EE 285 Lab 11

Practice using good techniques to write the programs below. Try to write efficient code. Use comments as needed to explain your code.

A. Writing a File (33 points)

As a simple first exercise with writing files, you will create a Fibonacci series and write the values to a data file. The Fibonacci sequence is rather famous and easily constructed. The first two numbers in the sequence are 0 and 1. To make the third number, you add the first two together to get 1 again. To get the 4th number, you add the 2nd and 3rd numbers together, giving a value of 2. You can continue on in this manner. Each member in the sequence is made by adding together the previous two members. Given this background, write a program that:

1. Declares an integer array that has 40 elements.
2. Fills the array with the first 40 values of the Fibonacci series.
3. Writes the values to a text file called "fibonnaci.dat". In the file, the values should be arranged in 4 rows of 10.
4. Printing the values on the screen (i.e. to `stdout`) is optional. It might not be a bad idea to do this, so that you can see what is going on.
5. Demonstrate your program by running it for the lab instructor and then opening the file and showing that it has the correct sequence in the correct format\

B. Reading a file (33 points)

Something that is frequently useful is to compare two files to see if they are identical.

Write a program that:

1. Reads in two text files. The text files consist of 100 integers arranged in 10 rows of 10.
2. The program should should compare each item in the two files, checking to see if all the corresponding items are the same.
3. If they identical, the program should report that conclusion. If they are not identical, the program should report how many of the 100 items were not the same. In other words, how many mismatches were there?
4. There are four test files available, "test1.dat", "test2.dat", "test3.dat", and "test4.dat" These are in on GitHub under Lab 11 — download them from there and put them in place needed for writing the files for your program. You should run your program for each possible combination of files.
5. Demonstrate your program to your lab instructor.

C. Reading and writing files. (34 points)

Now we will try a bit of grade-school cryptography. There is a text file, message.dat on the GetHub that has a short message in it. Download the file and put it into your workspace.

Write a program that:

1. Reads in the file, which will be called "message.dat". This is available on the GitHub.
2. Encrypts the message by performing a "5-position Caesar Cipher" on the letters of the message. The cipher works by taking each letter of the message and replacing it with the letter 5 positions over in the alphabet. For example, the letter A would be replaced with F, B would be replaced with G, C would be replaced with H, etc. At the other end of the alphabet, the letters rotate back to the beginning — Z would be replaced with E, Y, would be replaced with D, etc. For simplicity, you can assume that all of the letters will be upper-case. (However, you are more than welcome to extend the encryption to include lower-case letters and numbers.) (If you are interested, you can look at the Wikipedia entry for "Caesar Cipher" — it's a real thing used by the real Caesar.)
3. Once the encryption is complete, the encrypted message is printed to a file called "rjxxflj.dat".
4. Demonstrate your program to your lab instructor. Be sure to show the new file that contains the encrypted message.
5. For fun, you can write the companion program that will translate the encrypted file back into readable form.