Week 12 Assignment

Problem

Generate quadruples for given arithmetic expression using LEX and YACC

intermediate_code_generator.y

```
%{
#include"y.tab.h"
#include<stdio.h>
char addtotable(char,char,char);


int index1=0;
char temp = 'A'-1;


struct expr{

char operand1;
char operand2;
char operator;
char result;
};



%}

%union{
char symbol;
```

```
}



%left '+' '-'
%left '/' '*'

%token <symbol> LETTER NUMBER
%type <symbol> exp
%%

statement: LETTER '=' exp ';' {addtotable((char)$1,(char)$3,'=');};
exp: exp '+' exp {$$ = addtotable((char)$1,(char)$3,'+');}
    |exp '-' exp {$$ = addtotable((char)$1,(char)$3,'-');}
    |exp '/' exp {$$ = addtotable((char)$1,(char)$3,'/');}
    |exp '*' exp {$$ = addtotable((char)$1,(char)$3,'*');}
    |'(' exp ')' {$$= (char)$2;}
    |NUMBER {$$ = (char)$1;}
    |LETTER {(char)$1;};

%%

struct expr arr[20];

void yyerror(char *s){
    printf("Errror %s",s);
}
```

```
char addtotable(char a, char b, char o){
    temp++;
    arr[index1].operand1 =a;
    arr[index1].operand2 = b;
    arr[index1].operator = o;
    arr[index1].result=temp;
    index1++;
    return temp;
}

void threeAdd(){

    int i=0;
    char temp='A';
    while(i<index1){
        printf("%c:=\t",arr[i].result);
        printf("%c\t",arr[i].operand1);
        printf("%c\t",arr[i].operator);
        printf("%c\t",arr[i].operand2);
        i++;
        temp++;
        printf("\n");
    }
}

void fouradd(){
    int i=0;
    char temp='A';
    while(i<index1){
```

```
        printf("%c\t",arr[i].operator);
        printf("%c\t",arr[i].operand1);
        printf("%c\t",arr[i].operand2);
        printf("%c",arr[i].result);
        i++;
        temp++;
        printf("\n");
    }

}

int find(char l){
    int i;
    for(i=0;i<index1;i++)
        if(arr[i].result==l) break;
    return i;
}



void triple(){
    int i=0;
    char temp='A';
    while(i<index1){
        printf("%c\t",arr[i].operator);
        if(!isupper(arr[i].operand1))
        printf("%c\t",arr[i].operand1);
        else{
            printf("pointer");
```

```
            printf("%d\t",find(arr[i].operand1));
        }
        if(!isupper(arr[i].operand2))
        printf("%c\t",arr[i].operand2);
        else{
            printf("pointer");
            printf("%d\t",find(arr[i].operand2));
        }
        i++;
        temp++;
        printf("\n");
    }

}


int yywrap(){
    return 1;
}

int main(){
    printf("Enter the expression: ");
    yyparse();
    threeAdd();
    printf("\n");
    fouradd();
    printf("\n");
    triple();
    return 0;
```

}

intermediate_code_generator.l

```
%{
#include"y.tab.h"
extern char yyval;
%}

%%

[0-9]+ {yylval.symbol=(char)(yytext[0]);return NUMBER;}
[a-z] {yylval.symbol= (char)(yytext[0]);return LETTER;}
. {return yytext[0];}
\n {return 0;}

%%
```

}**Input & Output:**

```
Enter the expression: a=b*c+1/3-5*f;
A:= b * c
B:= 1 / 3
C:= A + B
D:= 5 * f
E:= C - D
F:= a = E

* b c A
/ 1 3 B
+ A B C
* 5 f D
- C D E
= a E F

* b c
/ 1 3
+ pointer0 pointer1
* 5 f
- pointer2 pointer3
= a pointer4
```