PRACTICAL-9

AIM:

Implementation of code optimization for Common sub-expression elimination, Loop in variant code movement.

PROGRAM CODE:

```
#include <stdio.h>
#include <conio.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;
//clrscr();
printf("\nPARTH PATEL\n19DCS098\n");
printf("enter no of values=");
scanf("%d", &n);
//n=5;
for (i = 0; i < n; i++)
{
printf("\t left: \t");

scanf(" %c", &op[i].l);
printf("\t right: \t");
scanf("%s", op[i].r);
}
/*for (i = 0; i < n; i++)
{
printf("\n right: \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("\n after 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 < 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';
//pr[i].r = "NULL";
strcpy( pr[i].r , "NULL");
}
}
}
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);
}
}
//getch();
}
```

OUTPUT:

```
enter no of values=3
        left:  a
        right:         b+c
        left:  b
        right:         5
        left:  c
        right:         7*2
 intermediate Code
 a= b+c
 b= 5
 c= 7*2

 after dead code elimination
b =      5
c =      7*2
eliminate common expression
b        =5
c        =7*2
optimized code
b =5
c =7*2
```