```cpp
// Angel Zheondre Calcano
// PS6

#include <iostream>
#include <string>
#include <exception>
#include <stdexcept>

#include "MarkovModel.hpp"

#define BOOST_TEST_DYN_LINK
#define BOOST_TEST_MODULE Main
#include <boost/test/unit_test.hpp>

using namespace std;

BOOST_AUTO_TEST_CASE(order0) {
  // normal constructor
  BOOST_REQUIRE_NO_THROW(MarkovModel("gagggagaggcgagaaa", 0));

  MarkovModel mm("gagggagaggcgagaaa", 0);

  BOOST_REQUIRE(mm.order() == 0);
  BOOST_REQUIRE(mm.freq("") == 17); // length of input in constructor
  BOOST_REQUIRE_THROW(mm.freq("x"), std::runtime_error);

  BOOST_REQUIRE(mm.freq("", 'g') == 9);
  BOOST_REQUIRE(mm.freq("", 'a') == 7);
  BOOST_REQUIRE(mm.freq("", 'c') == 1);
  BOOST_REQUIRE(mm.freq("", 'x') == 0);

}

BOOST_AUTO_TEST_CASE(order1) {
  // normal constructor
  BOOST_REQUIRE_NO_THROW(MarkovModel("gagggagaggcgagaaa", 1));

  MarkovModel mm("gagggagaggcgagaaa", 1);

  BOOST_REQUIRE(mm.order() == 1);
  BOOST_REQUIRE_THROW(mm.freq(""), std::runtime_error);
  BOOST_REQUIRE_THROW(mm.freq("xx"), std::runtime_error);

  BOOST_REQUIRE(mm.freq("a") == 7);
  BOOST_REQUIRE(mm.freq("g") == 9);
  BOOST_REQUIRE(mm.freq("c") == 1);

  BOOST_REQUIRE(mm.freq("a", 'a') == 2);
  BOOST_REQUIRE(mm.freq("a", 'c') == 0);
  BOOST_REQUIRE(mm.freq("a", 'g') == 5);

  BOOST_REQUIRE(mm.freq("c", 'a') == 0);
  BOOST_REQUIRE(mm.freq("c", 'c') == 0);
  BOOST_REQUIRE(mm.freq("c", 'g') == 1);

  BOOST_REQUIRE(mm.freq("g", 'a') == 5);
  BOOST_REQUIRE(mm.freq("g", 'c') == 1);
  BOOST_REQUIRE(mm.freq("g", 'g') == 3);

  BOOST_REQUIRE_NO_THROW(mm.randk("a"));
  BOOST_REQUIRE_NO_THROW(mm.randk("c"));
  BOOST_REQUIRE_NO_THROW(mm.randk("g"));

  BOOST_REQUIRE_THROW(mm.randk("x"), std::runtime_error);

  BOOST_REQUIRE_THROW(mm.randk("xx"), std::runtime_error);

}

BOOST_AUTO_TEST_CASE(order2) {
  // normal constructor
  BOOST_REQUIRE_NO_THROW(MarkovModel("gagggagaggcgagaaa", 2));

  MarkovModel mm("gagggagaggcgagaaa", 2);

  BOOST_REQUIRE(mm.order() == 2);

  BOOST_REQUIRE_THROW(mm.freq(""), std::runtime_error);
  BOOST_REQUIRE_THROW(mm.freq("x"), std::runtime_error);
  BOOST_REQUIRE_NO_THROW(mm.freq("xx"));
  BOOST_REQUIRE_THROW(mm.freq("", ’g’), std::runtime_error); // kgram is wrong length
  BOOST_REQUIRE_THROW(mm.freq("x", ’g’), std::runtime_error); // kgram is wrong length
  BOOST_REQUIRE_THROW(mm.freq("xxx", ’g’), std::runtime_error); // kgram is wrong length


  BOOST_REQUIRE(mm.freq("aa") == 2);
  BOOST_REQUIRE(mm.freq("aa", ’a’) == 1);
  BOOST_REQUIRE(mm.freq("aa", ’c’) == 0);
  BOOST_REQUIRE(mm.freq("aa", ’g’) == 1);

  BOOST_REQUIRE(mm.freq("ag") == 5);
  BOOST_REQUIRE(mm.freq("ag", ’a’) == 3);
  BOOST_REQUIRE(mm.freq("ag", ’c’) == 0);
  BOOST_REQUIRE(mm.freq("ag", ’g’) == 2);

  BOOST_REQUIRE(mm.freq("cg") == 1);
  BOOST_REQUIRE(mm.freq("cg", ’a’) == 1);
  BOOST_REQUIRE(mm.freq("cg", ’c’) == 0);
  BOOST_REQUIRE(mm.freq("cg", ’g’) == 0);

  BOOST_REQUIRE(mm.freq("ga") == 5);
  BOOST_REQUIRE(mm.freq("ga", ’a’) == 1);
  BOOST_REQUIRE(mm.freq("ga", ’c’) == 0);
  BOOST_REQUIRE(mm.freq("ga", ’g’) == 4);

  BOOST_REQUIRE(mm.freq("gc") == 1);
  BOOST_REQUIRE(mm.freq("gc", ’a’) == 0);
  BOOST_REQUIRE(mm.freq("gc", ’c’) == 0);
  BOOST_REQUIRE(mm.freq("gc", ’g’) == 1);

  BOOST_REQUIRE(mm.freq("gg") == 3);
  BOOST_REQUIRE(mm.freq("gg", ’a’) == 1);
  BOOST_REQUIRE(mm.freq("gg", ’c’) == 1);
  BOOST_REQUIRE(mm.freq("gg", ’g’) == 1);

}
```