Program 1

Due Sunday by 11:59pm **Points** 100 **Submitting** a file upload
File Types c, cpp, and java

Vigenere with Cipher Block Chaining

This assignment involves implementing the Vigenere cipher as a block cipher and using the Cipher Block Chaining mode of operation, which we have studied in lecture and in our textbook. Specifically, your program must: (1) read in an input file, (2) strip off any characters that are not alphabetic letters, (3) change upper case letters to lower case, and (4) encrypt the resulting data using the Vigenere cipher as a block cipher in the Cipher Block Chaining mode of operation. The input plaintext file name, the Vigenere key, and the Initialization Vector will be supplied to your program as command line arguments. The input file will be in the format described below. The program output must be in the format described below. The program will be graded according to the Grading Rubric that appears at the bottom of this assignment.

Programming Language

The program must be written in C, C++, or Java, whichever you find more convenient. No other programming or scripting languages are permitted.

Teaming

You may develop by yourself or in teams of two students currently in the course. Teams of more than two students are not permitted. If you choose to work as a team:

1. Both students must submit the ***same*** file on Webcourses. If you do not personally submit a program file, you will get a grade of zero for the assignment. If you are working as a team and the files submitted are different, then both team members will receive 20-point deductions.
2. The source file that you submit must contain the required course header. There will be a 20-point deduction if your source file does not contain the required course header.
3. The course header in the source file that you submit must name ***both*** team members as the authors. If only one of you submits a header identifying both team members, then whoever forgot to name the other team member will receive a deduction of 20 points.

What You Should Submit

Your entire program should be contained in ***exactly one file***. If it is a C/C++ program, the suggested file name is "cbcvigenere.c" or "cbcvigenere.cpp". If your program is written in Java, the file (and hence the main class) must be named "CbcVigenere.java" and must not include a package statement. This is necessary so we can use our test scripts for compiling and testing your program.

Your program source file should have a header identifying you as the program author. The header should use the following form:

```
//-----------------------------------------------------------------
// University of Central Florida
// CIS3360 - Fall 2017
// Program Author: <your name or names, if working as a team>
//-----------------------------------------------------------------
```

Please note:

1. You must submit exactly one file, which must be a .c, .cpp, or .java source code file. We will not accept compiled versions of your program, nor will we accept multi-file programs. We must be able to read your code as you have written it.

2. You may submit as many updated versions of your program file as desired, up to the submission deadline. Do not worry about the number that Webcourses adds to the file name for the second and subsequent submissions. Our graders will know how to handle it.

Command Line Parameters

The program must read in three command line arguments or parameters. If you are unsure what is meant by this, or how to use them, please review the article on this topic in the “Programming Resources” section of this Webcourse.

The programming resources article contains complete programs in C and Java that illustrate how to input and read command line arguments. You are strongly advised to key in the appropriate sample program and to make it work on your system so you understand how it works before developing the program for this assignment. For this assignment, you will only need to read in three strings, but one of them will be a file name.

Please note: Most IDEs, like Eclipse and NetBeans, allow you to configure your program's project to pass a set of command line arguments to the program when it is run. You may wish to use the sample plaintext file, keyword, and initialization vector from this assignment for development purposes. Of course, setting up IDEs in this manner just configures them for one particular set of command line arguments. Once your program works with this one set of arguments, it is more convenient to copy the source code into a new folder on your desktop where you will be able to use different files and parameters by simply typing them and pressing the "Enter" button.

Your program may NOT prompt the user to enter the command arguments, nor may it simply wait for the arguments to be entered, nor may it assume that the arguments will have any particular names or values.

The command line arguments for this program are as follows:

1. The first argument will be the name of the plaintext input file. The file name will be a string. It may or may not have the ".txt" file extension. The file will have no more than 4991 characters in it. Your program must open the file, read it, and process it as described further below. Some sample file names are: "sample7.txt" and "file23" .

