LAB ASSIGNMENTS

1. YACC

File: ic.arithemtic.l

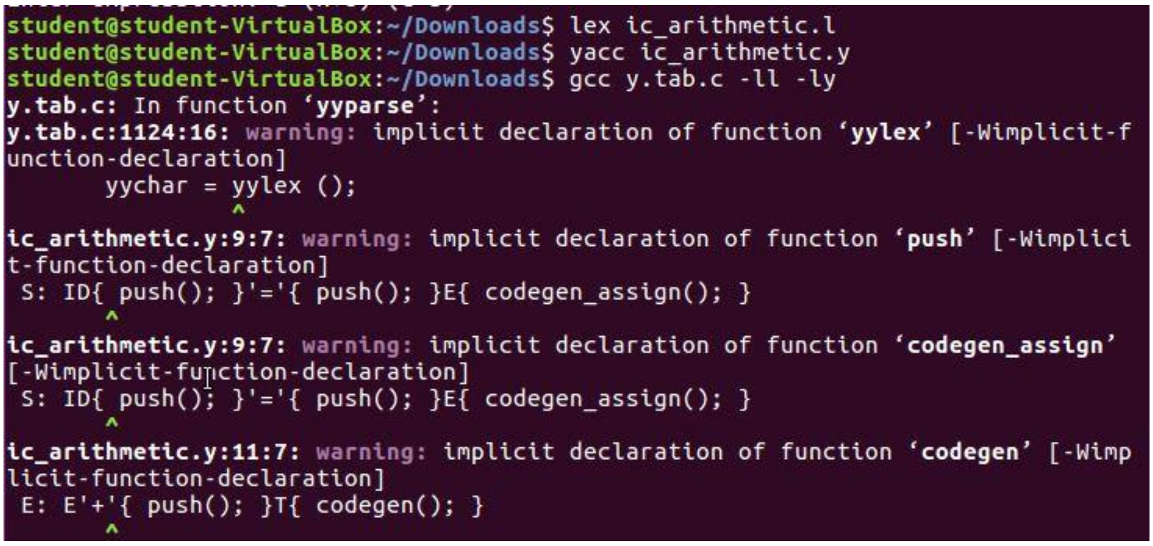
```
ALPHA [A-Za-z]
DIGIT [0-9]
%%
{ALPHA}({ALPHA}|{DIGIT})* return ID;
{DIGIT}+ {yylval = atoi(yytext); return NUM; }
[\n\t] yyterminate();
. return yytext[0];
%%
```

File: ic.arithemtic.l

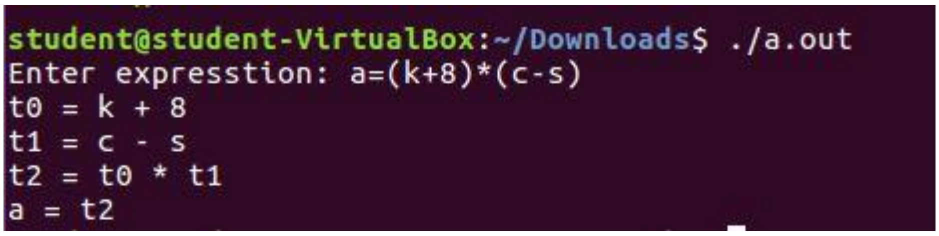
```
%token ID NUM
%right '='
%left '+' '-'
%left '*' '/'
%left UMINUS

%%

S: ID{ push(); }'='{ push(); }E{ codegen_assign(); }
 ;
E: E'+'{ push(); }T{ codegen(); }
 | E'-'{ push(); }T{ codegen(); }
 | T
 ;
T: T'*'{ push(); }F{ codegen(); }
 | T'/'{ push(); }F{ codegen(); }
 | F
 ;
F: '('E')'
 | '-'{ push(); }F{codegen_umin(); }%prec UMINUS
 | ID{ push(); }
 | NUM{ push(); }
 ;

%%

#include "lex.yy.c"
#include <ctype.h>

char st[100][10];
```

```
int top = 0;
char i_[2] = "0";
char temp[2] = "t";

int main()
{
	printf("Enter expresstion: ");
	yyparse();
}
void push()
{
	strcpy(st[++top], yytext);
}
void codegen()
{
	strcpy(temp, "t");
	strcat(temp, i_);
	printf("%s = %s %s %s\n", temp, st[top - 2], st[top - 1], st[top]);
	top -= 2;
	strcpy(st[top], temp);
	i_[0]++;
}
void codegen_umin()
{
	strcpy(temp, "t");
	strcat(temp, i_);
	printf("%s = -%s\n", temp, st[top]);
	top--;
	strcpy(st[top], temp);
	i_[0]++;
}
void codegen_assign()
{
	printf("%s = %s\n", st[top - 2], st[top]);
	top -= 2;
}
```

