  = 0; i < n; i++)
{ printf("\tleft\t");
  op[i].l = getche();
  printf("\tright:\t");
  scanf("%s",
  op[i].r);
 }
  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("%c\t=", pr[k].l);
  printf("%s\n", pr[k].r);
 }

 //sub expression elimination
 for (m = 0; m
```

```c
  < 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("eliminate common expression\n");
  for (i = 0; i < z; i++) {
   printf("%c\t=",     pr[i].l); printf("%s\n",
   pr[i].r);
  }
  // duplicate production elimination

  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"); for
  (i = 0; i < z; i++)
{ if (pr[i].l != '\0') {  printf("%c=",
    pr[i].l);  printf("%s\n",
    pr[i].r);
   } } getch();
}
```
ROLL NO:210701115

OUTPUT:

```
┌──(kali㉿kali)-[~/Documents/cdlab]
└─$ vi exp10.c

┌──(kali㉿kali)-[~/Documents/cdlab]
└─$ gcc exp10.c

┌──(kali㉿kali)-[~/Documents/cdlab]
└─$ ./a.out
Enter no of values: 5
        Left:    a
        Right:   9
        Left:    b
        Right:   c+d
        Left:    e
        Right:   c+d
        Left:    f
        Right:   b+e
        Left:    r
        Right:   f

Intermediate Code
a=9
b=c+d
e=c+d
f=b+e
r=f

After Dead Code Elimination
b        =c+d
e        =c+d
f        =b+e
r        =f

Eliminate Common Expression
b        =c+d
b        =c+d
f        =b+b
r        =f
```

RESULT:

Thus, a C program to implement the dead code elimination and common sub expression elimination (code optimization) techniques has been developed.