Exp16 : Constant Propogation

1. Start
2. Define structure in three address format (operator , argument1 , argument2 , result)
3. Read no of expressions in n
4. Read the expressions line by line (i) in struct array – arr
5. for i=0 to n do
    1. if (isdigit(arr[i].arg1) & isdigit(arr[i].arg2) or isdigit(arr[i].arg1) & arr[i].op = '=')
        1. op = arr[i].op
        2. arg1 = atoi(arr[i].arg1)
        3. arg2 = atoi(arr[i].arg2)
        4. org = arr[i].res
        5. switch (op)
            case + : res = arg1 + arg2
            case - : res = arg1 - arg2
            case * : res = arg1 * arg2
            case / : res = arg1 / arg2
            case % : res = arg1 % arg2
            case = : res = arg1
        6. res1 = string(res)
        7. change(i , res1 , org)
6. print arr

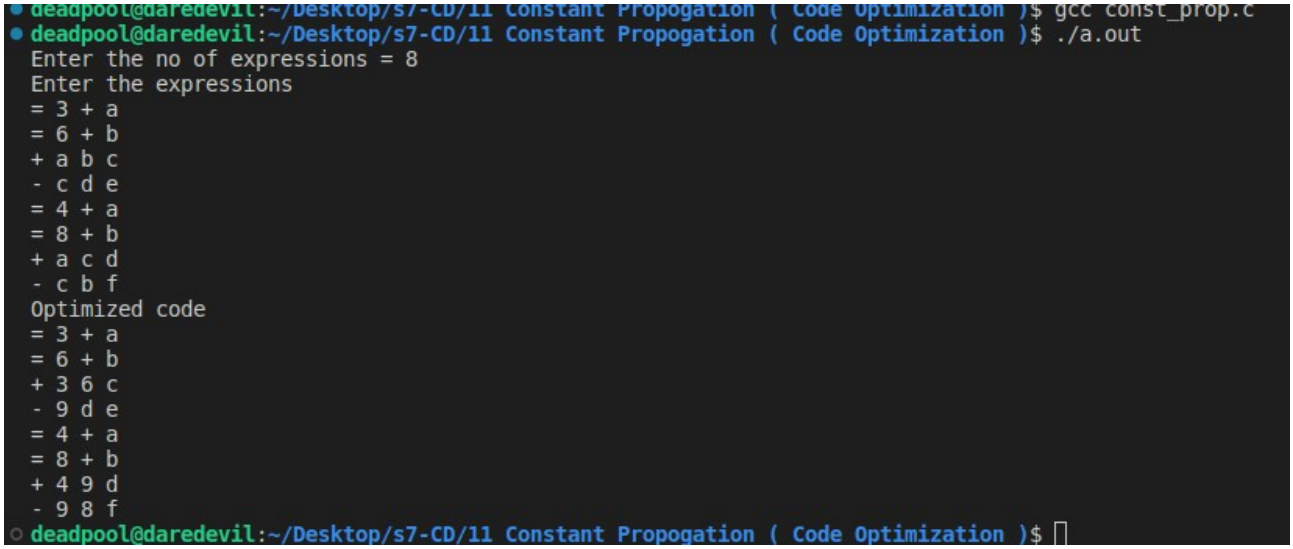
7. change (p , *res , org[])
    1. for i=p+1 to n do
        1. if(arr[i].op = "=" & arr[i].res = org) then
            break
        2. else
            1. if(arr[p].res = arr[i].arg1) then
                arr[i].arg1 = res
            2. if(arr[p].res = arr[i].arg2) then
                arr[i].arg2 = res
8. Stop

Constant Propogation (C)

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

struct exp{
   char op[2],arg1[5],arg2[5],res[5];
}arr[10];
int n;

void change(int p,char *res,char org[5]){
   for(int i=p+1;i<n;i++){
      if(strcmp(arr[i].res,org)==0){
         break;
      }else{
         if(strcmp(arr[p].res,arr[i].arg1)==0){
            strcpy(arr[i].arg1,res);
```

```
            }
            if(strcmp(arr[p].res,arr[i].arg2)==0){
                strcpy(arr[i].arg2,res);
            }
        }
    }
}

int main(){
    int i,arg1,arg2,res;
    char op,res1[5],org[5];

    printf("Enter the no of expressions = ");
    scanf("%d",&n);
    printf("Enter the expressions\n");
    for(i=0;i<n;i++){
        scanf("%s%s%s%s",arr[i].op,arr[i].arg1,arr[i].arg2,arr[i].res);
    }
    for(i=0;i<n;i++){
        if(isdigit(arr[i].arg1[0]) && isdigit(arr[i].arg2[0]) || strcmp(arr[i].op,"=")==0 &&
isdigit(arr[i].arg1[0])){
            op = arr[i].op[0];
            arg1 = atoi(arr[i].arg1);
            arg2 = atoi(arr[i].arg2);
            strcpy(org, arr[i].res);
            switch(op){
                case '+' : res = arg1+arg2; break;
                case '-' : res = arg1-arg2; break;
                case '*' : res = arg1*arg2; break;
                case '/' : res = arg1/arg2; break;
                case '%' : res = arg1%arg2; break;
                case '=' : res = arg1; break;
                default : break;
            }
            sprintf(res1,"%d",res);
            change(i,res1,org);
        }
    }
    printf("Optimized code\n");
    for(i=0;i<n;i++){
        printf("%s %s %s %s\n",arr[i].op,arr[i].arg1,arr[i].arg2,arr[i].res);
    }
    return 0;
}
```

output


deadpool@daredevil:~/Desktop/s7-CD/11 Constant Propogation (Code Optimization)$ gcc const_prop.c
deadpool@daredevil:~/Desktop/s7-CD/11 Constant Propogation (Code Optimization)$ ./a.out
Enter the no of expressions = 8
Enter the expressions
= 3 + a
= 6 + b
+ a b c
- c d e
= 4 + a
= 8 + b
+ a c d
- c b f
Optimized code
= 3 + a
= 6 + b
+ 3 6 c
- 9 d e
= 4 + a
= 8 + b
+ 4 9 d
- 9 8 f
deadpool@daredevil:~/Desktop/s7-CD/11 Constant Propogation (Code Optimization)$