```
    1: // Copyright 2022 Matthew Lorette Anaya
    2:
    3: #include "RandWriter.h"
    4:
    5: RandWriter::RandWriter(std::string text, int k) {
    6:    rw_txt = text;
    7:     rw_k = k;
    8:
    9:     if (rw_txt.length() < static_cast<unsigned int>(rw_k)) {
   10:         throw std::invalid_argument("RandWriter(string text, int k): orde
r k"
   11:         " must be less than or equal to text length.");
   12:     }
   13:
   14:     // Table setup
   15:     unsigned int pos = 0;
   16:     for (unsigned int i = 0; i < rw_txt.length(); i++) {
   17:         std::string kgram;
   18:         std::map<char, int> freq_table;
   19:
   20:         // kgrams parsing
   21:         for (unsigned int j = i; j < i + rw_k; j++) {
   22:             if (j >= rw_txt.length()) {
   23:                 pos = j - rw_txt.length();
   24:             } else {
   25:                 pos = j;
   26:             }
   27:             kgram += rw_txt.at(pos);
   28:         }
   29:
   30:         // Frequency table setup
   31:         pos++;
   32:         if (pos >= rw_txt.length()) { pos -= rw_txt.length(); }
   33:         freq_table.insert(std::make_pair(rw_txt.at(pos), 0));
   34:
   35:         // Mapping
   36:         if (rw_table.count(kgram) == 0) {
   37:             rw_table.insert(std::make_pair(kgram, freq_table));
   38:         }
   39:
   40:         rw_table[kgram][rw_txt.at(pos)]++;
   41:     }
   42: }
   43:
   44: int RandWriter::orderK() const {
   45:         return rw_k;
   46: }
   47:
   48: std::string RandWriter::getText() const {
   49:         return rw_txt;
   50: }
   51:
   52: std::map<std::string, std::map<char, int>> RandWriter::get_table() const
{
   53:     return rw_table;
   54: }
   55:
   56: int RandWriter::freq(std::string kgram) const {
   57:     if (kgram.length() < static_cast<unsigned int>(rw_k)) {
   58:         throw std::runtime_error("freq(string kgram): kgram must be of"
   59:         " length greater than or equal to order k.");
   60:     }
   61:     int count = 0;
   62:     for (unsigned int i = 0; i < rw_txt.length(); i++) {
   63:         unsigned int pos = 0;
```

```
   64:         std::string kg;
   65:         // parse input text for kgrams
   66:         for (unsigned int j = i; j < i + rw_k; j++) {
   67:             // get characters for kgrams
   68:             if (j >= rw_txt.length()) {
   69:                 pos = j - rw_txt.length();
   70:             } else {
   71:                 pos = j;
   72:             }
   73:             kg += rw_txt.at(pos);
   74:         }
   75:         if (kgram == kg) { count++; }
   76:     }
   77:     return count;
   78: }
   79:
   80: int RandWriter::freq(std::string kgram, char c) const {
   81:     if (kgram.length() < static_cast<unsigned int>(rw_k)) {
   82:         throw std::runtime_error("freq(string kgram, char c): kgram must
be"
   83:         " of length greater than or equal to order k.");
   84:     }
   85:     return rw_table.at(kgram).at(c);
   86: }
   87:
   88: char RandWriter::kRand(std::string kgram) const {
   89:     if (kgram.length() < static_cast<unsigned int>(rw_k)) {
   90:         throw std::runtime_error("kRand(string kgram): kgram must be of"
   91:         " length greater than or equal to order k.");
   92:     }
   93:     if (rw_table.count(kgram) == 0) {
   94:         throw std::runtime_error("kRand(string kgram): kgram does not"
   95:         " exist.");
   96:     }
   97:     std::string alphabet;
   98:     for (auto const &var1 : rw_table) {
   99:         if (var1.first == kgram) {
  100:             for (auto const &var2 : var1.second) {
  101:                 alphabet += var2.first;
  102:             }
  103:         }
  104:     }
  105:     std::random_device device;
  106:     std::mt19937 mt_rand(device());
  107:     std::uniform_int_distribution<int> distribution(0, alphabet.length()
  108:     - 1);
  109:
  110:     return alphabet[distribution(mt_rand)];
  111: }
  112:
  113: std::string RandWriter::generate(std::string kgram, int L) const {
  114:     if (kgram.length() < static_cast<unsigned int>(rw_k)) {
  115:         throw std::runtime_error("generate(string kgram, int L): kgram mu
st"
  116:         " be of length greater than or equal to order k.");
  117:     }
  118:     std::string generated = kgram;
  119:     // generate new characters based on kgrams
  120:     for (int i = rw_k; i < L; i++) {
  121:         generated += kRand(generated.substr(i - rw_k, rw_k));
  122:     }
  123:     return generated;
  124: }
  125:
  126: std::ostream& operator<<(std::ostream& out, const RandWriter& rw) {
```

```
  127:     out << "Markov Model\tOrder: " << rw.rw_k << std::endl;
  128:     out << "kgram:\tfrequency:\tfrqncy of next char:\tprob of next char:"
 <<
  129:     std::endl;
  130:
  131:     for (auto const &var1 : rw.rw_table) {
  132:         // var1.first = kgram
  133:         out << var1.first << "\t";
  134:         out << rw.freq(var1.first) << "\t\t";
  135:         for (auto const &var2 : var1.second) {
  136:             // var2.first = next char
  137:             // var2.second = data
  138:             out << var2.first << ":" << var2.second << " ";
  139:         }
  140:         out << "\t\t\t";
  141:         for (auto const &var2 : var1.second) {
  142:             out << var2.first << ":" << var2.second << "/" <<
  143:             rw.freq(var1.first) << " ";
  144:         }
  145:         out << std::endl;
  146:     }
  147:     return out;
  148: }
```