Ahsanullah University of Science and Technology
Department of CSE
Fall 2024
Course Name: Formal Language and Compilers Lab Course No: CSE 4130
Assignment 2

Write a C/C++ program that reads a C source program which has already been scanned and filtered as done in Assignment 1 (with comments removed and unnecessary spaces cleaned). This filtered code will be used as the input for lexical analysis.

Step 1: First, separate the lexemes. Make sure to correctly distinguish two-character operators (==, <=, >=, !=, etc.) along with other elements like separators, one character operators, parentheses, brackets, numeric constants, alphanumeric strings with or without underscore (i.e., identifiers). **Try to handle as many cases as possible, including different keywords, relational and arithmetic operators, separators, and other token types commonly found in C programs.**

```
char c ; int x1 , x_2 ; float y1 , y2 ; x1 = 5 ; x_2 = 10 ; y1 = 2.5 + x1 * 45 ; y2 = 100.o5 - x_2 / 3 ; if ( y1 <= y2 ) c = ' y ' ; else c = ' n ' ;
```

Step 2: Tokenize the separated lexemes by assigning them to appropriate categories — kw for keywords, id for identifiers, and so on. **Make sure to label any lexeme that does not fit into a known category as unkn (unknown).**

```
[kw char] [id c] [sep ;] [kw int] [id x1] [sep ,] [id x_2] [sep ;] [kw float] [id y1] [sep ,] [id y2] [sep ;] [id x1] [op =] [num 5] [sep ;] [id x_2] [op =] [num 10] [sep ;] [id y1] [op =] [num 2.5] [op +] [id x1] [op *] [num 45] [sep ;] [id y2] [op =] [unkn 100.o5] [op -] [id x_2] [op /] [num 3] [sep ;] [kw if] [par (] [id y1] [op <=] [id y2] [par )] [id c] [op =] [sep '] [id y] [sep '] [sep ;] [kw else] [id c] [op =] [sep '] [id n] [sep '] [sep ;]
```

Note that, **100.o5** does not fall under any valid tokens, so it is labeled as **unkn**.

Your code must have:

1. Scanned and filtered C source code as the input file
2. Two output files: one for Step 1 and one for Step 2
3. DFA implementation for tokenizing identifiers and numeric constants **(this is mandatory; otherwise marks will be deducted)**
4. DFA implementation for other token types is optional — you can tokenize those using any suitable method you prefer.
5. Display the input and both outputs (Step 1 and Step 2) on the console
6. Attach a screenshot of your output file named with your ID

Ahsanullah University of Science and Technology
Department of CSE
Fall 2024
Course Name: Formal Language and Compilers Lab Course No: CSE 4130
Assignment 2

Sample Input and Output:

Step 1 and Step 2 are the expected outputs for Assignment 2.

```
Sample Input: input.c
char c; int x1, x_2; float y1, y2; x1=5; x_2= 10; y1=2.5+x1*45; y2=100.o5-x_2/3; if(y1<=y2) c='y'; else c='n';

Sample output:

Step 1:
char c ; int x1 , x_2 ; float y1 , y2 ; x1 = 5 ; x_2 = 10 ; y1 = 2.5 + x1 * 45 ; y2 = 100.o5 - x_2 / 3 ; if ( y1 <= y2 )
 c = ' y ' ; else c = ' n ' ;


Step 2:
[kw char][id c][sep ;][kw int][id x1][sep ,][id x_2][sep ;][kw float][id y1][sep ,][id y2][sep ;][id x1][op =][num 5][se
p ;][id x_2][op =][num 10][sep ;][id y1][op =][num 2.5][op +][id x1][op *][num 45][sep ;][id y2][op =][unkn 100.o5][op -
][id x_2][op /][num 3][sep ;][kw if][par ([id y1][op <=][id y2][par )][id c][op =][sep '][id y][sep '][sep ;][kw else][
id c][op =][sep '][id n][sep '][sep ;]
```