American University of Beirut
Department of Computer Science
CMPS 231 –Programming Languages

Spring 2022/2023

Asst 4: Due on Mar. 21 at midnight

In this assignment, you will implement a syntax highlighter for SML. Rather than implementing your syntax highlighter from scratch, you will implement it as a lexer, using a lexer generator. Instead of the lexer passing tokens to a parser, it will print code to an HTML file where each token is formatted based on its lexical category.

To implement your lexer, follow the following steps:

1. Download and install *flex*, a free and open source lexer generator. You can download flex at the following link: http://gnuwin32.sourceforge.net/packages/flex.htm. Click on "Setup" next to "Complete package, except sources" to download the setup file and execute it.
2. Edit the file `asst4.flex` to implement your lexer. A rule for the *val* 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 should be colored dark yellow (rgb=#a0a000). These include the keywords: datatype, of, val, fun, let, in, end, if, then, else, orelse, andalso, case.
    b. Keywords for built-in types should be colored green (rgb=#00c000). These include the keywords: int, bool, string.
    c. Literals should be colored red (rgb=#ff0000). These include:
        i. Integer literals: may contain an arbitrary number of numeric characters.
        ii. Boolean literals: true, false
        iii. String Constants: 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.)
    d. Identifiers should be colored magenta (rgb=#ff00ff) if they start with an upper case letter, and not be colored if they start with a lower case letter. Identifiers may start with any alphabetical character or underscore, and may contain any number of alphabetical characters, numeric characters, or underscore.
    e. Operators and separators should be colored blue (rgb=#0000ff). These include: "+", "-", "*", "|", "=>", "<=", ">=", "=", "<", ">", "::", ":", ".", "[", "]", ",", "(", ")", ";".
    f. Comments should be colored light blue (rgb=#00aaff). Comments start with "(*", ends 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.)
    g. Whitespace should be replaced with appropriate HTML code that ensures that it is properly displayed and not ignored. These include: space, tab (\t), new line (\n), and carriage return (\r). Tab should be treated as four consecutive spaces.

3. When you are done editing `asst4.flex`, use flex to generate the lexer code by executing the following command in the command line:

   `flex asst4.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 syntax highlighter.
5. Test your program. The generated binary takes the input file name as the first argument (`test.sml` by default) and the output file name as the second argument (`test.html` by default). You are provided with the following reference files to help test your tool: `test.sml`, `test-reference.html`. Executing your tool with `test.sml` as input should generate a file identical to `test-reference.html`.

Academic Integrity Policy

This assignment falls under the policy for Academic Integrity that is listed in the course syllabus. The policy will be strictly applied with no exceptions. The policy is listed again here as a reminder:

> *"Every student is expected to abide by the AUB Student Code of Conduct which is to be considered as part of this syllabus. Students are encouraged to study together, however each student must individually prepare their assignment solutions. Students may not receive any help during exams in any form including from human, written, or electronic sources. Cheating and plagiarism are not permitted and will be strictly penalized. The first violation of academic integrity will result in a zero on the assignment/exam of concern. The second violation will result in a zero in the course and will be reported to the Dean."*

You may not post or share part of or all of this assignment with anyone (let alone posting it publicly). Doing this will be considered as a violation of the academic integrity.

Submission Instructions

Submit your modified `asst4.flex` file via Moodle. Do not submit any other files or compressed folders.