# CSC242 – Assignment 5 – 250 Points

**Background**

Complete the following programming projects under Chapter 8. Streams – End-of-Chapter Exercises - Programming Projects - P8.2 and P8.7.

**From the textbook: Big C++: Late Objects, Enhanced:**

**P8.2**

Write a program that checks the spelling of all words in a file. It should read each word of a file and check whether it is contained in a word list. A word list is available on most UNIX systems (including Linux and Mac OS X) in the file /usr/share/dict/words. (If you don’t have access to a UNIX system, you can find a copy of the file on the Internet by searching for /usr/share/dict/words.) The program should print out all words that it cannot find in the word list. Follow this pseudocode:

* Open the dictionary file
* Define a vector of strings called words
* For each word in the dictionary file:
  + Append the word to the words vector
* Open the file to be checked
* For each word in that file:
  + If the word is not contained in the words vector:
    - Print the word.

**P8.7**

Random monoalphabet cipher. The Caesar cipher, which shifts all letters by a fixed amount, is far too easy to crack. Here is a better idea. As the key, don’t use numbers but words. Suppose the key word is FEATHER. Then first remove duplicate letters, yielding FEATHR, and append the other letters of the alphabet in reverse order:



Now encrypt the letters as follows:

A group of black letters

AI-generated content may be incorrect.

Write a program that encrypts or decrypts a file using this cipher. For example:

crypt -d -kFEATHER encrypt.txt output.txt

decrypts a file using the keyword FEATHER. It is an error not to supply a keyword.

**Instructions**

1. After analyzing the problem and specifications described in the background above, in the provided Word document, create the pseudocode (language agnostic and using conventions in the text) that describes the algorithm and logic for the proposed solution to the problem scenario.
2. Verify that the algorithm and logic are well structured (unambiguous, executable, and terminating) based on conventions described in the reading.
3. Create a C++ project in Visual Studio (or your preferred toolset) and translate the algorithm and logic described in pseudocode to a working program tested and compiled.
4. Run the program to ensure that it is working as expected. Copy and paste the finished code into the green box below (in the Submission area). Take a screenshot of the output results (and only the output results; do not include code, other windows, or tool windows) and paste it into the orange box below (in the Submission area).
5. Download the document, add each team member’s name, copy and paste the source code for each problem, snap and paste a screenshot of the output for each problem, and submit the edited document, once for the whole team.
6. Upload this completed document for your submission, one document per team.

**Grading**

This assignment is worth 250 points (150 group points for coding the project and 100 individual points from the peer evaluation) and is due on Sunday. Refer to the rubric for more specifics on how this will be graded.

**Course Learning Outcomes**

1. Perform problem analysis to develop solutions to solve problems and translate the solution into C++ code.

2. Design, write, test, and debug the computer programs as a team.

**Submission**

**P8.2**

Copy and paste the code here. Ensure that the formatting is kept—if it is hard for you to read, it will be hard for the grader:

|  |
| --- |
| Copy and paste your code for this problem |
| #include <iostream>  #include <fstream>  #include <vector>  #include <string>  #include <unordered\_set>  #include <algorithm>  #include <cctype>  // Mary Kilbourne, Anthony Sessoms, and Enrique Gutierrez collaborated on a spell checker based off a dictionary  using namespace std;  // --- From Enrique: Lowercase conversion utility ---  string to\_lower\_str(const string& s) {  string result = s;  transform(result.begin(), result.end(), result.begin(), ::tolower);  return result;  }  // --- From Anthony: Dictionary loader using unordered\_set for fast lookup ---  // --- From Mary: Also stores dictionary in a vector to match pseudocode ---  void loadDictionary(const string& filename, unordered\_set<string>& dict, vector<string>& wordsVec) {  ifstream dictFile(filename);  string word;  if (!dictFile) {  cerr << "Error: Could not open dictionary file: " << filename << endl;  exit(1);  }  while (dictFile >> word) {  word = to\_lower\_str(word);  dict.insert(word);  wordsVec.push\_back(word); // --- Mary’s pseudocode: Append to vector ---  }  dictFile.close();  }  // --- From Anthony & Enrique: Word sanitization ---  string sanitizeWord(string word) {  // Remove leading/trailing non-alpha characters  while (!word.empty() && !isalpha(word.front())) word.erase(0, 1);  while (!word.empty() && !isalpha(word.back())) word.pop\_back();  return to\_lower\_str(word);  }  int main() {  unordered\_set<string> dictionary;  vector<string> words; // --- From Mary: vector of strings called words ---  string dictPath = "dictionary.txt"; // --- From Mary: Open dictionary file ---  string inputPath = "input.txt"; // --- From Mary: Open file to be checked ---  // --- From Mary: Load dictionary words into vector ---  loadDictionary(dictPath, dictionary, words);  // --- From Mary: Open the file to be checked ---  ifstream inputFile(inputPath);  if (!inputFile) {  cerr << "Error: Could not open input file: " << inputPath << endl;  return 1;  }  string word;  while (inputFile >> word) {  string sanitized = sanitizeWord(word);  // --- From Mary: If word not in dictionary, print it ---  if (!sanitized.empty() && dictionary.find(sanitized) == dictionary.end()) {  cout << "Misspelled word: " << word << endl; // --- From Anthony: labeled output ---  }  }  inputFile.close();  return 0;  } |

