Lab 7: Write a program to generate ICG using LEX and YACC.

ICG.l

```
%{
#include<stdio.h>
#include "y.tab.h"
struct symtab *install_id(char *s);
%}
l [A-Za-z]
d [0-9]
id {l}({l}|{d})*
num {d}+(\.{d}+)?
%%
{id} {yylval.p=install_id(yytext); return id;}
{num} {strcpy(yylval.v,yytext); return num;}
[-+*/^=;] {return yytext[0];}
.|\n {}
%%
int yywrap()
{
return 1;
}
```

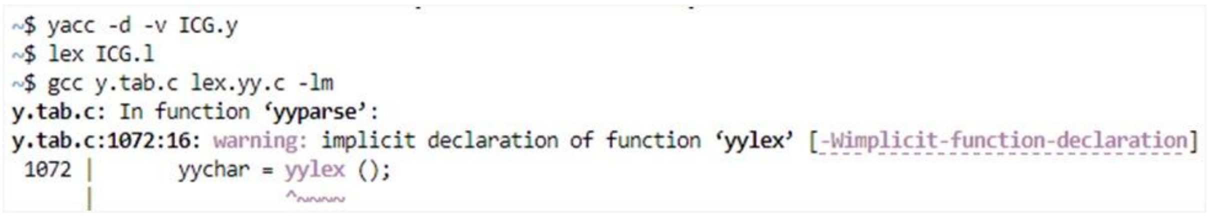
ICG.y

```
%{
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
void itoa1(int a,int k);
char* insert_quad(char op[10],char arg1[10],char arg2[10],int j);
struct symtab{
char *name;
double value;
}SYM[20];
struct symtab *install_id(char *s);

void display_sym();
typedef struct quadruple{
char op[10];
char arg1[10];
char arg2[10];
char res[10];
}QUAD;
QUAD Q[30];

void display_quad();
int i=0;
int tempvar=1;
```

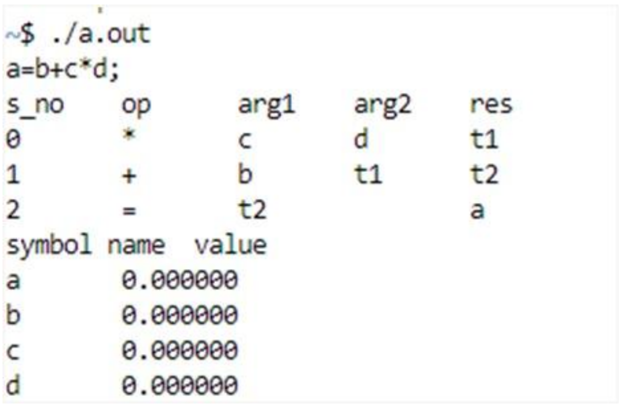
```
char temp[10],st[10];
%}
%union{
struct symtab *p;
char v[10];
}
%token<p> id
%token <v>num
%right '='
%left '+' '-'
%left '*' '/'
%right '^'
%nonassoc UMINUS
%type<v>E
%%
S: S OS
  |OS
  ;
OS:AS
;
AS:id'='E';'{strcpy(Q[i].op,"=");
   strcpy(Q[i].arg1,$3);
   strcpy(Q[i].arg2,"");
   strcpy(Q[i].res,$1->name);
   i++;}
;

E:E'+'E{strcpy($$,insert_quad("+",$1,$3,i));i++;}
|E'-'E{strcpy($$,insert_quad("-",$1,$3,i));i++;}
|E'*'E{strcpy($$,insert_quad("*",$1,$3,i));i++;}
|E'/'E{strcpy($$,insert_quad("/",$1,$3,i));i++;}
|E'^'E{strcpy($$,insert_quad("^",$1,$3,i));i++;}
|'-'E{strcpy($$,insert_quad("UMINUS",$2,"",i));i++;}
|id{strcpy($$,$1->name);}
|num{strcpy($$,$1);}
;
%%
char* insert_quad(char op[10],char arg1[10],char arg2[10],int j)
{

strcpy(Q[j].op,op);
strcpy(Q[j].arg1,arg1);
strcpy(Q[j].arg2,arg2);
strcpy(temp,"t");
itoa1(tempvar++,10);
strcat(temp,st);
strcpy(Q[j].res,temp);
return temp;
```

```
}

int yyerror(char *s)
{
printf("error=%s\n",s);
return 1;
}
main()
{
yyparse();
display_quad();
display_sym();
}
struct symtab *install_id(char *s)
{
struct symtab *p;
for(p=SYM;p<&SYM[20];p++)
{
if(p->name&&!strcmp(p->name,s))
return p;
else
if(!p->name)
{
p->name=strdup(s);
return p;
}
}
}

void display_sym()
{
struct symtab *p;
printf("symbol name  value\n");
for(p=SYM;p<&SYM[20];p++)
{
if(p->name)
printf("%s\t%lf\n",p->name,p->value);
}
}

void itoa1(int t,int b)
{
int j=0,k;
char m[10];
while(t!=0)
{
m[j]=t%b+48;
t=t/b;
j++;
}
m[j]='\0';

j=0;
for(k=strlen(m)-1;k>=0;k--)
st[j++]=m[k];
st[j]='\0';
}

void display_quad()
{
int j;
printf("s_no\top\targ1\targ2\tres\n");
for(j=0;j<i;j++)
printf("%d\t%s\t%s\t%s\t%s\n",j,Q[j].op,Q[j].arg1,Q[j].arg2,Q[j].res);
}
```

OUTPUT:



```
~$ yacc -d -v ICG.y
~$ lex ICG.l
~$ gcc y.tab.c lex.yy.c -lm
y.tab.c: In function ‘yyparse’:
y.tab.c:1072:16: warning: implicit declaration of function ‘yylex’ [-Wimplicit-function-declaration]
 1072 |       yychar = yylex ();
      |                ^~~~~
```



```
~$ ./a.out
a=b+c*d;
s_no    op      arg1    arg2    res
0       *       c       d       t1
1       +       b       t1      t2
2       =       t2              a
symbol name  value
a       0.000000
b       0.000000
c       0.000000
d       0.000000
```