```
 1: // $Id: insert_list.cpp,v 1.1 2016-08-12 15:33:15-07 - - $
 2:
 3: // Example of insert_ascending to a list.
 4: // This runs in O(n) time.
 5: // Obviously there are better algorithms.
 6:
 7: #include <list>
 8: #include <iostream>
 9: #include <iterator>
10: #include <string>
11: #include <vector>
12: using namespace std;
13:
14: // Insert ascending order but no duplicates.
15: // Can't use any equality operator, only less.
16: template <typename T, class Less = less<T>>
17: void insert_ascending (list<T>& list, const T& item) {
18:    Less less;
19:    auto curr = list.begin();
20:    while (curr != list.end()) {
21:       if (not (less (*curr, item))) break;
22:       ++curr;
23:    }
24:    if (curr == list.end()) {
25:       list.push_back (item);
26:    }else if (less (item, *curr)) {
27:       list.insert (curr, item);
28:    }
29: }
30:
31: int main() {
32:    list<string> list;
33:    istream_iterator<string> cin_itor (cin);
34:    istream_iterator<string> end_file;
35:    ostream_iterator<string> cout_itor (cout, "\n");
36:    vector<string> data (cin_itor, end_file);
37:    cout << endl << "Unsorted data:" << endl;
38:    copy (data.begin(), data.end(), cout_itor);
39:    for (auto& word: data) insert_ascending (list, word);
40:    cout << endl << "Sorted data:" << endl;
41:    copy (list.begin(), list.end(), cout_itor);
42: }
43:
44: /*
45: //TEST// echo hello world foo bar baz qux This is some test data. \
46: //TEST// | insert_list >insert_list.out 2>&1
47: //TEST// mkpspdf insert_list.ps insert_list.cpp* \
48: //TEST//         insert_list.out
49: */
```

```
    1: @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ mkc: starting insert_list.cpp
    2: insert_list.cpp: $Id: insert_list.cpp,v 1.1 2016-08-12 15:33:15-07 - - $
    3: g++ -g -O0 -std=gnu++14 -rdynamic -Wall -Wextra -Wold-style-cast insert_
list.cpp -o insert_list -lglut -lGLU -lGL -lX11 -lrt -lm
    4: cpplint.py.perl insert_list.cpp
    5: Done processing insert_list.cpp
    6: @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ mkc: finished insert_list.cpp
```

```
 1:
 2: Unsorted data:
 3: hello
 4: world
 5: foo
 6: bar
 7: baz
 8: qux
 9: This
10: is
11: some
12: test
13: data.
14:
15: Sorted data:
16: This
17: bar
18: baz
19: data.
20: foo
21: hello
22: is
23: qux
24: some
25: test
26: world
```