Copy and paste your screenshot of the output results here. Ensure that only the output is being copied; do not copy the code or surrounding windows:

|  |
| --- |
| Copy and paste your screenshot for this problem |
|  |

**P8.7**

Copy and paste the code here. Ensure that the formatting is kept—if it is hard for you to read, it will be hard for the grader:

|  |
| --- |
| Copy and paste your code for this problem |
| #include <iostream>  #include <fstream>  #include <string>  #include <map>  #include <vector>  #include <array>  #include <cctype>  #include <algorithm>  // Mary Kilbourne, Anthony Sessoms, and Enrique Gutierrez collaborated on a program to encrypt a file  using namespace std;  // --- From Mary: Builds cipher alphabet using keyword + reversed remaining letters  string createCipherAlphabet\_Mary(const string& keyword) {  string cipher = "";  for (char c : keyword) {  if (isalpha(c) && cipher.find(toupper(c)) == string::npos) {  cipher += toupper(c);  }  }  for (char c = 'Z'; c >= 'A'; --c) {  if (cipher.find(c) == string::npos) {  cipher += c;  }  }  return cipher;  }  // --- From Anthony: Builds cipher alphabet using keyword + forward remaining letters  string createCipherAlphabet\_Anthony(const string& keyword) {  string key = "";  string seen = "";  for (char ch : keyword) {  ch = toupper(ch);  if (isalpha(ch) && seen.find(ch) == string::npos) {  key += ch;  seen += ch;  }  }  for (char ch = 'A'; ch <= 'Z'; ++ch) {  if (key.find(ch) == string::npos) {  key += ch;  }  }  return key;  }  // --- From Enrique: Builds cipher alphabet using keyword + reversed remaining letters  string createCipherAlphabet\_Enrique(const string& keyword) {  string key;  vector<bool> seen(26, false);  for (char c : keyword) {  c = toupper(c);  if (isupper(c) && !seen[c - 'A']) {  seen[c - 'A'] = true;  key += c;  }  }  for (char c = 'Z'; c >= 'A'; --c) {  if (!seen[c - 'A']) {  key += c;  }  }  return key;  }  // --- From All: Encrypts a character using the cipher map  char encryptChar(char ch, const map<char, char>& encMap) {  if (isalpha(ch)) {  char upper = toupper(ch);  char result = encMap.at(upper);  return islower(ch) ? tolower(result) : result;  }  return ch;  }  // --- From All: Main function: uses fixed input/output and lets you choose cipher mode  int main() {  string keyword = "FEATHER";  string mode = "mary"; // Options: mary, anthony, enrique  // Choose cipher alphabet based on who made what cipher  string cipherAlphabet;  if (mode == "mary") {  cipherAlphabet = createCipherAlphabet\_Mary(keyword);  }  else if (mode == "anthony") {  cipherAlphabet = createCipherAlphabet\_Anthony(keyword);  }  else if (mode == "enrique") {  cipherAlphabet = createCipherAlphabet\_Enrique(keyword);  }  else {  cerr << "Unknown mode: " << mode << endl;  return 1;  }  map<char, char> encMap;  for (int i = 0; i < 26; ++i) {  encMap['A' + i] = cipherAlphabet[i];  }  ifstream inFile("encrypt.txt");  ofstream outFile("output.txt");  if (!inFile || !outFile) {  cerr << "Error opening files.\n";  return 1;  }  char ch;  while (inFile.get(ch)) {  outFile.put(encryptChar(ch, encMap));  }  cout << "The file has been encrypted.\n";  return 0;  } |

Copy and paste your screenshot of the output results here. Ensure that only the output is being copied; do not copy the code or surrounding windows:

|  |
| --- |
| Copy and paste your screenshot for this problem |
|  |

*Upload this completed document for the Assignment submission, one copy for the whole team.*