CS150 Assignment 7
Cryptography

Date assigned: Monday, November 23, 2015
Date Due: Tuesday, December 8, 2015, 9:40 am (40 points)

There is no late grace period for this last assignment!!!!!

Cryptography is an exciting area of Computer Science concerned with hiding and protecting information. This is the branch of Computer Science that allows you to send your credit card securely to a web site or to decrypt or encrypt secret messages.

For this project, you will be building a program that will take a plain text file and produce an encrypted file (or vice versa). The encryption scheme you need to implement is inspired by the Enigma Machine, a German encryption machine from WWII (http://en.wikipedia.org/wiki/Enigma_machine). The Enigma machine code was broken by Alan Turing and his team at Bletchley Park during World War II.

Encryption Scheme:

Simple substitution

A simple substitution cipher (sometimes called a Caesar cipher) maps each letter of the alphabet to another letter of the alphabet as shown below:

Input

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W

Output

To encrypt a letter, find the letter on the top and use the letter below it as the encrypted output. Using the above table, the encrypted output for C would be Z. To decrypt a letter, find the letter on the bottom and use the letter above it as the decrypted output.

To make things more interesting, a secret key is used to specify how the input characters are mapped to output characters. The key, which is a single character, specifies which letter the output row starts on. The key in the above example is X. (The input row always starts with A).

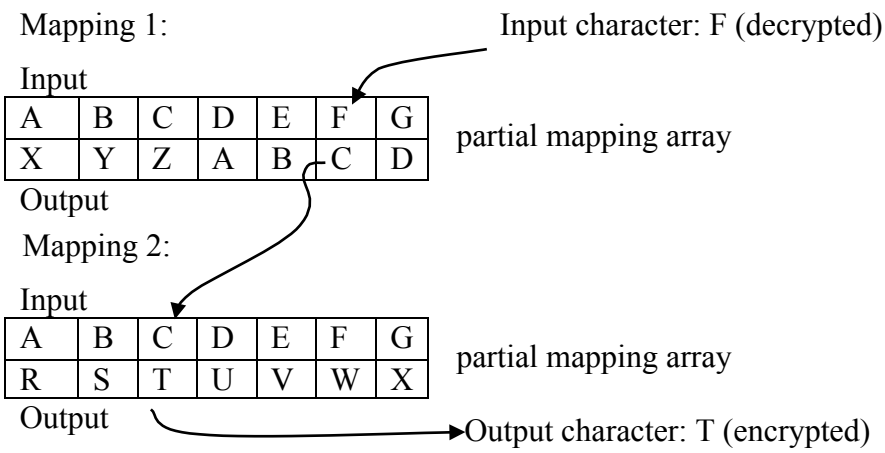
N-Way Substitution

You will need to implement an N-Way substitution encryption scheme. This means you will have N substitution mappings. The character to be encrypted will be used as the input to the first mapping; the output of the first mapping will be the input to the second mapping. The output of the second mapping will be the input to the third mapping, etc. The output of the N-1 mapping is the input to the N mapping, and the output of the N mapping is the encrypted character.

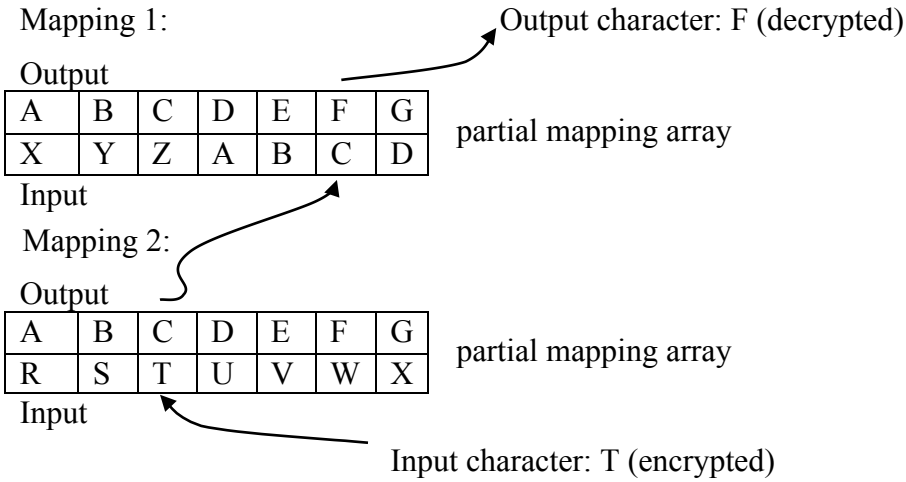
Each mapping has its own key, so you will have a total of N secret keys for this scheme. The value N, which is specified by the user, may be between 2 and 25.

You must use a single, multidimensional array to represent all of these mappings.

Encryption example (N = 2, keys of X and R): The initial input character of F gets mapped to C by the first mapping, and this becomes the input of the second mapping. Under the second mapping, this input character gets mapped to T which is the encrypted output character.



Decryption example (N = 2, keys of X and R): Decryption works in the opposite direction with the inputs and outputs reversed in each mapping. The input character of T gets mapped to C under the second mapping, and this becomes the input of the first mapping. Under the first mapping, this input character C gets mapped back to F which is the decrypted output character.