OUPUT

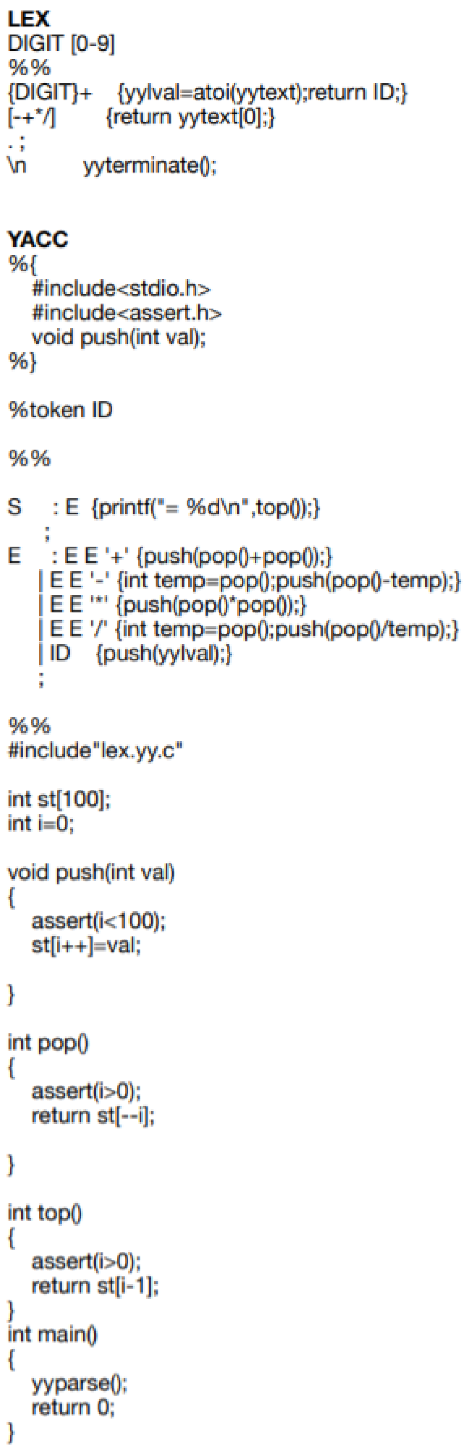


```
student@student-VirtualBox:~/Downloads$ lex ic_arithmetic.l
student@student-VirtualBox:~/Downloads$ yacc ic_arithmetic.y
student@student-VirtualBox:~/Downloads$ gcc y.tab.c -ll -ly
y.tab.c: In function ‘yyparse’:
y.tab.c:1124:16: warning: implicit declaration of function ‘yylex’ [-Wimplicit-f
unction-declaration]
       yychar = yylex ();
                ^
ic_arithmetic.y:9:7: warning: implicit declaration of function ‘push’ [-Wimplici
t-function-declaration]
 S: ID{ push(); }'='{ push(); }E{ codegen_assign(); }
       ^
ic_arithmetic.y:9:7: warning: implicit declaration of function ‘codegen_assign’
[-Wimplicit-function-declaration]
 S: ID{ push(); }'='{ push(); }E{ codegen_assign(); }
       ^
ic_arithmetic.y:11:7: warning: implicit declaration of function ‘codegen’ [-Wimp
licit-function-declaration]
 E: E'+'{ push(); }T{ codegen(); }
       ^
```



```
student@student-VirtualBox:~/Downloads$ ./a.out
Enter expresstion: a=(k+8)*(c-s)
t0 = k + 8
t1 = c - s
t2 = t0 * t1
a = t2
```

