An-Najah National University
Department of Computer Engineering
Compiler Construction-10636419
Summer 2023
Programming Assignment #1
Scanner
Deadline: July 23/2024
Dr. Raed Alqadi

This assignment will produce the scanner for the compiler that you will write for this course. The language that you compile is named ***COMP2023***. The scanner reads a file containing a ***COMP2023*** program and produces a sequence of tokens corresponding to the program's lexemes.
This assignment has four parts:

- Write a set of regular expressions to describe the lexemes of the language.
- Use and Test the File Descriptor Written in PA#0.
- Write and test a scanner.
- Write a report about the ***Unix utility lex,*** which automatically produces scanners from regular expressions describing the lexemes.
-

Note: You must Do Assignment 0 First (File Descriptor)

Lexical Structure of *COMP2023*

Identifiers: An identifier is a sequence of one or more upper-case and lower-case letters, digits, or underscores '_' that does not start with a digit. Examples: A_Funny33COMP2023dentifier, x, z101. Upper- and lower-case letters are different, so dog and DoG are different identifiers.
The following are reserved words (keywords) in ***COMP2023*** and cannot be used as identifiers:

program var constant integer bool string float true
false if fi then else while do od and or
read write for from to by.

Reserved words are always lower case, so for example ***ProGram*** is a legal identifier.

Integer literals are a sequence of digits, optionally preceded by a negative sign (-). They are interpreted as base 10 numbers. Examples: 1, 2, -987623.

Strings are sequences of characters that contain any character except a double quote "or new-line. Strings are delimited by double quote characters and may not contain new lines. An un-terminated string is an error. String examples:***""*** and ***"a string".***

Operators: The following one or two-character symbols are part of ***COMP2023***

( ) : := + - * / = !=
< <= > >= . ; [] , { }

Comments are indicated by double # (**# #**) and contain any characters except a new line. The scanner should recognize comments and strip them from the input; it should ***not*** return a comment token.
Comment Examples: ## A short comment ##
##A longer comment until the end of line
White space (blanks, tabs, new-lines, and form-feeds) are allowed between lexemes. White space is optional where no ambiguity would result, e.g., it is fine to write x+y, but a spaced is needed between **if** and **x** in **if x > 1 then y : = 1**;

All characters not listed above are illegal in **COMP2023** programs.

Task #1
Your first task is to write a set of regular expressions describing these lexemes.

Task #2
Test Your File Descriptor module and make sure it works. Make sure that the Report Error Function reports that an error occurred on the current line of input and print the message as an explanation. The error message should look like:

A:= A + 1. - 3
^
Error: "Bad floating point number" on line 101 of prog1.j

In other words, print the current line of input, print a pointer to the current position on the line, then print the error message along with a. line number.

Scanner
Each time the scanner is invoked, it returns the next token from the input file. A ***token*** is a ***pair c***onsisting of a type-which is an enumerated value that distinguishes the objects recognized by the scanner-and an optional value. The value field is filled only for identifiers, strings, and integers. If the token is an identifier, the value field should contain a string containing the identifier. If the token is a string, the value field points to the string or contains the string. Finally, if the token is an integer, the value field contains the number's integer value (not a sequence of characters).
Note : Use the header file scanner.h to help you in writing the code

Task #3
Write the class Scanner and implement its functions. It should have the function

token *scan (file_descriptor *fd)

The scanner should recognize the tokens for the ***COMP2023*** language defined above. The only additional token is one indicating that the scanner has read up to the end of file. Note that reserved words are distinct from identifiers. Each keyword has a distinct enumerated value while all user-defined identifiers are grouped together under one token and are distinguished by their hash table entry.
Identifiers and strings can be of any length. There should be no limit on the length of input lines.

When translating a sequence of characters to an integer value, you may, but do not have to, detect and report overflow.

Task # 4 lex (Do Not Do This Part Yet)
The final part of the assignment is to produce is to write a report about the Unix utility ***lex*** and how it can be used to produce a scanner.