For this assignment, you need to allow the user to encrypt or decrypt as many files as they want. The user will need to specify the N keys and the input and output files.

Your output must look exactly like the following (user input is in red):

```
********************
* ENCRYPTION
********************

Enter E)ncode, D)ecode, Q)uit: E
Enter Number of Mappings (2-25): 6
Enter the 6 keys: OREGON
Enter plaintext filename: plaintext.txt
Enter ciphertext filename: ciphertext.txt
Encode Success

Enter E)ncode, D)ecode, Q)uit: Q
```

Functions

You must use at least the following functions:

- ```
  void printHeading (const char heading[]);
  ```

  Description: Prints the heading as displayed on the sample output of the previous page.

- ```
  void getMappingData (int &numberOfMappings, char key[]);
  ```

  Description: Prompts the user for the number of mappings and the keys as displayed on the sample output. The number of mappings and keys are returned through the parameters **numberOfMappings** and **key**. To validate the key, you will need to use a string function called strlen, which returns the number of characters in the key. For example, the following program segment will allow the user to enter a key such as **OREGON** and then output 6. You will need to #include <string> for the use of strlen. Remember, do not declare any string variables.

  ```
  cin >> key;  //Typing in OREGON produces a key OREGON\0
  cout << static_cast<int>(strlen (key)); // 6 is output
  ```

- ```
  void openFileForRead (ifstream &inputFile, const char message[]);
  ```

  Description: Accepts a message to be displayed (e.g. Enter plaintext filename:), then prompts the user to enter a plaintext file validating that the file exists and is open. Do not proceed until a proper file can be opened.

- ```
  void openFileForWrite (ofstream &outputFile, const char message[]);
  ```

  Description: Similar to openFileForRead only we are writing to a file.

- ```
  void constructMappingArray (char mappingArray[][LETTERS_IN_ALPHABET],
                              const char keyArray[],
                              int numberOfMappings);
  ```

  Description: Using the keyArray and numberOfMappings, you are to create the mappingArray as discussed above.

- ```
  void printMappingArray (const char mappingArray[][LETTERS_IN_ALPHABET],
                          int numberOfMappings);
  ```

  Description: This function is simply for debugging. After you create the mappingArray, print it out to make sure you have a correct mappingArray. If not, there is no point in going on until this array is correct.

- ```
  char decodeCharacter (char ciphertextCharacter,
                        const char mappingArray[][LETTERS_IN_ALPHABET],
                        int numberOfMappings);
  ```

  Description: Using the number of mappings and the mappingArray, this function takes an encoded character and returns a decoded character using the process described above.
  ```

- ```
  char encodeCharacter (char plaintextCharacter,
                        const char mappingArray[][LETTERS_IN_ALPHABET],
                        int numberOfMappings);
  ```

  Description: Using the number of mappings and the mappingArray, this function takes a plaintext character and returns a ciphertext character using the process described above.

- You can add more functions as you see fit. If you only use the functions described above, your main function might be a little long, say around 65 lines of code.

Part A (Due Wednesday, December 2, 2015 by 1pm)

You are to implement the functions **printHeading**, **getMappingData**, **openFileForRead**, **openFileForWrite**, **constructMappingArray**, and **printMappingArray**. For part A only, display the mappingArray. You do not need to turn in a printed copy of your program for part A. Only an electronic copy is necessary for this part.

Part B (Due Tuesday, December 8, 2015 by 9:40am)

The completed project is due by the date and time specified including a printed copy and an electronic copy.

Notes

- You are to use arrays to help you encode and decode your message.
- Do not declare any string variables in your solution. Only char arrays are allowed.
- Be sure to validate all input. Continue asking for input until valid input is given. The secret keys must be uppercase letters.
- Only upper case alphabetic characters are to be encoded and decoded --any other characters are to be left as is and outputted. This includes any punctuation, spaces, returns, etc.
- You need to encode your favorite joke and email the encoded message to me by 9:40 am the day the project is due. In the subject line of the email, specify all the secret keys you use (at least 5).
- At least one encoded file will be placed on Grace a few days before the assignment is due.

To complete this assignment you must submit the following:

1. **An electronic copy of your program on Grace**
   a. Add a new project named 07_Cryptography to your previously created assignment solution called PUNetIDAssignments. It is *vital* that you name your project correctly!
   b. Type your program (fully documented/commented) into the project. You must follow the coding standards!
   c. Pay attention to the example output! Your program's output must look **exactly** like the example output! The spacing and newlines in your output must match exactly.
  ```

d. Make sure that your program compiles and runs correctly. If you get any errors, double check that you typed everything correctly.

e. Make sure that your program does not produce any warnings.

f. Once you are sure that the program works correctly it is time to submit your program. You do this by logging on to Grace and placing your complete solution folder in the **CS150-01 Drop** folder. This solution folder must contain previous assignments.

g. The solution must be in the drop folder by 9:40am on the day that it is due. Remember, there is no late grace period for this last assignment.

2. A hard copy of your program

a. The hard copy must be placed on the instructor's desk by 9:40am on the day that it is due.

b. The hard copy must be printed in color, double-sided, and stapled if necessary.

c. Your tab size must be set to 2 and you must not go past column 80 in your output.

Good luck! And remember, if you have any problems, come and see me straight away.

START EARLY!!