Compiler Construction Lab

Rishabh Kaushick, C3, 152

Intermediate Code Generation

Assignment7_postfix.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];

%%
```

Assignment7_postfix.y

```
%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()
```

```c
{
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;
}
```

Screenshots

Postfix Expression

Assignment8.l

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

Assignment8.y

```
%{
 #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)
{
```

```c
 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;
}
```

Screenshots


```
rishabh@Rishabh-Lenovo:~/Dropbox/University/Symbiosis/Academic/Semester 8/Compiler Construction$ lex Assignment8.l
rishabh@Rishabh-Lenovo:~/Dropbox/University/Symbiosis/Academic/Semester 8/Compiler Construction$
rishabh@Rishabh-Lenovo:~/Dropbox/University/Symbiosis/Academic/Semester 8/Compiler Construction$ yacc Assignment8.y
rishabh@Rishabh-Lenovo:~/Dropbox/University/Symbiosis/Academic/Semester 8/Compiler Construction$ gcc y.tab.c -ll -ly
y.tab.c: In function 'yyparse':
y.tab.c:1109:16: warning: implicit declaration of function 'yylex' [-Wimplicit-function-declaration]
        yychar = yylex ();
                 ^~~~~
Assignment8.y:8:22: warning: implicit declaration of function 'top' [-Wimplicit-function-declaration]
 S : E {printf("= %d\n",top());}
                        ^~~
Assignment8.y:10:11: warning: implicit declaration of function 'pop'; did you mean 'popen'? [-Wimplicit-function-declaration]
 E : E E '+' {push(pop()+pop());}
                  ^~~
                  popen
y.tab.c:1274:7: warning: implicit declaration of function 'yyerror'; did you mean 'yyerrok'? [-Wimplicit-function-declaration]
       yyerror (YY_("syntax error"));
       ^~~~~~~
       yyerrok
rishabh@Rishabh-Lenovo:~/Dropbox/University/Symbiosis/Academic/Semester 8/Compiler Construction$ ./a.out
15 20 +
= 35
rishabh@Rishabh-Lenovo:~/Dropbox/University/Symbiosis/Academic/Semester 8/Compiler Construction$ ./a.out
15 5 -
= 10
rishabh@Rishabh-Lenovo:~/Dropbox/University/Symbiosis/Academic/Semester 8/Compiler Construction$ ./a.out
5 5 *
= 25
rishabh@Rishabh-Lenovo:~/Dropbox/University/Symbiosis/Academic/Semester 8/Compiler Construction$
```
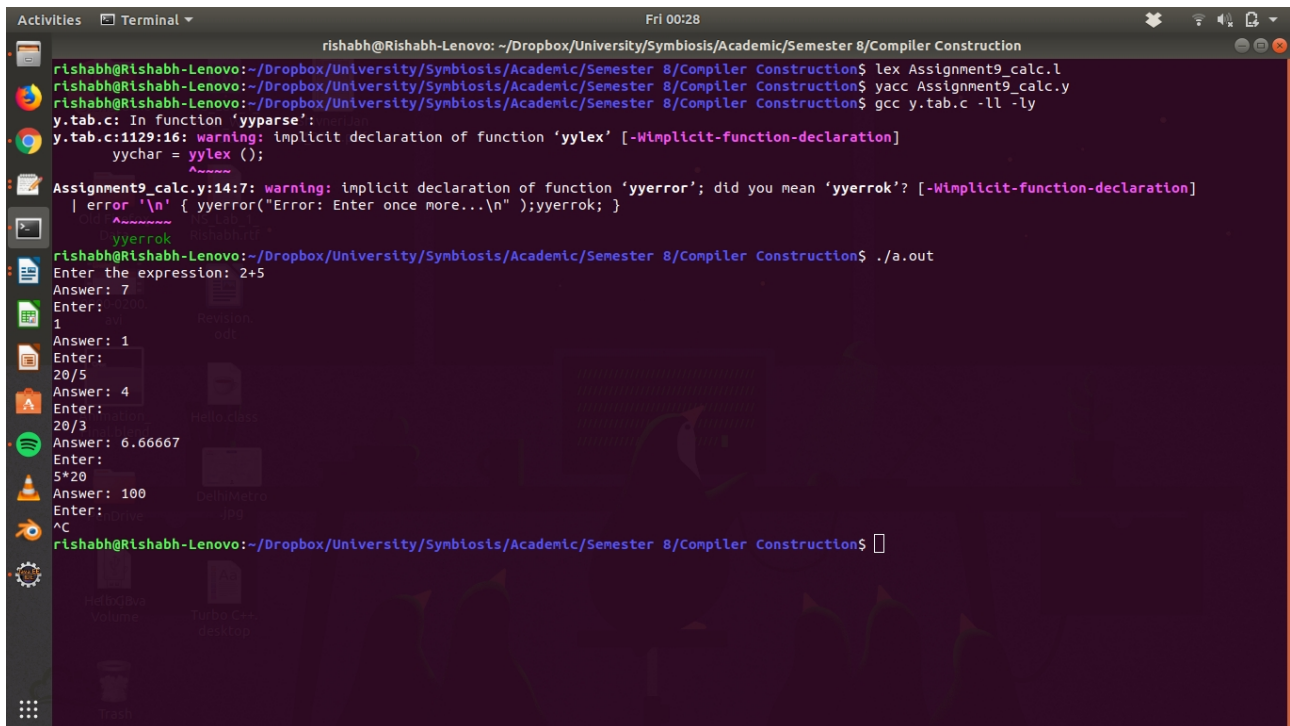
Calculator

Assignment9_calc.l

```
DIGIT [0-9]+\.?|[0-9]*\.[0-9]+
%%
[ ]
{DIGIT} {yylval=atof(yytext);return NUM;}
\n|. {return yytext[0];}
```

Assignment9_calc.y

```
%{
 #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();
}
```

Screenshot