Solutions

Lexemes are sequence of character/Symbols that makes up a token, characters in the source program matching a pattern( a description of the form that the lexemes of a token may take) for a token which is identified by the lexical analyser as an instance of that token, they can be seen as a property of the programming structure used be the parser (token), that holds the character/symbols from the input

On the other hand a token is a structure that holds not only the name of the token, but the characters/symbols that make up the token and the start and end position of the string of characters that make up the token, with the start and end position being used for error reporting, highlighting, etc. A token is a pair consisting of a token name and an optional attribute value. The token name is an abstract symbol representing a kind of lexical unit, e.g., a particular keyword, or sequence of input characters denoting an identifier. The token names are the input symbols that the parser processes.

A sample table illustrating tokens and lexemes

|  |  |  |
| --- | --- | --- |
| Token | Informal Description | Lexemes |
| if  | characters i, f  |  if |
| else  | characters e, l, s, e  | else |
| comparison | < or > or <= or >= or == or !=  | <=, != |
| id  | letter followed by letters and digits  | pi, score, D2 |
| number  | any numeric constant  |  3.14159, 0, 6.02e23 |
| literal  | anything but ", surrounded by "'s  | "core dumped" |

An example includes:

x = a + b \* 2

Which yields the lexemes: {x, =, a, +, b, \*, 2}

With corresponding tokens: {<id, 0>, <=>, <id, 1>, <+>, <id, 2>, <\*>}