Ps6 Source Code: Makefile

```
CFLAGS = -O3 -Wall -Werror -std=c++11 -ansi -pedantic
DEPS = -lboost_unit_test_framework

all:    TextGenerator test.o

TextGenerator:  TextGenerator.o MModel.o
    g++ TextGenerator.o MModel.o -o TextGenerator

TextGenerator.o: TextGenerator.cpp MModel.h
    g++ -c TextGenerator.cpp MModel.h $(CFLAGS)

MModel.o: MModel.cpp MModel.h
    g++ -c MModel.cpp  MModel.h $(CFLAGS)

test.o:
    g++ test.cpp MModel.cpp MModel.h -o test.o $(CFLAGS) $(DEPS)

clean:
    rm *.o
    rm *.gch
    rm TextGenerator
```

Ps6 Source Code: TextGenerator.cpp

```
// Copyright 2020 Karl Marx
#include <string>
#include "MModel.h"

int main(int argc, char* argv[]) {
  if (argc != 3) {
    std::cout << "Invalid Number of Arguments. Must == 3";
    return -1;
  }

  int k = atoi(argv[1]);
  int L = atoi(argv[2]);

  std::string input;
  std::string temp;

  while (std::cin >> temp) {
    input += " " + temp;
```

```
    temp = "";
  }

  std::string output;

  MModel model(input, k);

  output += "" + model.generate(input.substr(0, k), L);
  std::cout << output << std::endl;

  return 0;
}
```

Ps6 Source Code: test.cpp

```
// Copyright 2020 Karl Marx

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

BOOST_AUTO_TEST_CASE(Exception) {
    MModel a("hello", 2);
    BOOST_REQUIRE_THROW(a.freq("h"), std::runtime_error);
    BOOST_REQUIRE_THROW(a.freq("hhh"), std::runtime_error);
    BOOST_REQUIRE_THROW(a.freq("hhh", 'e'), std::runtime_error);
    BOOST_REQUIRE_THROW(a.freq("h", 'e'), std::runtime_error);
    BOOST_REQUIRE_THROW(a.generate("hhh", 10), std::runtime_error);
}
```

Ps6 Source Code: MModel.h

```
// Copyright 2020 Karl Marx
#ifndef MMODEL_H //NOLINT
#define MMODEL_H
#include <iostream>
#include <algorithm>
#include <string>
#include <map>
#include <stdexcept>
#include <vector>
```

```
10 #include <utility>
11
12 class MModel {
13  public:
14      MModel(std::string text, int k);
15      int kOrder();
16      int freq(std::string kgram);
17      int freq(std::string kgram, char c);
18      char kRand(std::string kgram);
19      std::string generate(std::string kgram, int L);
20      friend std::ostream& operator<< (std::ostream &os, MModel &model) {
21       os << "Order =" << model.order << std::endl;
22       os << "Alphabet = "<< model.alphabet << std::endl;
23
24       std::map<std::string, int>::iterator temp;
25
26       for (temp = model.kgrams.begin(); temp != model.kgrams.end(); temp++) {
27       os << temp->first << "     " << temp->second << std::endl;
28       }
29
30   return os;
31      }
32
33  private:
34      std::map <std::string, int> kgrams;
35      int order;
36      std::string alphabet;
37      std::string original;
38 };
39 #endif //NOLINT
```

Ps6 Source Code: MModel.cpp

```
1  // Copyright 2020 Karl Marx
2  #include <string>
3  #include "MModel.h" //NOLINT
4
5  MModel::MModel(std::string text, int k) {
6    srand(time(NULL));
7    order = k;
8    original = text;
9
10   for (unsigned i = 0; i < text.size(); i++)
11     if (std::string::npos == alphabet.find(text.at(i)))
```

```
12        alphabet += text.at(i);
13
14    for (unsigned i = 0; i < text.size(); i++) {
15      std::string temp;
16      std::string temp2;
17
18      for (unsigned j = i; j < i + k; j++)
19      temp = (j >= text.size()) ? temp += text.at(j - text.size()) : temp += text.
20
21      kgrams[temp] = (kgrams.end() == kgrams.find(temp)) ? 1 : kgrams[temp] += 1;
22
23      for (unsigned j = 0; j < alphabet.size(); j++)
24        if (kgrams.end() == kgrams.find(temp + alphabet[j]))
25          kgrams[temp + alphabet.at(j)] = 0;
26
27      for (unsigned j = i; j < i + k + 1; j++)
28        temp2 = (j >= text.size()) ? temp2 += (text.at(j - text.size())) : temp2 +
29
30      kgrams[temp2] += 1;
31    }
32  }
33
34  int MModel::kOrder() {
35        return order;
36  }
37
38  int MModel::freq(std::string kgram) {
39    if ((signed)kgram.length() != order) {
40        throw
41           std::runtime_error("invalid kgram in freq(std::string)");
42    }
43    int return_value = (order == 0) ? (original.size()) : (kgrams[kgram]);
44    return return_value;
45  }
46
47  int MModel::freq(std::string kgram, char c) {
48    if ((signed)kgram.length() != order) {
49        throw
50          std::runtime_error("invalid kgram in freq(std::string, char)");
51    }
52    if (order == 0) {
53      int counter = 0;
54      for (int i = 0; i < (signed)original.size(); i++) {
55        if (original.at(i) == c) {
56          counter++;
57        }
```

```
58      }
59      return counter;
60    } else {
61      return kgrams[kgram + c];
62    }
63    return 0;
64  }
65
66  char MModel::kRand(std::string kgram) {
67  if (kgram.length() != (unsigned)order) {
68      throw std::runtime_error("invalid kgram in kRand");
69    }
70  if (kgrams.end() == kgrams.find(kgram)) {
71      throw std::runtime_error("invalid krgram in kRand");
72  }
73    double _freq = 0;
74    double val = 0;
75    for (unsigned int i = 0; i < alphabet.length(); i++) {
76      _freq =  (double)(freq(kgram, alphabet.at(i))) / freq(kgram);//NOLINT
77      if ((double)(rand() % freq(kgram)) / freq(kgram) < _freq + val && _freq != 0
78        return alphabet.at(i);
79      }
80      val += _freq;
81    }
82    return '#';
83  }
84
85  std::string MModel::generate(std::string kgram, int L) {
86    if (kgram.size() < static_cast<unsigned>(order))
87        throw std::runtime_error("Invalid Kgram");
88
89  L = L - order + 1;
90  while(kgram.size() < (unsigned)L)
91      kgram += kRand(kgram.substr(kgram.size()-order, order));
92  return kgram;
93  }
```