The lexer should read the source code character by character, generate tokens, and then send them to the parser. For this assignment, I had to implement a lexer that defines the tokens representing the operators, boolean literals and integer literals for my language. I wrote two separate classes. One class was used exclusively to handle the different tokens that were needed. I handled that with a Token_kind enumeration, and for printing purposes I used a string array with all the token names written out as a string. The Token struct has three different kinds that can correspond to the Token_kind enumeration, which are non-literal tokens, boolean tokens, and integer tokens.

The lexer itself was handled in a separate file. I defined the lexer struct and within it I had all the necessary functions that would make a lexer work. I first started off with defining a lexer object which contained a char pointer to the first element, and a char pointer to the last element. Eof() is a member function that returns true if the end of the object is reached. Lookahead() is used to determine the next character to be converted into a token. Consume() increments the first pointer, while next() contains a switch statement that performs the lookahead() and consume() operations until all tokens have been generated.