**Ex No:13**

**Date:12-05-2021**

# Implementation of DAG

**Aim:** To generate DAG for the intermediate code.

## Program:

```
#include <iostream>
#include <stack>
#include <map>
#include <cstdio>
using namespace std;

struct nd {
   nd *l,*r;
   char op;
   char a;
};
map<char,stack<nd*>> fresh;
int main() {
   int n;
   cout<<"Enter no of Expression: ";
   cin>>n;
   cout<<endl;
   char p[n],q[n],r[n],s[n];
   for(int i=0;i<n;i++) {
      char op,a1,a2,res;
      cout<<"Exp "<<i+1<<" : ";
      cin>>op>>a1>>a2>>res;
      cout<<endl;
      if(op=='=') {
         fresh[res] = fresh[a1];
      } else {
         nd *n1,*n2;
         bool fl=0;
         for (auto it:fresh) {
            bool b1=0,b2=0,b3=0;
            nd* tmpn=it.second.top();
            b1 = tmpn->op==op;
            if (tmpn->l!=NULL) b2=(tmpn->l->a==a1)&&(tmpn->l==fresh[tmpn->l->a].top());
            if (tmpn->r!=NULL) b3=(tmpn->r->a==a2)&&(tmpn->r==fresh[tmpn->r->a].top());
            if(b1&&b2&&b3) {
               fl=1;
               fresh[res] = fresh[tmpn->a];
               break;
            }
         if (fl==0) {
```

```
            if(fresh[a1].empty()) {
               n1 = new nd;
               n1 -> l = NULL;
               n1 -> r = NULL;
               n1 -> a = (a1);
               fresh[a1].push(n1);
            } else n1 = fresh[a1].top();
            if(fresh[a2].empty()) {
               n2 = new nd;
               n2 -> l = NULL;
               n2 -> r = NULL;
               n2 -> a = (a2);
               fresh[a2].push(n2);
            } else n2 = fresh[a2].top();
              nd *resn = new nd;
            resn -> l = n1;
            resn -> r = n2;
            resn -> op= (op);
            resn -> a = (res);
            fresh[res].push(resn);
            p[i]=op;
            q[i]=fresh[a1].top()->a
            ;
            r[i]=fresh[a2].top()->a;
            s[i]=res;
         }
      }
   }
   printf("After applying DAG in given expression: \n");
   for(int k=0; k<n;k++)
      if(q[k]>='a' && q[k]<='z')
      printf("%c %c %c %c\n",p[k],q[k],r[k],s[k]);
   return 0;
}
```

## Output:

```
Enter no of Expression: 3

Exp 1 : + a b x

Exp 2 : + x c y

Exp 3 : + x y d

After applying DAG in given expression:
+ a b x
+ x c y
+ x y d
```

**Result:** **The program for DAG generation is executed successfully.**