CS210 - Programming Languages
Lab Assignment #1
Fall 2020

The purpose of this assignment is to develop the first part of a language processor, sometimes called the *scanner*. Note that for the next several assignments you will be building on previous assignments, eventually creating an entire *lexical analyzer* (or *lexer*). The language we will be analyzing is called CCX, which is a C-like language that is used for programming embedded systems. You don't really need to know anything about this language - however, since its syntax is similar to C, you can probably figure out the meaning of a program written in CCX. An example CCX program is provided below - this example can be used as a test input for your scanner.

The first step in the lexical analysis process is to take the source code input file specified on the command line and separate it into *lexemes* - the fundamental components that make up the language. Your program should open the input file, then print out each individual lexeme on a separate line on the output. Your program can assume that the input is a correctly written CCX program - we won't worry about syntactically incorrect programs until a later assignment.

Your program should be written in C (not C++). Since you will want your program to do its own parsing, your program should use C's raw I/O system calls to perform the file opening and input (similar to your assignment 0).

A sample CCX program is shown below. This program simply prints the string `"Hello, world"` to the screen and then prints the arguments to the program if any were provided. You have probably seen similar programs like this before.

```
/*
 * Hello world with args.
 */
procedure main(argc: integer; argv: string_vector_type) is
begin
    printf("Hello, world\n");
    loop
        argc := argc - 1;
        exit when (argc = 0);
        printf("arg[%d]: %s\n", argc, argv[argc]);
    end loop;
end main;
```

Figure 1: CCX Sample 1

The output that your program produces while processing this file should look *exactly* like that below.

```
procedure
main
(
argc
:
integer
;
argv
:
string_vector_type
)
is
begin
printf
(
"Hello, world\n"
)
;
loop
argc
:=
argc
-
1
;
exit
when
(
argc
=
0
)
;
printf
(
"arg[\%d]: \%s\n"
,
argc
,
argv
[
argc
]
)
;
end
loop
;
end
main
;
```

Some Hints:

- Comments use the traditional C syntax for comments - that is, they start with `/*` and end with `*/` . The comment text should not be included in your output.
- Some lexemes start with an alpha character, then consist of zero or more alpha or numeric characters. The underscore character is considered to be an alpha character.
- Numeric lexemes start with a numeric character, then zero or more additional numeric characters.
- Strings start with `"`, include 1 or more characters, and are terminated with `"`. The entire string should be treated as a single lexeme.
- Please examine this output closely. Your program will be tested by comparing the output of your program with the desired output - your output should match exactly. There should be no extra spaces or extra lines in the output that your program produces, except those contained in strings.