2. POSTFIX EVALUTION



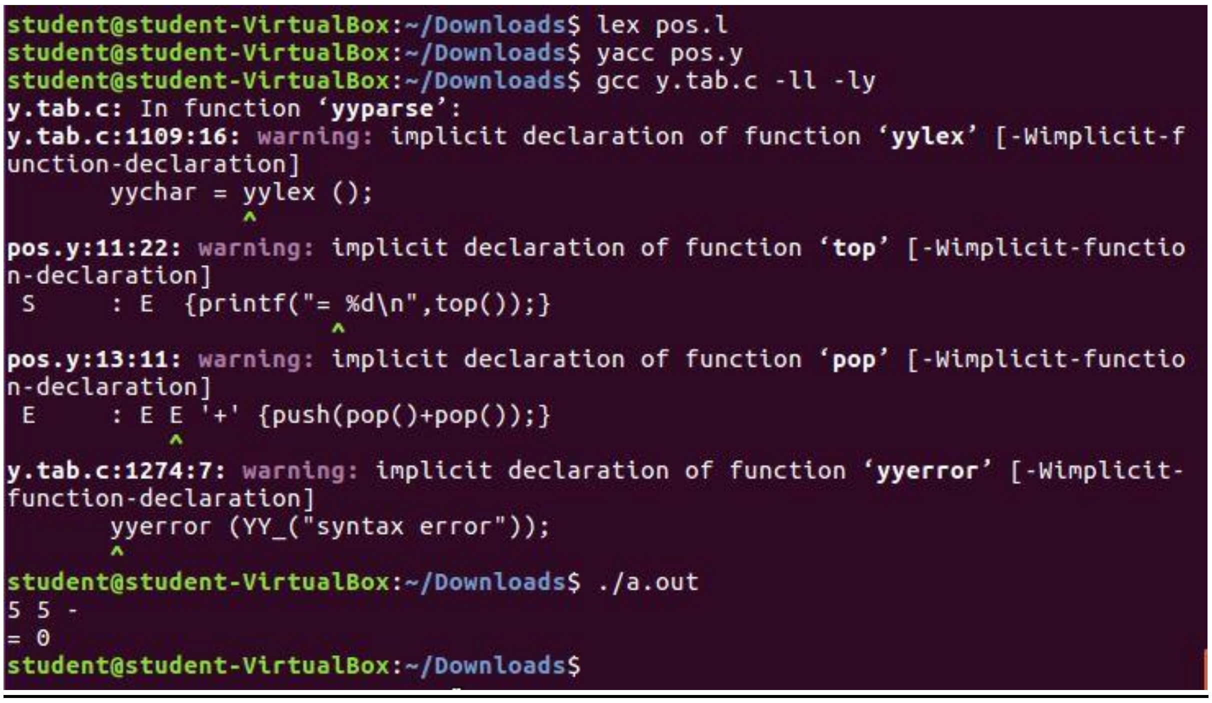
LEX

```
DIGIT [0-9]
%%
{DIGIT}+    {yylval=atoi(yytext);return ID;}
[-+*/]      {return yytext[0];}
. ;
\n        yyterminate();
```

YACC

```
%{
   #include<stdio.h>
   #include<assert.h>
   void push(int val);
%}

%token ID

%%

S    : E  {printf("= %d\n",top());}
     ;
E    : E E '+' {push(pop()+pop());}
     | E E '-' {int temp=pop();push(pop()-temp);}
     | E E '*' {push(pop()*pop());}
     | E E '/' {int temp=pop();push(pop()/temp);}
     | ID   {push(yylval);}
     ;

%%
#include"lex.yy.c"

int st[100];
int i=0;

void push(int val)
{
   assert(i<100);
   st[i++]=val;

}

int pop()
{
   assert(i>0);
   return st[--i];

}

int top()
{
   assert(i>0);
   return st[i-1];
}
int main()
{
   yyparse();
   return 0;
}
```

