CS319 Lab 4: writing a syntax checker

Goal: use your `stack` class from Lab 3, and what you learned about files in Week 5, to write a syntax checker for a C++ program.

Consider the following piece of C++ code:

```
if ( (n[i] != -1) && (f(g(i)) == 0)
{
  std::cout << "error?";
}
```

If you try to compile this, your compiler may respond with something unhelpful like:

```
test.cpp: In function ‘int main()’:
test.cpp:126: parse error before ‘<<’ token
```

This "parse error" is caused by the `if` statement having five '`(`'-symbols but only four '`)`'-symbols.

For this assignment, you will develop a program that will check through C++ program to make sure that all braces `(`, `{`, and `[` are matched by `)`, `}` and `]`, as appropriate.

However, it is not enough to check that for ever left brace there is matching right one — they also have to be in the correct order. For example `{ ( } )` has the same number of each symbol, but not in the appropriate order.

However, the algorithm below (given in Pseudo-code) may work:

```
get next character A;
if A is one of (, { or [
   place a onto a stack
else if A is one of ), } or ]
   pop B from the stack
   check that B matches A
else ignore A
```

You program should operate as follows:

1. The user should be prompted for the name of a `C++` file to check. This should be opened as an *input file stream*.

2. To read the whole file, one `char` at a time, you can use a `while` and `InFile.eof()`, as in Lecture notes. Or you can use that, if you are at the end of the file, then an error is returned by `cin.get(a)` and so the following `while` statement will exit:

   ```
   while( InFile.get(a) )
   {
     ...
   }
   ```

3. If you encounter a brace such as `(`, `{`, or `[`, you add it to a `char`acter stack.

4. When you encounter a `)`, `}` or `]`, you pop the top entry of the stack and see if they match.

5. The program should report any problems encountered, including

   (a) It a symbol popped from the stack does not match, an informative error message should be reported, including line number.

   (b) When you finished reading the whole file, the stack should be empty. If it is not, it should report a suitable syntax error.

   (c) If there are no problems, the program should report the number of lines checked and that no problems were found.

If you try your program on a *very* simple `C++` program it should work. However, if you try it on, say, the source code for itself, it may well give a false negative (i.e., tell you that there is an error when in fact there is none).

There are two likely causes:

(a) It might interpret the occurrence of `'('` in :

```
if (c == '(')) braces.push(c);
```

as an opening of a brace, rather than just mention of an open brace.

(b) A brace in a commented piece of code could get added to the stack.

Try to address these, and any other similar point in your code.

Please upload your code to Blackboard no later than 9am, Monday 27/02/2017.

In order to maximise the marks you obtain, keep the following in mind:

- **This is an open-ended assignment**. There isn't a perfect solution. Carefully document the capabilities and limitations of your code. For example, it is desirable that your program correctly parse comments. But if it doesn't, that should be included in the documentation, rather than being ignored.
- The more robust your program, the better.
- Test your code against the six test cases at `http://www.maths.nuigalway.ie/~niall/CS319/lab4`
- Check the rubric on Blackboard so that you understand the grading scheme.
- Comment your program sensibly. This means that the program text should include, at the very least, your name and the date of completion. Where necessary, include comments that describe what the program is doing. However, try to make the program "self documenting" as possible by using descriptive function and variable names.