```
    1: /*************************************************************************
***********/
    2: /* mmtest.cpp
          */
    3: /* Yoo Min Cha
          */
    4: /* Markov's Model
          */
    5: /* Professor Martin
          */
    6: /* 07 April 2014
          */
    7: /*************************************************************************
***********/
    8:
    9: // build mmtest with $make mmtest
   10:
   11: #include <iostream>
   12: #include <string>
   13: #include <exception>
   14: #include <stdexcept>
   15:
   16: #include "MarkovModel.hpp"
   17:
   18: #define BOOST_TEST_DYN_LINK
   19: #define BOOST_TEST_MODULE Main
   20: #include <boost/test/unit_test.hpp>
   21:
   22: using namespace std;
   23:
   24: BOOST_AUTO_TEST_CASE(order0) {
   25:   // normal constructor
   26:   BOOST_REQUIRE_NO_THROW(MarkovModel("gagggagaggcgagaaa", 0));
   27:
   28:   MarkovModel mm("gagggagaggcgagaaa", 0);
   29:
   30:   BOOST_REQUIRE(mm.order() == 0);
   31:   BOOST_REQUIRE(mm.freq("") == 17); // length of input in constructor
   32:   BOOST_REQUIRE_THROW(mm.freq("x"), std::runtime_error);
   33:
   34:   BOOST_REQUIRE(mm.freq("", 'g') == 9);
   35:   BOOST_REQUIRE(mm.freq("", 'a') == 7);
   36:   BOOST_REQUIRE(mm.freq("", 'c') == 1);
   37:   BOOST_REQUIRE(mm.freq("", 'x') == 0);
   38:
   39: }
   40:
   41: BOOST_AUTO_TEST_CASE(order1) {
   42:   // normal constructor
   43:   BOOST_REQUIRE_NO_THROW(MarkovModel("gagggagaggcgagaaa", 1));
   44:
   45:   MarkovModel mm("gagggagaggcgagaaa", 1);
   46:
   47:   BOOST_REQUIRE(mm.order() == 1);
   48:   BOOST_REQUIRE_THROW(mm.freq(""), std::runtime_error);
   49:   BOOST_REQUIRE_THROW(mm.freq("xx"), std::runtime_error);
   50:
   51:   BOOST_REQUIRE(mm.freq("a") == 7);
   52:   BOOST_REQUIRE(mm.freq("g") == 9);
   53:   BOOST_REQUIRE(mm.freq("c") == 1);
   54:
   55:   BOOST_REQUIRE(mm.freq("a", 'a') == 2);
   56:   BOOST_REQUIRE(mm.freq("a", 'c') == 0);
   57:   BOOST_REQUIRE(mm.freq("a", 'g') == 5);
   58:
```

```
   59:   BOOST_REQUIRE(mm.freq("c", 'a') == 0);
   60:   BOOST_REQUIRE(mm.freq("c", 'c') == 0);
   61:   BOOST_REQUIRE(mm.freq("c", 'g') == 1);
   62:
   63:   BOOST_REQUIRE(mm.freq("g", 'a') == 5);
   64:   BOOST_REQUIRE(mm.freq("g", 'c') == 1);
   65:   BOOST_REQUIRE(mm.freq("g", 'g') == 3);
   66:
   67:   BOOST_REQUIRE_NO_THROW(mm.randk("a"));
   68:   BOOST_REQUIRE_NO_THROW(mm.randk("c"));
   69:   BOOST_REQUIRE_NO_THROW(mm.randk("g"));
   70:
   71:   BOOST_REQUIRE_THROW(mm.randk("x"), std::runtime_error);
   72:
   73:   BOOST_REQUIRE_THROW(mm.randk("xx"), std::runtime_error);
   74:
   75: }
   76:
   77: BOOST_AUTO_TEST_CASE(order2) {
   78:   // normal constructor
   79:   BOOST_REQUIRE_NO_THROW(MarkovModel("gagggagaggcgagaaa", 2));
   80:
   81:   MarkovModel mm("gagggagaggcgagaaa", 2);
   82:
   83:   BOOST_REQUIRE(mm.order() == 2);
   84:
   85:   BOOST_REQUIRE_THROW(mm.freq(""), std::runtime_error);
   86:   BOOST_REQUIRE_THROW(mm.freq("x"), std::runtime_error);
   87:   BOOST_REQUIRE_NO_THROW(mm.freq("xx"));
   88:   BOOST_REQUIRE_THROW(mm.freq("", 'g'), std::runtime_error); // kgram is
wrong length
   89:   BOOST_REQUIRE_THROW(mm.freq("x", 'g'), std::runtime_error); // kgram is
 wrong length
   90:   BOOST_REQUIRE_THROW(mm.freq("xxx", 'g'), std::runtime_error); // kgram
is wrong length
   91:
   92:
   93:   BOOST_REQUIRE(mm.freq("aa") == 2);
   94:   BOOST_REQUIRE(mm.freq("aa", 'a') == 1);
   95:   BOOST_REQUIRE(mm.freq("aa", 'c') == 0);
   96:   BOOST_REQUIRE(mm.freq("aa", 'g') == 1);
   97:
   98:   BOOST_REQUIRE(mm.freq("ag") == 5);
   99:   BOOST_REQUIRE(mm.freq("ag", 'a') == 3);
  100:   BOOST_REQUIRE(mm.freq("ag", 'c') == 0);
  101:   BOOST_REQUIRE(mm.freq("ag", 'g') == 2);
  102:
  103:   BOOST_REQUIRE(mm.freq("cg") == 1);
  104:   BOOST_REQUIRE(mm.freq("cg", 'a') == 1);
  105:   BOOST_REQUIRE(mm.freq("cg", 'c') == 0);
  106:   BOOST_REQUIRE(mm.freq("cg", 'g') == 0);
  107:
  108:   BOOST_REQUIRE(mm.freq("ga") == 5);
  109:   BOOST_REQUIRE(mm.freq("ga", 'a') == 1);
  110:   BOOST_REQUIRE(mm.freq("ga", 'c') == 0);
  111:   BOOST_REQUIRE(mm.freq("ga", 'g') == 4);
  112:
  113:   BOOST_REQUIRE(mm.freq("gc") == 1);
  114:   BOOST_REQUIRE(mm.freq("gc", 'a') == 0);
  115:   BOOST_REQUIRE(mm.freq("gc", 'c') == 0);
  116:   BOOST_REQUIRE(mm.freq("gc", 'g') == 1);
  117:
  118:   BOOST_REQUIRE(mm.freq("gg") == 3);
  119:   BOOST_REQUIRE(mm.freq("gg", 'a') == 1);
  120:   BOOST_REQUIRE(mm.freq("gg", 'c') == 1);
```

```
  121:   BOOST_REQUIRE(mm.freq("gg", 'g') == 1);
  122:
  123: }
```