OUTPUT



```
student@student-VirtualBox:~/Downloads$ lex pos.l
student@student-VirtualBox:~/Downloads$ yacc pos.y
student@student-VirtualBox:~/Downloads$ gcc y.tab.c -ll -ly
y.tab.c: In function ‘yyparse’:
y.tab.c:1109:16: warning: implicit declaration of function ‘yylex’ [-Wimplicit-f
unction-declaration]
       yychar = yylex ();
                ^
pos.y:11:22: warning: implicit declaration of function ‘top’ [-Wimplicit-functio
n-declaration]
 S     : E  {printf("= %d\n",top());}
                      ^
pos.y:13:11: warning: implicit declaration of function ‘pop’ [-Wimplicit-functio
n-declaration]
 E     : E E '+' {push(pop()+pop());}
           ^
y.tab.c:1274:7: warning: implicit declaration of function ‘yyerror’ [-Wimplicit-
function-declaration]
       yyerror (YY_("syntax error"));
       ^
student@student-VirtualBox:~/Downloads$ ./a.out
5 5 -
= 0
student@student-VirtualBox:~/Downloads$
```

3. DESK CALCULATOR

LEX

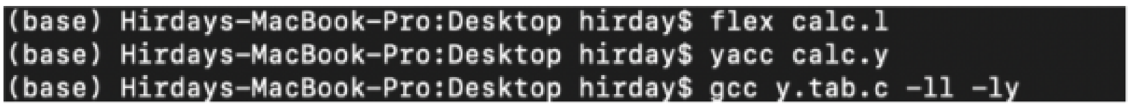
```
DIGIT [0-9]+\.?|[0-9]*\.[0-9]+

%%

[ ]
{DIGIT}    {yylval=atof(yytext);return NUM;}
\n|.           {return yytext[0];}
```

YACC

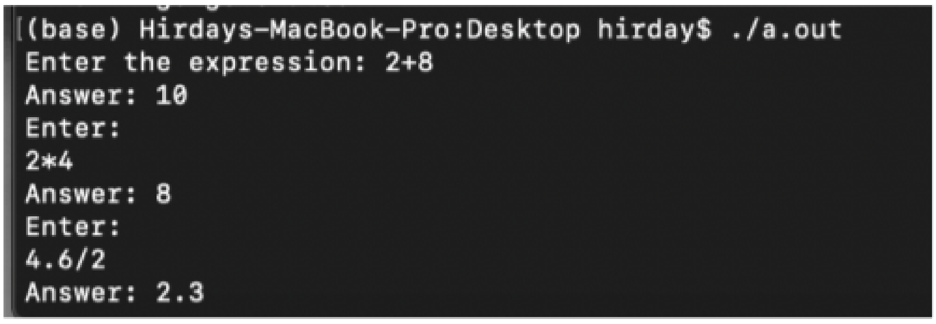
```
%{
   #include<ctype.h>
   #include<stdio.h>
   #define YYSTYPE double
%}

%token NUM

%left '+' '-'
%left '*' '/'
%right UMINUS

%%

S        : S E '\n' { printf("Answer: %g \nEnter:\n", $2); }
         | S '\n'
         |
         | error '\n' { yyerror("Error: Enter once more...\n" );yyerrok; }
         ;
E        : E '+' E    { $$ = $1 + $3; }
         | E'-'E    { $$=$1-$3; }
         | E'*'E    { $$=$1*$3; }
         | E'/'E    { $$=$1/$3; }
         | '('E')'    { $$=$2; }
         | '-'E %prec UMINUS { $$= -$2; }
         | NUM
         ;

%%

#include "lex.yy.c"
```

```
int main()
{
   printf("Enter the expression: ");
   yyparse();
}
```

OUTPUT



```
(base) Hirdays-MacBook-Pro:Desktop hirday$ flex calc.l
(base) Hirdays-MacBook-Pro:Desktop hirday$ yacc calc.y
(base) Hirdays-MacBook-Pro:Desktop hirday$ gcc y.tab.c -ll -ly
```



```
(base) Hirdays-MacBook-Pro:Desktop hirday$ ./a.out
Enter the expression: 2+8
Answer: 10
Enter:
2*4
Answer: 8
Enter:
4.6/2
Answer: 2.3
```