##### [Click here to enter a date]

**Lexical Analyser**

**Lexical Automatic Generator ‘lex’**

### **Sandra Sherif Wasfi 3940**

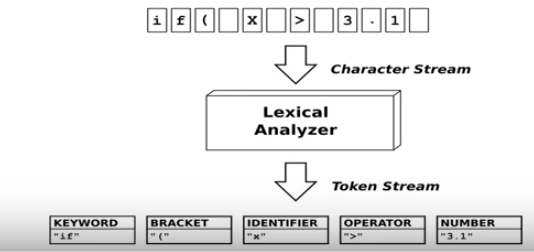
### **Rimon Adel Karam 3960**

### **Youssef Mohamed 3940**

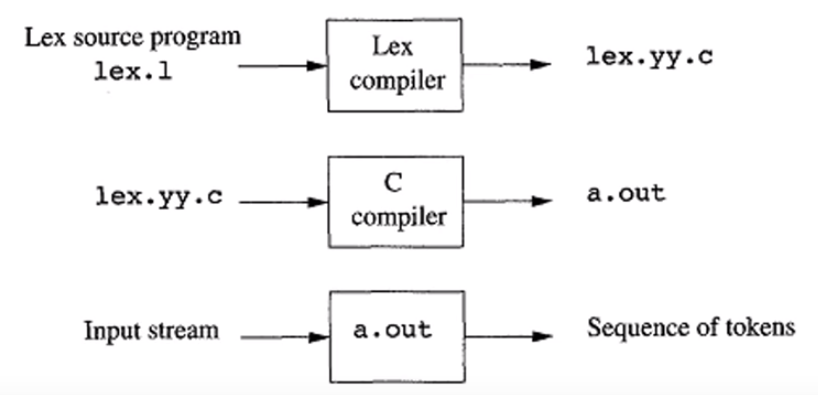
### **Adel atef meleka 3960**

**What is LEX TOOL**

* Lex is a program generator designed for lexical processing of character input streams. It accepts a high-level, problem-oriented specification for character string matching, and produces a program in a general-purpose language which recognizes regular expressions. The regular expressions are specified by the user in the source specifications given to Lex. The Lex written code recognizes these expressions in an input stream and partitions the input stream into strings matching the expressions. At the boundaries between strings program sections provided by the user are executed. The Lex source file associates the regular expressions and the program fragments. As each expression appears in the input to the program written by Lex, the corresponding fragment is executed.
* The user supplies the additional code beyond expression matching needed to complete his tasks, possibly including code written by other generators. The program that recognizes the expressions is generated in the general-purpose programming language employed for the user's program fragments. Thus, a high-level expression language is provided to write the string expressions to be matched while the user's freedom to write actions is unimpaired. This avoids forcing the user who wishes to use a string manipulation language for input analysis to write processing programs in the same and often inappropriate string handling language.
* Lex is not a complete language, but rather a generator representing a new language feature which can be added to different programming languages, called ``host languages.'' Just as general-purpose languages can produce code to run on different computer hardware, Lex can write code in different host languages. The host language is used for the output code generated by Lex and for the program fragments added by the user. Compatible run-time libraries for the different host languages are also provided. This makes Lex adaptable to different environments and different users. Each application may be directed to the combination of hardware and host language appropriate to the task, the user's background, and the properties of local implementations. At present, the only supported host language is C, although Fortran. Lex itself exists on UNIX, GCOS, and OS/370; but the code generated by Lex may be taken anywhere the appropriate compilers exist.
* It’s a tool which generate lexical analyzer.
* Lexical analyzer is the first phase of compiler which take input as source code and generate output as tokens.



* The input notation for the LEX tool is referred as the LEX language and the itself is the LEX compiler.
* The LEX compiler transforms the input patterns into a transition diagram and generates code in a file called lex.yy.c .
* Here is a simple diagram representing phases done a the LEX tool…

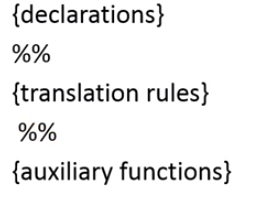


**WORKING**

* An input file, called **lex.l** , is written in the LEX language and describes the lexical analyzer to be generated .
* The LEX compiler transforms lex.l to a C program, in a file that is always name **lex.yy.c** .
* The later file is compiled by the C compiler into file called **a.out** ,as always.
* The C-compiler output is a working lexical analyser that can take a stream of input characters and produce a stream of tokens.

**WORKING**

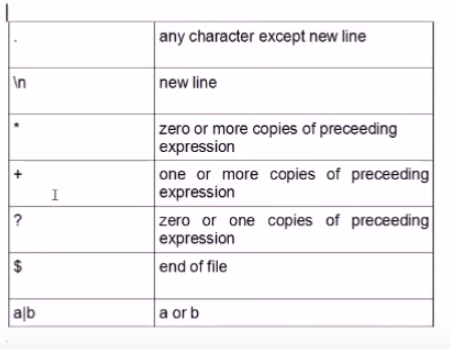
A LEX program has the following form



* The declarations section includes declarations of variables, i.e. #include<…>.
* The translations rules have the form: **Pattern {Action}.**
* The third section holds whatever auxiliary functions are used in the actions. Alternatively, these functions can be compiled separately and loaded with the lexical analyzer.

LEX PATTERNS

Here is attached2



**Sample program**

