```
 1: // Angel Zheondre Calcano
 2: // PS6
 3:
 4: #include <iostream>
 5: #include <string>
 6: #include <exception>
 7: #include <stdexcept>
 8:
 9: #include "MarkovModel.hpp"
10:
11: #define BOOST_TEST_DYN_LINK
12: #define BOOST_TEST_MODULE Main
13: #include <boost/test/unit_test.hpp>
14:
15: using namespace std;
16:
17: BOOST_AUTO_TEST_CASE(order0) {
18:   // normal constructor
19:   BOOST_REQUIRE_NO_THROW(MarkovModel("gagggagaggcgagaaa", 0));
20:
21:   MarkovModel mm("gagggagaggcgagaaa", 0);
22:
23:   BOOST_REQUIRE(mm.order() == 0);
24:   BOOST_REQUIRE(mm.freq("") == 17); // length of input in constructor
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:
34: BOOST_AUTO_TEST_CASE(order1) {
35:   // normal constructor
36:   BOOST_REQUIRE_NO_THROW(MarkovModel("gagggagaggcgagaaa", 1));
37:
38:   MarkovModel mm("gagggagaggcgagaaa", 1);
39:
40:   BOOST_REQUIRE(mm.order() == 1);
41:   BOOST_REQUIRE_THROW(mm.freq(""), std::runtime_error);
42:   BOOST_REQUIRE_THROW(mm.freq("xx"), std::runtime_error);
43:
44:   BOOST_REQUIRE(mm.freq("a") == 7);
45:   BOOST_REQUIRE(mm.freq("g") == 9);
46:   BOOST_REQUIRE(mm.freq("c") == 1);
47:
48:   BOOST_REQUIRE(mm.freq("a", 'a') == 2);
49:   BOOST_REQUIRE(mm.freq("a", 'c') == 0);
50:   BOOST_REQUIRE(mm.freq("a", 'g') == 5);
51:
52:   BOOST_REQUIRE(mm.freq("c", 'a') == 0);
53:   BOOST_REQUIRE(mm.freq("c", 'c') == 0);
54:   BOOST_REQUIRE(mm.freq("c", 'g') == 1);
55:
56:   BOOST_REQUIRE(mm.freq("g", 'a') == 5);
57:   BOOST_REQUIRE(mm.freq("g", 'c') == 1);
58:   BOOST_REQUIRE(mm.freq("g", 'g') == 3);
59:
60:   BOOST_REQUIRE_NO_THROW(mm.randk("a"));
61:   BOOST_REQUIRE_NO_THROW(mm.randk("c"));
```

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