```
// "Copyright 2020 <Greg Kaplowitz>"
#define BOOST_TEST_DYN_LINK
#define BOOST_TEST_MODULE Main
#include <boost/test/unit_test.hpp>
#include <iostream>
#include <string>
#include "markov.h"

BOOST_AUTO_TEST_CASE(kgram_freq) {
  MarkovModel m1("gagggagaggcgagaaa", 2);
  BOOST_REQUIRE(m1.freq("aa") == 2);
}

BOOST_AUTO_TEST_CASE(char_freq) {
  MarkovModel m1("gagggagaggcgagaaa", 2);
  BOOST_REQUIRE( m1.freq("aa", 'a') == 1);
  BOOST_REQUIRE( m1.freq("aa", 'c') == 0);
  BOOST_REQUIRE( m1.freq("aa", 'g') == 1);
}

BOOST_AUTO_TEST_CASE(freq_without_a_valid_kgram) {
  MarkovModel m1("gagggagaggcgagaaa", 2);
   BOOST_REQUIRE(m1.freq("az") == 0);
    BOOST_REQUIRE( m1.freq("aa", 'z') == 0);
}

BOOST_AUTO_TEST_CASE(invalid_argument) {
  MarkovModel m1("gagggagaggcgagaaa", 2);
  BOOST_REQUIRE_THROW(m1.freq("aaaa"), std::invalid_argument);
  BOOST_REQUIRE_THROW(m1.freq("aaaa", 'a'), std::invalid_argument);
   BOOST_REQUIRE_THROW(m1.kRand("aaaa"), std::invalid_argument);
   BOOST_REQUIRE_THROW(m1.generate("aaaa", 3), std::invalid_argument);
}

BOOST_AUTO_TEST_CASE(generate) {
  MarkovModel m1("gagggagaggcgagaaa", 2);
   BOOST_REQUIRE(m1.generate("gg", 10) == "ggagagaggg");
}


```