lex 2

lex File Format

```
%{
      C Code
%}

Regular expression (RE) definitions

name      RE

%%

Actions associated with each regular expression

RE   Action

%%

C Code
```

lex2a.l

```
%{
#include <stdio.h>

int numIds = 0;
int numInts = 0;
int numLines = 0;
%}

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

%%

{letter}({letter}|{digit})*  {numIds++;}
{digit}+                    {numInts++;}
\n                          {numLines++;}
[ \r\t]                      { }

%%

int yywrap() {

  printf("The number of identifiers is %d\n", numIds);
  printf("The number of integers is %d\n", numInts);
  printf("The number of lines is %d\n", numLines);
  return 1;
}
```

lex2b.l

```
%{
#include <stdio.h>
#define IDENT 1
#define INTEGER 2
#define LINE 3

%}

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

%%

{letter}({letter}|{digit})*  {return IDENT;}
{digit}+                    {return INTEGER;}
\n                          {return LINE;}
[ \r\t]                     { }

%%
```

lex2b.l

```
int main(int argh, char * argv[]) {
  int numIds = 0;
  int numInts = 0;
  int numLines = 0;

  int token;
  while ((token = yylex())) {
    if (token == IDENT) numIds++;
    else if (token == INTEGER) numInts++;
    else numLines++;
  }
  printf("The number of identifiers is %d\n", numIds);
  printf("The number of integers is %d\n", numInts);
  printf("The number of lines is %d\n", numLines);
}

int yywrap() {

  return 1;
}
```

lex2c.l

```
%{
#include <stdio.h>
#include "IOMngr.h"
#define YY_INPUT(buf,result,max_size) \
{ int c = getNextSourceChar(); \
      result = (c == EOF) ? YY_NULL : (buf[0] = c, 1); \
    }

int numIds = 0;
int numInts = 0;
int numLines = 0;

%}

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

%%

{letter}({letter}|{digit})*  {numIds++;}
{digit}+            {numInts++;}
\n                  {numLines++;}
[ \r\t]                    { }

%%
```

lex2c.l

```
int main(int argc, char * argv[]) {
    openFiles(argv[1], argv[2]);
    yylex();
}

int yywrap() {

  printf("The number of identifiers is %d\n", numIds);
  printf("The number of integers is %d\n", numInts);
  printf("The number of lines is %d\n", numLines);
  return 1;
}
```

lex2d.l

```
%{
#include <stdio.h>
#include “IOMngr.h”
#define IDENT 1
#define INTEGER 2
#define LINE 3
#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]

%%

{letter}({letter}|{digit})*  {return IDENT;}
{digit}+                    {return INTEGER;}
\n                           {return LINE;}
[ \r\t]                      { }

%%
```

lex2d.l

```
int main(int argc, char * argv[]) {
  int numIds = 0;
  int numInts = 0;
  int numLines = 0;

  int token;
  openFiles(argv[1], argv[2]);
  while ((token = yylex())) {
    if (token == IDENT) numIds++;
    else if (token == INTEGER) numInts++;
    else numLines++;
  }
  printf("The number of identifiers is %d\n", numIds);
  printf("The number of integers is %d\n", numInts);
  printf("The number of lines is %d\n", numLines);
}

int yywrap() {

  return 1;
}
```

lex2e.l

```
%{
#include <stdio.h>
#include "SymTab.h"
#define IDENT 1
#define HEX 2
extern int yylex();
extern char *yytext;
typedef struct {
    int count;
} Attr;
%}

letter  [a-z]
digit   [0-9]|[A-F]

%%

{letter}+   {return IDENT;}
{digit}+    {return HEX;}

[ \r\n\t]   {}

%%
```

lex2e.l

```
int main(int argc, char *argv[]) {
    SymTab *table = createSymTab(17);
    int more;
    int newToken;
    int token;
    Attr *a;
    while ((token = yylex())) {
        newToken = enterName(table, yytext);
        if (newToken) {
            a = (Attr *) malloc(sizeof(Attr));
            a->count = 1;
            setCurrentAttr(table, a);
        } else {
            a = getCurrentAttr(table);
            a->count++;
        }
    }
    more = startIterator(table);
    while (more) {
        printf("%s\t%d\n", getCurrentName(table), (a = getCurrentAttr(table))->count);
        free(a);
        more = nextEntry(table);
    }
    destroySymTab(table);
}

int yywrap() {

    return 1;
}
```

lex2f.l

```
%{
#include <stdio.h>
#include “SymTab.h"
#include "IOMngr.h"
#define IDENT 1
#define HEX 2
#define YY_INPUT(buf,result,max_size) \
{ int c = getNextSourceChar(); \
      result = (c == EOF) ? YY_NULL : (buf[0] = c, 1); \
    }
extern int yylex();
extern char *yytext;
typedef struct {
    int type;
    int count;
} Attr;
%}

letter  [a-z]
digit   [0-9]|[A-F]

%%

{letter}+   {return IDENT;}
{digit}+    {return HEX;}

[ \r\n\t]   {}

%%
```

lex2f.l

```
int main(int argc, char *argv[]) {
    SymTab *table = createSymTab(17);
    openFiles(argv[1], argv[2]);
    int more;
    int newToken;
    int token;
    Attr *a;
    while ((token = yylex())) {
        newToken = enterName(table, yytext);
        if (newToken) {
            a = (Attr *) malloc(sizeof(Attr));
            a->type = token;
            a->count = 1;
            setCurrentAttr(table, a);
        } else {
            a = getCurrentAttr(table);
            a->count++;
        }
    }
    more = startIterator(table);
    while (more) {
        printf("%s\t%s\t%d\n", getCurrentName(table),
                               ((Attr *) getCurrentAttr(table))->type == IDENT ? "Identifier" : "Hexadecimal",
                               ( a = getCurrentAttr(table))->count);
        free(a);
        more = nextEntry(table);
    }
    destroySymTab(table);
}

int yywrap() {

    return 1;
}
```