INF400 2023-2024 Fall Semester

Burak Arslan

Homework I

1 Regular expression (10 pts)

What is the regexp that corresponds to the DFA in L03S14?

Write a python script that prints whether the following strings match your regexp:

- `0.1`
- `0.`
- `.`
- `0`
- `0.`
- `.1`

Sample output:

```
'0.1': True
'0.':  False
'.':   False
'0':   False
'0.':  False
'.1':  True
```

2 Lexer (95 pts)

As part of this homework, you are asked to implement a lexical analyzer for our own language, kiraz, as the first stage of the course project.

To ease the pains of starting out with a greenfield project, a `CMake`-based project with needed boilerplate and a base class named `Token` will be provided.

- You will use either `C++` with the provided package or `C` with your own boilerplate.
- You are supposed to work on Linux. Working on other platforms could (be made to) work but you will be on your own.
- Using `CMake` is **mandatory** though.

Your lexer will have to tokenize the following kiraz module correctly:

```
import io

func main() -> int32 {
    let text: string = "Hello world!\n";
    io.print(text);
    return 0;
}
```

The correct token array for the above code is as follows:

```
KW_IMPORT IDENTIFIER(io) OP_NEWLINE
KW_FUNC IDENTIFIER(main) OP_LPAREN OP_RPAREN OP_RETURNS
                                IDENTIFIER(int32) OP_LBRACE OP_NEWLINE
IDENTIFIER(let) IDENTIFIER(text) OP_COLON
      IDENTIFIER(string) OP_ASSIGN L_STRING("Hello world!\n")
                                                OP_SCOLON OP_NEWLINE
IDENTIFIER(io) OP_DOT IDENTIFIER(print) OP_LPAREN
                      IDENTIFIER(text) OP_RPAREN OP_SCOLON OP_NEWLINE
IDENTIFIER(return) L_INTEGER(0, 10) OP_SCOLON OP_NEWLINE
OP_RBRACE OP_NEWLINE
```

Please note that indented lines are continuations of previous lines. So the output above is actually 6 lines long.

While the above is almost a pangram for the kiraz lexicon, additional test cases will be provided.

Also note that lexers generated by `flex` simply copy unrecognized tokens to standard output. This is not acceptable – your lexer needs to recognize everyting and just deal with it. More specifically, it needs to reject invalid tokens and exit with code 3 (ie the main function must return 3). Any sort of recovery is out of scope[1] of this project, so the lexer is supposed to terminate on first unrecognized token. You must also print the rejected string.

> *Hint:* You may need a `REJECTED` token type. The main function may need to be modified to accomodate it.

Again, DO NOT ATTEMPT TO RECOVER. Just bail out.

Token Definitions

- **Alphabet**: For the time being, 26 letters of the english alphabet (both upper and lower case), 10 arabic digits, underscore and the symbols: `{}()+-/<=>`. Anything else needs to be explicitly rejected by the lexical analyzer.
- **Identifiers**: They start with a letter or an underscore, and continue with a letter, digit or underscore.
- **Integer Literals**: At least one digit.
- **Strings Literals**: Anything between double quotes (").

[1] fr. Hors sujet

- **Operators**:

OP_RETURNS	->
OP_EQUALS	==
OP_ASSIGN	=
OP_GT	>
OP_LT	<
OP_GE	>=
OP_LE	<=
OP_LPAREN	(
OP_RPAREN	)
OP_LBRACE	{
OP_RBRACE	}
OP_PLUS	+
OP_MINUS	-
OP_MULT	*
OP_DIVF	/
OP_COMMA	,
OP_NEWLINE	\n
OP_COLON	:
OP_SCOLON	;
OP_DOT	.

- **Keywords**:

KW_IMPORT	import
KW_FUNC	func
KW_IF	if
KW_WHILE	while
KW_CLASS	class

Submission Format

You are expected to turn in a zip file that contains:

- A file named `sol1.py` that contains the expected python script.
- A directory named `sol2` that contains the lexical analyzer. You are supposed to let your code do the talking for you, but if you feel like additional notes would help, you can put one of `hw1.{md,pdf}` inside.

頑張ってください。