COP4020
Assignment 2
cxref - lexical analysis for a C cross reference listing

- *cxref* performs lexical analysis to produce a cross reference listing of a C file. After processing the file, all identifiers that are not reserved words will be printed in alphabetical order. Beside each identifier will be the list of line numbers in which the identifiers appear. If an identifier appears in a line more than once, then only list that line number once. Note that identifiers in comments, strings, character literals, or preprocessor commands should be ignored.
- You should not make any assumptions about the number of identifiers or the number of lines on which each identifier can appear. You should create a single lex (flex) file to generate the cross reference listing. You should have lex rules to detect the following cases.

  1) start of a C comment (either multi-line comment with **/*** or line comment with **//**)
  2) end of a C comment
  3) start or end of a string
  4) start or end of a character literal
  5) list of reserved words
  6) newline
  7) identifier
  8) C preprocessor command (assume it starts at the beginning of the line and does not span multiple lines)
  9) default character

  The list of C reserved words *are given below.*

auto	**int**	
break	**long**	
case	**register**	
char	**return**	
const	**short**	
continue	**signed**	
default	**sizeof**	
do	**static**	
double	**struct**	
else	**switch**	
extern	**typedef**	**enum**
float	**union**	
for	**unsigned**	
goto	**void**	**volatile**
if	**while**	

- You should comment your program so that others (e.g. the grader) can understand it. You should also have comments at the top of the file indicating your name, this course, and the assignment. You have to implement this assignment in C/C++ so it will be compatible with lex (flex). Create a file called *cxref.l*. To process your lex (flex) specification you can simply issue the command:

```
lex cxref.l  (or flex cxref.l)
```

- You also need to link this file with the lex (or flex) library as done with the following command.

```
gcc -g -o cxref lex.yy.c -ll
```

or

```
gcc -g -o cxref lex.yy.c -lfl
```

- The file ~uh/cop4020/proj2/sample.c on linprog.cs.fsu.edu contains an example C code as shown below.

```
static char *sccsid = "@(#)echo.c4.1 (Berkeley) 10/1/80";
#include <stdio.h>

main(argc, argv)
int argc;
char *argv[];
{
      register int i, nflg;

      nflg = 0;
      if(argc > 1 && argv[1][0] == '-' && argv[1][1] == 'n')
            { nflg++;
            argc--;
            argv++;
      }
      for(i=1; i<argc; i++) {
            fputs(argv[i], stdout);
            if (i < argc-1)
                  putchar(' ');
      }
      if(nflg == 0)
            putchar('0);
      exit(0);
}
```

- The following output will appear when the above program is input to ~uh/cop4020/proj2/cxref.exe on linprog.cs.fsu.edu as follows.

```
% ~uh/cop4020/proj2/cxref.exe < ~uh/cop4020/proj2/sample.c

    argc: 4, 5, 11, 13, 16, 18
    argv: 4, 6, 11, 14, 17
    exit: 23
   fputs: 17
       i: 8, 16, 17, 18
    main: 4
    nflg: 8, 10, 12, 21
 putchar: 19, 22
  sccsid: 1
  stdout: 17
```

- Your output should match my format exactly so TAs and I can check your cxref program for correctness using the diff Unix command.
- You *MUST* implement this assignment in flex. The generated C program should be compiled and should be executed on linprog.cs.fsu.edu.
- As you did for the proj1 submission, you should submit your source file (*cxref.l*) and *makefile* as a single tar (or zip) file to the *Canvas* course website.
- The late submission will be penalized 10%