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

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