2. The second argument will be the Vigenere keyword. This will be a string using only alphabetic characters in lower case. The length of this string will determine the block size for encryption. For example, for the keyword "pizza", the block size will be 5 characters. The maximum length of the keyword will be 10 characters.

3. The third and final argument will be the initialization vector, which will also be a string using only alphabetic characters in lower case. The length of the initialization vector will be the same as the length of the Vigenere keyword. For the Vigenere keyword "pizza", for example, "gevrs" would be an acceptable initialization vector.

When a program is launched from the command line in a Command or Terminal window, the command line arguments will be typed immediately after the program name, separated by spaces. All of this will be entered before the "enter" key is pressed. No brackets or quotation marks are used. For example, we would type "CbcVigenere sample7.txt pizza gevrs" at the command prompt, and only then press the "enter" key to launch the CbcVigenere program for the plaintext file "sample7.txt", the Vigenere keyword "pizza", and the initialization vector "gevrs". To run the same program using the different plaintext file "somefile", the different keyword "pepperoni" and the different initialization vector "qwertyuio", we will type, "CbcVigenere somefile pepperoni qwertyuio" and then press the enter key.

Compiling and Running from the Command Line

Your program must compile and run from the command line. If you are unsure what is meant by this, please review the article on this topic in the “Programming Resources” section this Webcourse.

You should be able to compile your program using one the following commands:

C program: prompt> gcc -lm -o CbcVigenere [your_file_name].c

C++ program: prompt> g++ -lm -o CbcVigenere [your_file_name].cpp

Java program: prompt> javac CbcVigenere.java

Once the program is compiled, you should be able to run your program against several different combinations of input input file, keyword, and IV, but be sure the keyword and IV have the same number of characters. You can also make up your own input file to use for testing. You can also try different keywords and IVs for the ramble.txt file.

You do not need to recompile your program to run different test combinations. This is why we designed the program to use command arguments.

Each program test can be launched using command line parameters in the following form:

C/C++ program in Windows: prompt> CbcVigenere [plaintext_file_name] [vigenere_key] [initialization_vector]

C/C++ program in Linux: prompt> ./CbcVigenere [plaintext_file_name] [vigenere_key] [initialization_vector]

Java program on all systems: prompt> java CbcVigenere [plaintext_file_name] [vigenere_key] [initialization_vector]

Please note: the "[" and "]" brackets in the above command illustrations are for display purposes only. An actual command would look like: "CbcVigenere plain.txt mykey myinit" with no brackets or quotation marks.

Testing Your Program

You are strongly advised to test your program in the same manner as we will grade it. This means (a) that you should be able to compile and run it from the command line within a Command or Terminal window, and (b) that you should use command line arguments. A good test will be to use the sample data file, keyword, and initialization vector as for the sample program below. Your program output should be the same as what appears below.

If you are writing your program in C/C++ and you do not currently have the ability to compile and run your program from within a Command Window, you may wish to review the "Programming Resources" article on installing MinGW for Windows, which will install the gcc and g++ compilers for compiling and running C/C++ programs. There is no cost for either MinGW, or gcc, or g++.

Input File Format

The plaintext file to be encrypted can be any valid text file with no more than 4991 characters in it. Thus, it is safe to store all characters in the file in a character array of size 5000, including any needed padding characters. The characters in the file will be in UTF-8 format. The file will contain the letters to be encrypted but it will also generally contain punctuation, numbers, special characters, and whitespace in it, which your program should ignore.

Your program should encrypt only the alphabetic characters (i.e., letters) in the input file. The program should also convert all upper case letters to lower case. For example, the letter "M" should be converted to "m".

Since all input characters that are not letters should be ignored and all letters should be in lower case, you may wish to store only the lower case letters in your input array (or whatever other data structure you may wish to use).

