```
1: // $Id: lexicalcast.cpp,v 1.6 2015-02-05 19:08:00-08 - - $
 2:
 3: //
 4: // Illustrate to_string, from_string, lexical_cast.
 5: // Assume the use of << and >> on sstreams.
 6: //
 7:
 8: #include <iostream>
 9: #include <sstream>
10: #include <stdexcept>
11: #include <string>
12: #include <typeinfo>
13:
14: using namespace std;
15:
16: //
17: // convert a thing into a string.
18: //
19:
20: template <typename item_t>
21: string to_string (item_t that) {
22:
       ostringstream stream;
23:
       stream << that;</pre>
24:
       return stream.str ();
25: }
26:
27: //
28: // Scan a string to grab a thing.
29: //
30:
31: template <typename item_t>
32: item_t from_string (const string &that) {
33:
       stringstream stream;
34:
       stream << that;</pre>
35:
       item_t result;
       bool converted = stream >> result // Is string is a valid item_t?
36:
37:
                     && stream >> std::ws // Flush trailing white space.
                                           // Must now be at end of stream.
38:
                     && stream.eof();
39:
       if (not converted) {
40:
          throw domain_error (string (typeid (item_t).name())
41:
                + " from_string (" + that + ")");
42:
43:
       return result;
44: }
45:
47: // Lexically cast a thing to a string then to another thing.
48: //
50: template <typename target_t, typename source_t>
51: target_t lexical_cast (source_t that) {
       return from_string (to_string (that));
52:
53: }
54:
55: //
56: // Main.
57: //
58:
```

02/05/15 19:08:00

\$cmps109-wm/Examples/wk06b-algorithms/lexicalcast.cpp

```
2/2
```

```
59: int main () {
60:    cout << to_string<double> (9) << endl;
61:    cout << from_string<int> ("42") << endl;
62:    return 0;
63: }
64:
65: //TEST// lexicalcast >lexicalcast.out 2>&1
66: //TEST// mkpspdf lexicalcast.ps lexicalcast.cpp lexicalcast.out
67:
```

02/05/15	\$cmps109-wm/Examples/wk06b-algorithms/
19:08:00	lexicalcast.out

1/1

1: 9 2: 42