CS 442/542
Homework 3

“Due” Friday October 16

Homework 3

Implement a program that finds the tokens in a source file. The file will contain ten types of tokens: identifiers, integers, floats, +, -, *, /, (,), =. The file can also contain blanks, tabs, newlines and carriage returns. If the file contains unexpected characters an error message should be displayed.

An identifier is a letter followed by zero or more letters or digits.

An integer is a sequence of one or more digits

A float is a sequence of one or more digits followed by a decimal point followed by a sequence of one or more digits.

The program must create two symbol tables. The first symbol table will store the identifiers. For each identifier the symbol table must store a list of the line numbers on which each identifier appears. The second symbol table will store integers and floats. For each item on the second symbol table the program should keep track of the type of the item (integer or float) and the number of times the item appears in the input.

The program should keep track of the number of times each +, -, *, /, (,) and = is found in the file. No action is necessary for blanks, tabs, newlines or carriage returns.

Homework 3

After all the characters in the source file have been read, the program should print to standard out the following information. For each identifier print the identifier following by the line numbers on which the identifier appears in the file. For each identifier each line number should appear at most once. For each integer and float item print the item, its data type and the number of times it appears in the file. Print the number of times each +, -, *, /, (,), = appeared in the file (i.e. + 20).

Before the program stops all the dynamically allocated space must be freed.

The program must use symtab and iomngr (i.e. lex must get its input from iomngr). The program must used lex (flex) to recognize the tokens. You should create a .h file to define codes for the token types and a struct for the attribute structure for the second symbol table and a .c file for a driver program (to create the symbol tables, open files, call yylex, manage interaction with the symbol tables and the attribute structures for the symbol table entries). You will also need to create a list data structure to hold line numbers.

Homework 3 Example
.h file

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "SymTab.h"
#include "IOMngr.h"
```

```
extern char *yytext;
extern int yylex();

#define TRUE 1
#define FALSE 2
#define Ident 3

typedef struct {
    char *type;
    int count;
} Attribute;
```

Homework 3 Example
lex File

```
%{
    #include "h3Example.h"

    #define YY_INPUT(buf,result,max_size) \
    { int c = getNextSourceChar(); \
    result = (c == EOF) ? YY_NULL : (buf[0] = c, 1); \
    }

%}


letter [A-Za-z]
digit [0-9]
```

Homework 3 Example lex File

```
%%

true                                {return TRUE;}
false                               {return FALSE;}
{letter}({letter}|{digit})*         {return Ident;}
[ ]                                 {}
\t                                  {}
\r                                  {}
\n                                  {}

.                                   {
                                        writeIndicator(getCurrentColumnNum());
                                        writeMessage("Illegal Character in lex");
                                    }


%%

int yywrap () {
    return 1;
}
```

Homework 3 Example
main.c

```
int main(int argc, char * argv[]) {

    table = createSymTab(17);
    if (!openFiles(argv[1], "listing")) {
        printf("open failed\n");
        exit(0);
    }
    readTokens();
    printSymTab();

}
```

Homework 3 Example
main.c

```c
#include "h3Example.h"

SymTab *table;
char boolean[] = "Boolean";
char identifier[] = "Identifier";

void printSymTab() {
    int hasMore;
    //printf("in print symtab\n");
    //struct SymEntry *entry = FirstEntry(table);
    hasMore = startIterator(table);
    printf("%20s\t%10s\t%10s\n", "Name", "Type", "Count");
    while (hasMore) {
        Attribute * a = getCurrentAttr(table);
        printf("%20s\t%10s\t%10d\n",getCurrentName(table), a->type ,a->count);
        hasMore = nextEntry(table);
    }
}
```

Homework 3 Example
main.c

```
void readTokens() {
    Attribute *a;
    int token;
    while ((token = yylex())) {
        int new = enterName(table, yytext);
        if (new) {
            a = (Attribute *) malloc(sizeof(Attribute));
            if (token == 3)
                a->type = strdup(identifier);
            else
                a->type = strdup(boolean);
            a->count = 1;
            setCurrentAttr(table, a);
        } else {
            a = getCurrentAttr(table);
            a->count++;
        }
    }
}
```

Homework 3 Example Build

- lex h3ExampleLex.l

- cc -o h3 lex.yy.c main.c SymTab.c IOMngr.c