The following is one of the actual plaintext source files that will be used to test your program. You can get a copy of the file **here**. Please note that the file contains many characters that are not letters, including white space, punctuation, and special characters. Your program must extract only the letters (alphabetic characters) and convert all upper case letters to lower case before performing the encryption processing.

```
CS: the science that deals with the theory and methods of processing information in digital
computers, the design of computer hardware and software, and the
applications of computers.

IT: the development, implementation, and maintenance of
computer hardware and software systems to organize and communicate information
electronically. Abbreviation:  IT

Computers are man-made tools that aid us in solving other problems. A biologist is trying to
figure out how life works, physicists and chemists are trying to figure out how items react
in our universe, mathematicians are trying to figure out rules for man-made systems.

Any research problem that may improve a computer's capability of helping solve a problem or
any research problem that sheds light about a new way to do something with a computer is part
of CS.

Most exciting research: Medical Applications (Expert Systems for Diagnosis,
Remote surgery, nano-devices with computing power to deliver medicine, etc.),
We need help trying to create a comprehensive EMR accessible to the right people only,
Cars that can drive themselves – seems like the best way we know how to solve lots of
problems is by throwing lots of computing power at them, instead of looking for
elegant solutions. This doesn't sound exciting, but it will be exciting when the
results are achieved. (ie Watson)

CS students tend to find jobs where they program at least some.
In the process, they are solving problems.
Challenges: It's impossible to teach all the new languages/toys.
Ultimately, we just need to teach our students how to think, so that they can pick up new
things on their own. Our biggest challenge is getting them to buy into that.

Ethical: Lots, with security etc.
```

Program Operation

Your program must accept the command arguments, extract the alphabetic characters from the input file, convert them to lower case, and then perform the encryption of the lower case characters using the Vigenere keyword and initialization vectors that were supplied as command arguments.

There are two parts to the encryption process for this program: (1) Cipher Block Chaining, and (2) the Vigenere cipher. Both require knowing the block size for the encryption. The block size is simply the number of characters in the Vigenere keyword. It is also the number of characters in the Initialization Vector (IV), since the IV and keyword that you will be given will always have the same length.

Cipher Block Chaining ("CBC")

You will recall that CBC uses the initialization vector (IV) to represent the "previous" block of ciphertext output before the first ciphertext block is computed. To compute the first ciphertext block, we use the formula: $C_1 = E_k (P_1 \oplus IV)$, where C_1 represents the ciphertext for the first block, E_k represents encryption using the key "k", P_1 is the first plaintext block, IV is the initialization vector, and $\oplus$ is the XOR operator. For blocks 2 and all

blocks thereafter, we use the formula $C_i = E_k (P_i \oplus C_{i-1})$, where C_i represents the ciphertext for the i^{th} block and C_{i-1} represents the ciphertext for the previous block.

For this program, we will implement the XOR operator by simply adding, letter-by-letter, the numerical values of the letters in the input block and the IV, taking the result modulo 26 so that each result will also be a valid letter. The numerical values are zero-based, so that the letters a, b, c, ..., z will have the values 0, 1, 2, ..., 25.

Vigenere Cipher Encryption

For this assignment, We will use the Vigenere cipher as the encryption algorithm (E_k in the equations above). We will implement the Vigenere cipher in a way that is similar to the implementation of the XOR operation described above. Specifically, we will perform the encryption by simply adding, letter-by-letter, the numerical values of the letters of the Vigenere keyword and the letters of the output of the XOR operation above, taking the result modulo 26 so that each result will also be a valid letter.

