CMPS 258 – PROGRAMMING LANGUAGES – SPRING 2021
WEEK 9 ASSIGNMENT

This assignment may be done in groups of two.

In this assignment, you will implement a tool for gathering statistics about SML code. Rather than implementing your tool from scratch, you will implement it as a lexer, using a lexer generator. Instead of the lexer passing tokens to a parser, it will gather statistics about each token and print a summary at the end.

To implement your lexer, follow the following steps:

1. Download and install *flex*, a free and open source lexer generator. On some Linux systems, you can install it using your package manager (e.g., `sudo apt-get install flex`). For Windows systems, you can obtain it this link: http://gnuwin32.sourceforge.net/packages/flex.htm.
2. Edit the provided file `stat.flex` to implement your lexer. A rule for the `datatype` keyword is provided for you as an example. Edit the file to implement the additional rules. In particular, you need to support the following subset of SML's lexical categories:
    a. Keywords for some language constructs: these include the words datatype, of, val, fun, fn, let, in, end, if, then, else, orelse, andalso, case. These can be used to gather statistics about the corresponding language constructs.
    b. Keywords for built-in types: these include the keywords int, bool, string. You do not need to collect statistics about these keywords, but you need to recognize them (i.e., accept code the uses them).
    c. Operators: these include:
        i. Arithmetic operators (or keywords for arithmetic operations): "+", "-", "*", div, mod. You can consider all occurrences of "*" to be arithmetic operators. In practice, "*" is also used to write tuple types, but these occurrences cannot be distinguished by a lexer and require parsing.
        ii. Relational operators: "<=", ">=", "<", ">", "=".
        iii. "Other" operators: "::", "=>", "|". You do not need to collect statistics about these keywords, but you need to recognize them (i.e., accept code the uses them).
    d. Separators: these include: ":", ".", "[", "]", ",", "(", ")", ";".
    e. Constants: these include:
        i. Integer constants which may contain an arbitrary number of numeric characters
        ii. Boolean constants which may be the keywords `true` or `false`
        iii. String Constants which start and end with double-quotes and may contain any number of characters. You must support escaped double-quote characters (\"). (Hint: Do not use regular expressions to match the entire string constant. Instead, match a double quote and use the action to read the entire string until you reach the next unescaped double-quote character. You can call the built-in `input()` function to read the next character in the stream. You will need to surround your action with curly braces { } since it will need to contain control flow constructs.)
    f. Identifiers: which may start with any alphabetical character or underscore, and may contain any number of alphabetical characters, numeric characters, underscores, and apostrophes.
    g. Comments: which start with "(*", end with "*)", and may contain any number of characters in between. You do not need to support nested comments. (Hint: Do not use regular expressions to match the entire comment. Instead, match "(*" and use the action

to read the entire comment until you reach "*)". You can call the built-in `input()` function to read the next character in the stream. You can call the built-in `unput(char)` function to return a character to the stream after reading it. You will need to surround your action with curly braces { } since it will need to contain control flow constructs.)

   h. Whitespace: which includes space, tab (\t), new line (\n), and carriage return (\r). New lines are useful for counting the number of lines while the remaining types of whitespace should be recognized and ignored.

3. When you are done editing `stat.flex`, use flex to generate the lexer code by executing the following command in the command line:

   ```
   flex stat.flex
   ```

   You may need to provide the full path to where you installed flex on your computer. This command will generate a C file called `lex.yy.c` that contains the implementation of your lexer.
4. Compile `lex.yy.c` using any C compiler of your choice. The generated binary is your code statistics tool.
5. Test your program. The generated binary takes the input file name as an argument (`test.sml` by default). You are provided with the following reference files to help test your tool: `test-reference.sml` and `test-reference.txt`. Executing your tool with `test-reference.sml` as input should generate output identical to that in `test-reference.txt`.

Submission Instructions

Submit your modified `stat.flex` file via Moodle. Do not submit any other files or compressed folders. Make sure to include a comment in the file with your name and AUBnet ID (e.g., abc01). If you did the assignment in a group of two, only one group member should submit. In this case, make sure to include both group members' names and AUBnet IDs in the file.