```
/* Filename: delim.cpp
 * --------------------
 * Author:  Darwin Jacob Groskleg
 * Class:   CSCI 162
 * Lab:     # 20
 * Date:    Sunday, March 18th, 2018
 *
 *
 * Purpose: implement a delimited text checker using an integer stack. Should
 * work for the following delimiters:
 *      parenthesis ( )
 *      brackets [ ]
 *      braces { }
 *      single quotes ' '
 *      double quotes " "
 *      angle brackets < >
 * Should detect these problems in text:
 *  - missing left delimiter  (unexpected close)
 *  - missing right delimiter (unclosed)
 *
 *
 * Console
 * ------------------------------------------------------------------------------
 * $> echo "yey'ey)<>" | ./delim
 * WARNING: expected right delimiter for ' before column 7
 * WARNING: missing left delimiter for ) in column 7
 *
 * $> echo "jkl{abba(\"yeyey\"<<><riroio>>)}asdf{[a],'pp'}vb" | ./delim
 * Text is properly delimited.
 *
 * $> echo "message(for: [\"Jane\", 'H'], <Hello END;" | ./delim
 * WARNING: missing right delimiter for < in column 39 (end of the input text)
 * WARNING: missing right delimiter for ( in column 39 (end of the input text)
 *
 * ------------------------------------------------------------------------------
 */

#include <iostream>

#include "intStack.h"   // original, unchanged
#include "matcher.h"

using namespace std;

int main() {
    bool success = true;
    char ch;
    int input_count = 0;

    const int MAX_LEFT_DELIMETERS = 100;
    intStack leftDelimStack(MAX_LEFT_DELIMETERS); // Be explicit
    Matcher m;

    while (cin.get(ch) && ch != '\n') {
        input_count++;

        if (!isalpha(ch)) {

            // NOTE This can be refactored to be much smaller but the result
            // is much less explicit and obvious at first read.
            if (m.isRight(ch)) {

                if (m.isMatching(leftDelimStack.Top(), ch))
                    leftDelimStack.Pop();

                else if (m.isLeft(ch))      // Treat like a left delim?
                    leftDelimStack.Push(ch);


```

```
70                  else {                      // if right and no match
71                      success = false;
72                      if(!leftDelimStack.Empty()) {
73                          cout << "WARNING: expected right delimiter for "
74                               << (char) leftDelimStack.Top()
75                               << " before column " << input_count
76                               << endl;
77                          leftDelimStack.Pop();
78                      }
79
80                      cout << "WARNING: missing left delimiter for "
81                           << ch << " in column "
82                           << input_count
83                           << endl;
84                  }
85
86              } else if (m.isLeft(ch))
87                  leftDelimStack.Push(ch);
88          }
89      }
90
91      // Handle what is left in the stack at the end of input
92      while (!leftDelimStack.Empty()) {
93          success = false;
94          cout << "WARNING: missing right delimiter for "
95               << (char) leftDelimStack.Top()
96               << " in column " << input_count << " (end of the input text)"
97               << endl;
98          leftDelimStack.Pop();
99      }
100
101     if (success)
102         cout << "Text is properly delimited." << endl;
103
104     return 0;
105 }
106
```

```
 1  /* Filename: matcher.h
 2   * --------------------
 3   * Author:  Darwin Jacob Groskleg
 4   * Class:   CSCI 162
 5   * Lab:     # 20
 6   * Date:    Sunday, March 18th, 2018
 7   *
 8   * Purpose: interface for matcher class. Defined behaviours on a set of left
 9   * and right, matching delimiters.
10   */
11
12  #ifndef MATCHER_H_INCLUDED
13  #define MATCHER_H_INCLUDED
14
15  class Matcher {
16      private:
17          static const int DELIMS = 6;
18          char left_delims[DELIMS]  = {'(', '[', '{', '\'', '"', '<'};
19          char right_delims[DELIMS] = {')', ']', '}', '\'', '"', '>'};
20      public:
21          Matcher() {};
22         ~Matcher() {};
23
24          bool isMatching(char left_delim, char right_delim) const;
25          bool isLeft(char delim) const;
26          bool isRight(char delim) const;
27  };
28
29  #endif // MATCHER_H_INCLUDED
```

```cpp
/* Filename: matcher.cpp
 * --------------------
 * Author:  Darwin Jacob Groskleg
 * Class:   CSCI 162
 * Lab:     # 20
 * Date:    Sunday, March 18th, 2018
 *
 * Purpose: implements matcher class. Defined behaviours on a set of left
 * and right, matching delimiters.
 */

#include "matcher.h"

/* Method Name: isMatching
 * -----------------------
 * Returns true if the two passed characters are matching delimiters.
 * Expects left and right arguments to be valid left and right delimiters
 * respectively.
 */
bool Matcher::isMatching(char left_delim, char right_delim) const {
    int left_position, right_position;

    for (int i=0; i<DELIMS; i++) {
        if (left_delims[i]  == left_delim)  left_position  = i;
        if (right_delims[i] == right_delim) right_position = i;
    }
    return (left_position == right_position);
}

/* Method Name: isLeft
 * -------------------------
 * Returns true if the given char is a left (opening) delimiter.
 */
bool Matcher::isLeft(char delim) const {
    for (int i=0; i<DELIMS; i++) {
        if (left_delims[i] == delim) return true;
    }
    return false;
}

/* Method Name: isRight
 * ------------------------
 * Returns true if the given char is a right (closing) delimiter.
 */
bool Matcher::isRight(char delim) const {
    for (int i=0; i<DELIMS; i++) {
        if (right_delims[i] == delim) return true;
    }
    return false;
}

```