CSE 340 - Spring 2013
Project 1

In Chapter 6 of your textbook, you learned about the lexical structure of languages, and conducting lexical analysis. In this project, you will create a lexical analyzer based on the language specification below. The analyzer will perform the following tasks:

1. Scan characters from an input file
2. Identify tokens in the character stream using the principle of longest substring
3. Output to a file each valid token and its type
4. Detect and output two types of lexical errors

This document serves as the specification for your program. Be sure you read it closely, and ensure that your program meets all of the requirements outlined in this specification through careful implementation, and adequate testing.

1 The Alphabet

The alphabet includes the following:

- The alphanumeric characters: `A-Z, a-z, 0-9`
- The symbols (separated by commas): `., +, -, *, /, =, >, <, {, }, [, ]`

2 Language Tokens

The language consists of the types of tokens, defined as follows:

1. Reserved Word
   - `begin, end, if, then, for, while, print, int, float`
2. Literal
   - Integer: `[0-9]+`
   - Float: `[0-9]*.[0-9]+`
3. Symbol
   - Math operator: `.,+, -, *, /, =`
   - Comparison operator: `>, <, >=, <=`
   - Shift operator: `>>, <<`
   - Grouping operator: `{, }, {}, [, ], [ ]`
4. Identifier
   - `[A-Za-z][A-Za-z0-9]*`

3 Requirements

Input:	A file to be processed (input as a command line argument)
Output:	A table showing the token, and either its type, or an error code (output to a text file named as `<name of input file>.out`)

3.1 Processing

1. White space is a token-delimiter.
2. Your scanner will follow the principle of longest substring. Proceed by reading the input one character at a time until you have reached a whitespace character, then output the token and its type (or an appropriate error code).
3. Reserved words have higher precedence. That is, although the token `while` also matches the definition of an identifier, it should be categorized as a reserved word.
4. Upon reading a character that is not in the alphabet, continue reading the entire token to output the token with the error code.
5. Create your own test cases to test your code. The example provided below is just one of many test files that will be ran against your code. It is up to you to write your own test files to fully test your code prior to submission.
6. As a starting point, look at figure 6.1 in your textbook. This outlines a preferred methodology for completing your project.
7. Submit your source file on Blackboard by the due date as a single zip file. (See naming conventions in the syllabus.)

3.2 Ouput Types

The output types consist of "exactly" the following:

1. reserved word
2. integer
3. float
4. math operator
5. comparison operator
6. shift operator
7. grouping operator
8. identifier

3.3 Error Codes

An error code is of the form `ERROR CODE` followed by a number. The errors and their codes are the following

- ERROR CODE 0: Lexical error: invalid character found in token
- ERROR CODE 1: Lexical error: token not valid in language

For instance, the token `Tok'ra` contains an invalid character, and your program should print `ERROR CODE 0: Lexical error: invalid character found in token`. The token `4chan` contains all valid characters, but does not fit any of the token type descriptions, and your program should print `ERROR CODE 1: Lexical error: token not valid in language`.

4 Example - Test Case

This example can be used as your first test case, however you should create more test input to fully test your code. The output serves as a description of how your output should be formatted. Ensure that your output is formatted exactly as shown in the example, with no additional output. That is, a computer program should be able to read your output file and make a line by line comparison with a solution file to automatically check your output.

4.1 Input

```
begin
  int money = 0
  while money < 1.0
  {
    print 1money
    money = money + .5
  }
  file << my_dollar
end
```

4.2 Output

```
begin       reserved word
int         reserved word
money       identifier
=           math operator
0           integer
while       reserved word
money       identifier
<           comparison operator
1.0         float
{           grouping operator
print       reserved word
1money      ERROR CODE 1: Lexical error: token not valid in language
money       identifier
=           math operator
money       identifier
+           math operator
.5          float
}           grouping operator
file        identifier
<<          shift operator
my_dollar   ERROR CODE 0: Lexical error: invalid character found in token
end         reserved word
```

5 Grading

Your code must run on the general (see details in syllabus). Check your code there before submitting. A combination of automated checking, and manually viewing your code, will be applied in grading. Working in groups is encouraged, however each student must design and implement their own, individual solution.

1. Correctly formatted output: 10
2. Correctly breaking up tokens using the principle of longest substring: 20
3. Detecting detecting errors in the input: 30
4. Correctly identifying token types: 40