**Instructions:**

1. Create a MS VisualStudio solution named **Assignment17**.
2. Create projects according to the assigned problems.
3. Use techniques and experience that you have obtained from previous chapters and this chapter.
4. Code and test your programs. They **MUST** be completed without syntax, logic and run-time errors.
5. Include comments of your name, date, and brief descriptions in all source codes.
6. Compress **Assignment17**folder into **ONE** zipped file.
7. Submit or re-submit your zipped file before its due date&time.

int main()  
{  
   do  
   {  
       switch (menuOption())  
       {  
       case 0: exit(1); break;  
      case 3: Challenge3(); break; //done inclass  
       case 4: Challenge4(); break;  
      case 6: Challenge6(); break; //done inclass  
     case 7: Challenge7(); break;  
       case 8: Challenge8(); break; // 20 points Extra credit  
       default: cout << "\t\tERROR - Invalid option. Please re-enter."; break;  
       }  
       cout << "\n";  
       system("pause");  
  
    } while (true);  
  
   return EXIT\_SUCCESS;

3. Capital Quiz

Write a program that creates a map containing the U.S. states as keys, and their capitals as values. (Use the Internet to get a list of the states and their capitals.) The program should then randomly quiz the user by displaying the name of a state and ask the user to enter that state’s capital. The program should keep a count of the number of correct and incorrect responses. (As an alternative to the U.S. states, the program can use the names of countries and their capitals.)

4. File Encryption and Decryption

Write a program that uses a map to assign “codes” to each letter of the alphabet. For example:

map<char, char> codes =

{ {'A', '%'}, {'a', '9'}, {'B', '@'}, {'b', '#'}, etc ...};

Using this example, the letter ‘A’ would be assigned the symbol %, the letter ‘a’ would be assigned the number 9, the letter ‘B’ would be assigned the symbol @, and so forth.The program should open a specified text file, read its contents, then use the map to write an encrypted version of the file’s contents to a second file. Each character in the second file should contain the code for the corresponding character in the first file. Write a second program that opens an encrypted file and displays its decrypted contents on the screen.

6. Word Frequency

Write a program that reads the contents of a text file. The program should create a map in which the keys are the individual words found in the file and the values are the number of times each word appears. For example, if the word “the” appears 128 times, the map would contain an element with “the” as the key and 128 as the value. The program should either display the frequency of each word or create a second file containing a list of each word and its frequency.

Hint: See Chapter 10 for a discussion of string tokenizing.

7. Word Index

Write a program that reads the contents of a text file. The program should create a map in which the key–value pairs are described as follows:

Key—The keys are the individual words found in the file.

Values—Each value is a set that contains the line numbers in the file where the word (the key) is found.

For example, suppose the word “robot” is found in lines 7, 18, 94, and 138. The map would contain an element in which the key was the string “robot”, and the value was a set containing the numbers 7, 18, 94, and 138.

Once the map is built, the program should create another text file, known as a word index, listing the contents of the map. The word index file should contain an alphabetical listing of the words that are stored as keys in the map, along with the line numbers where the words appears in the original file. Figure 17-9 shows an example of an original text file (Kennedy.txt) and its index file (index.txt).

Hint: See Chapter 10 for a discussion of string tokenizing

A screenshot of a computer

Description automatically generated

1. Prime Number Generation

A positive integer greater than 1 is said to be prime if it has no divisors other than 1 and itself. A positive integer greater than 1 is composite if it is not prime. Write a program that asks the user to enter an integer greater than 1, then displays all of the prime numbers that are less than or equal to the number entered. The program should work as follows:

Once the user has entered a number, the program should populate a vector with all of the integers from 2, up through the value entered.

The program should then use the STL’s for\_each function to step through the vector. The for\_each function should pass each element to a function object that displays the element if it is a prime number.