

```
1: // "Copyright 2020 <Greg Kaplowitz>"
2: #define BOOST_TEST_DYN_LINK
3: #define BOOST_TEST_MODULE Main
4: #include <boost/test/unit_test.hpp>
5: #include <iostream>
6: #include <string>
7: #include "markov.h"
8:
9: BOOST_AUTO_TEST_CASE(kgram_freq) {
10:     MarkovModel m1("gagggagagggcgagaaa", 2);
11:     BOOST_REQUIRE(m1.freq("aa") == 2);
12: }
13:
14: BOOST_AUTO_TEST_CASE(char_freq) {
15:     MarkovModel m1("gagggagagggcgagaaa", 2);
16:     BOOST_REQUIRE(m1.freq("aa", 'a') == 1);
17:     BOOST_REQUIRE(m1.freq("aa", 'c') == 0);
18:     BOOST_REQUIRE(m1.freq("aa", 'g') == 1);
19: }
20:
21: BOOST_AUTO_TEST_CASE(freq_without_a_valid_kgram) {
22:     MarkovModel m1("gagggagagggcgagaaa", 2);
23:     BOOST_REQUIRE(m1.freq("az") == 0);
24:     BOOST_REQUIRE(m1.freq("aa", 'z') == 0);
25: }
26:
27: BOOST_AUTO_TEST_CASE(invalid_argument) {
28:     MarkovModel m1("gagggagagggcgagaaa", 2);
29:     BOOST_REQUIRE_THROW(m1.freq("aaaa"), std::invalid_argument);
30:     BOOST_REQUIRE_THROW(m1.freq("aaaa", 'a'), std::invalid_argument);
31:     BOOST_REQUIRE_THROW(m1.kRand("aaaa"), std::invalid_argument);
32:     BOOST_REQUIRE_THROW(m1.generate("aaaa", 3), std::invalid_argument);
33: }
34:
35: BOOST_AUTO_TEST_CASE(generate) {
36:     MarkovModel m1("gagggagagggcgagaaa", 2);
37:     BOOST_REQUIRE(m1.generate("gg", 10) == "ggagagaggg");
38: }
39:
40:
```