Practical - 10

2CS701 – Compiler Construction

Harshit Gajipara
19BCE059

Aim:

To implement Code Optimization techniques: Implement any code optimization technique.

Code:

`practical10.c`

```c
#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("left: ");
        scanf(" %c", &op[i].l);
        printf("right: ");
        scanf(" %s", &op[i].r);
    }

    printf("\nIntermediate 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);
    }

    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\n", 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';
            }
        }
    }

    printf("\nOptimized Code\n");
    for (i = 0; i < z; i++)
```

```
    {
        if (pr[i].l != '\0')
        {
            printf("%c=", pr[i].l);
            printf("%s\n", pr[i].r);
        }
    }
}
```

Output:

```
Enter the Number of code lines : 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
pos: 2

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

Optimized Code
b=c+d
f=b+b
r=f
```

```
Enter the Number of code lines : 4
left : q
right : 32
left : p
right : a*b
left : s
right : a*b
left : r
right : p*s

Intermediate Code
q=32
p=a*b
s=a*b
r=p*s

After Dead Code Elimination
p = a*b
s = a*b
r = p*s
pos: 2

Eliminate Common Expression
p = a*b
p = a*b
r = p*p

Optimized Code
p=a*b
r=p*p
```

```
Enter the Number of code lines : 3
left : c
right : a*b
left : d
right : a*b+4
left : x
right : c

Intermediate Code
c=a*b
d=a*b+4
x=c

After Dead Code Elimination
c = a*b
x = c

Eliminate Common Expression
c = a*b
x = c

Optimized Code
c=a*b
x=c
```