Week-11

Generate quadruples for given arithmetic expression using LEX and YACC.

Quad.l code:

```
%{
#include<stdio.h>
#include "y.tab.h"
#include<string.h>
%}
%%
[a-z]([a-z]|[0-9])*   {       strcpy(yylval.exp,yytext);
                              return VAR;
                      }
\t      ;
\n      return 0;
.       return yytext[0];
%%
```

Quad.y code:

```
%{
#include<stdio.h>
#include<string.h>
struct quad
{
char op[5];
char arg1[10];
char arg2[10];
char result[10];
}QUAD[30];
int i=0,j;
%}
%union
```

```
{
char exp[10];
}
%token <exp> VAR
%type <exp> S E T F
%%
S: E
{
printf("\n There are %d quadrupls \n", i);
printf("\n List of Quadruples are: \n");
for(j=0;j<i;j++)
printf("%s\t%s\t%s\t%s\n",QUAD[j].op,QUAD[j].arg1,QUAD[j].arg2,QUAD[j].re
sult);
}
;
E: E'+'T { printf("\n E -> E+T, $1=%s, $3=%s, $$=%s\n",$1,$3,$$);
strcpy(QUAD[i].op,"*");
strcpy(QUAD[j].arg1,$1);
strcpy(QUAD[j].arg2,$3);
strcpy(QUAD[j].result,$$); i++;
i++;
}
| T { printf("\n E -> T, $1=%s, $$=%s\n",$1,$$);}
;
T: T'*'F { printf("\n T -> T*F, $1=%s, $3=%s, $$=%s\n",$1,$3,$$);
strcpy(QUAD[i].op,"*");
strcpy(QUAD[j].arg1,$1);
strcpy(QUAD[j].arg2,$3);
strcpy(QUAD[j].result,$$);
i++;
}
| F { printf("\n T -> F, $1=%s, $$=%s\n",$1,$$);}
;
F: VAR { printf("\n F -> VAR and $1=%s, $$=%s\n",$1,$$);}
;
```

```
%%
main()
{
yyparse();
}
int yywrap(){
return 1;
}
void yyerror(char *s)
{
printf("%s", s);
}
```

Output:

```
ubuntu@ubuntu:~$ lex quad.l
ubuntu@ubuntu:~$ yacc -d quad.y
ubuntu@ubuntu:~$ gcc lex.yy.c y.tab.c -w
ubuntu@ubuntu:~$ ./a.out
a+b*c
 F ->VAR and $1=a, $$=a
 T -> F, $1=a, $$=a
 E -> T,  $1=a,  $$=a
 F ->VAR and $1=b, $$=b
 T -> F, $1=b, $$=b
 F ->VAR and $1=c, $$=c
 T -> T*F, $1=b, $3=c, $$=b
 E ->E+T, $1=a, $3=b,$$=a
 THere are 3 quadrupls n
 List of Quadruples are:
*       b       c       b
+       a       b       a
```