**EX NO:10  
  
DATE:**

**IMPLEMENTATION OF CODE** **OPTIMIZER**

**AIM:**

To implement code optimizer phase of a compiler that eliminates dead code and common sub-expressions.

**ALGORITHM:**

**STEP 1:** Start

**STEP 2:** Define a structure S[20] that has members l,r,ad,f.   
**STEP 3:** Get the number of expressions from the user.   
**STEP 4:** Store the input in variable n.

**STEP 5**: Get n number of left and right and store it for each elements of S as S[i].l and S[i].r

**STEP 6:** Display the intermediate code generated as s[i].l=s[i].r

**STEP 7:** Initialize the value of f for all the nodes of Struct S as 0.

**STEP 8:** Check if the r value is getting repeated on any l value if it gets repeated change f value of the node to 1

**STEP 9:** compare the r of the node and r of previous nodes if equal print s[i].l=s[i].r

**STEP 10:** Else continue comparing the node with other previous nodes.

**STEP 11:** End

**PROGRAM**:

#include<stdio.h>

#include<string.h>

struct codo

{

char l[20],r[20],ad[20];

int f;

}s[20];

void main()

{

int i,j,k=0,n,ff=0;

printf("\nEnter total no.of.Expression:");

scanf("%d",&n);

printf("\nL,R-values:\n");

for(i=0;i<n;i++)

{

printf("\nLeft : ");

scanf("%s",s[i].l);

printf("Right : ");

scanf("%s",s[i].r);

}

printf("\nIntermediate code:\n"); for(i=0;i<n;i++)

printf("\n%s=%s",s[i].l,s[i].r);

printf("\nAfter dead code Elimination:\n"); for(i=0;i<n;i++)

s[i].f=0;

for(i=0;i<n;i++)

{

for(j=0;j<n;j++)

{

if(strstr(s[j].r,s[i].l))

{

s[i].f=1;

}

}

}

for(i=0;i<n;i++)

{

if(s[i].f==1)

{

printf("\n%s=%s",s[i].l,s[i].r);

}

}

printf("\nCommon Sub Expressions are..."); for(i=0;i<n;i++)

{

for(j=0;j<i;j++)

{

if(strcmp(s[i].r,s[j].r)==0)

{

printf("\n%s=%s\n%s=%s",s[j].l,s[j].r,s[i].l,s[i].r); break;

}

}

}

printf("\nAfter Eliminating common sub expressions:\n\n");

for(i=0;i<n;i++)

{

ff=0;

for(j=0;j<i;j++)

{

if(strcmp(s[i].r,s[j].r)==0)

{

ff=1;

break;

}

}

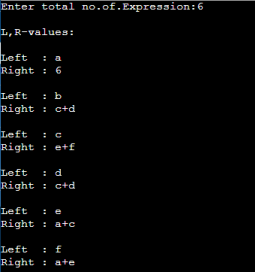
if(ff==0&&s[i].f)

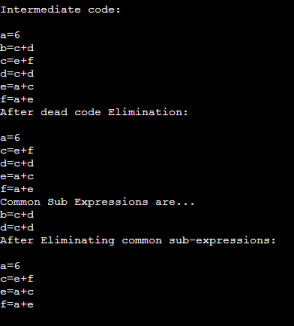
printf("%s=%s\n",s[i].l,s[i].r);

}

}

**OUTPUT:**





**RESULT:**

Thus the implementation of code optimizer phase of a compiler that eliminates dead code and common sub expressions is completed and verified successfully.