Sample Encryption

As an example, suppose we have the Vigenere keyword "secret" and the initialization vector (IV) "fspqrd". Suppose also that the lower case letters of the cleaned up plaintext file begin with "csthesciencethatdealswith..." . Now, the block size is 6 because that is the length of the keyword (and also the IV), so we must divide the plaintext into blocks of that size. The first two plaintext blocks are therefore "csthes" and "cience". Using comma-separated numbers within square brackets to show the numerical codes for strings, this is how we use Vigenere in Cipher Block Chaining (CBC) mode to encrypt those two blocks:

Block 1 (for plaintext "csthes"):

output of XOR step = "csthes" + IV = "csthes" + "fspqrd" = [2,18,19,7,4,18] + [5,18,15,16,17,3]

= [7,36,34,23,21,21] mod 26 = [7,10,8,23,21,21] = "hkixvv"

output of Vigenere step = "secret" + "hkixvv" = [18,4,2,17,4,19] + [7,10,8,23,21,21]

= [25,14,10,40,25,40] mod 26 = [25,14,10,14,25,14] = "zokozo" = C_1

For Block 2 (for plaintext "cience"), we perform the same operations, except that we use the ciphertext for Block 1 instead of the initialization vector:

output of XOR step = "cience" + C_1 = "cience" + "zokozo" = [2,8,4,13,2,4] + [25,14,10,14,25,14]

= [27,22,14,27,27,18] mod 26 = [1,22,14,1,1,18] = "bwobbs"

output of Vigenere step = "secret" + "bwobbs" = [18,4,2,17,4,19] + [1,22,14,1,1,18]

= [19,26,16,18,5,37] mod 26 = [19,0,16,18,5,11] = "taqsfl" = C_2

For Block 3, we repeat the process, this time using the third plaintext block and C_2, the ciphertext for Block 2. Subsequent blocks are processed similarly.

Padding the Last Block

Because the input plaintext file is divided into blocks, it is possible that the last block is not completely full. In that case, we must add additional characters so that the last plaintext block has the required number of characters. If padding is needed, your program should use the character "x", with integer value 23, for each pad character.

Program Output

All program output should be to the screen (console) and should follow the format of the the following sample program output for the input file shown above. You will observe that the program output contains four parts:

1. Echo of the author's name (or both names if you are working as a team) and the command line arguments (plaintext file name, Vigenere keyword, and initialization vector);
2. Clean Plaintext echo: the alphabetic characters of the plaintext input file, all in lower case, displayed in lines of exactly 80 characters;
3. Ciphertext output: the ciphertext output for the entire clean plaintext input, including encryption of any needed pad characters at the end of the last block. This section is also displayed in lines of exactly 80 characters;
4. Statistics section, reporting the total number of characters in the clean plaintext file (before padding), the block size, and the number of pad characters added, if any.

```
CBC Vigenere by [student-name]
Plaintext file name: ramble.txt
Vigenere keyword: secret
Initialization vector: pdqist

Clean Plaintext:

csthesciencethatdealswiththetheoryandmethodsofprocessinginformationindigitalcomp
utersthedesignofcomputerhardwareandsoftwareandtheapplicationsofcomputersitthedev
elopmentimplementationandmaintenanceofcomputerhardwareandsoftwaresystemstoorgani
zeandcommunicateinformationelectronicallyabbreviationitcomputersaremanmadetoolst
hataidusinsolvingotherproblemsabiologististryingtofigureouthowlifeworksphysicist
sandchemistsaretryingtofigureouthowitemsreactinouruniversemathematiciansaretryin
gtofigureoutrulesformanmadesystemsanyresearchproblemthatmayimproveacomputerscapa
bilityofhelpingsolveaproblemoranyresearchproblemthatshedslightaboutanewwaytodoso
methingwithacomputerispartofcsmostexcitingresearchmedicalapplicationsexpertsyste
msfordiagnosisremotesurgerynanodeviceswithcomputingpowertodelivermedicineetcwene
edhelptryingtocreateacomprehensiveemraccessibletotherightpeopleonlycarsthatcandr
ivethemselvesseemslikethebestwayweknowhowtosolvelotsofproblemsisbythrowinglotsof
computingpowerattheminsteadoflookingforelegantsolutionsthisdoesntsoundexcitingbu
titwillbeexcitingwhentheresultsareachievediewatsoncsstudentstendtofindjobswheret
heyprogramatleastsomeintheprocesstheyaresolvingproblemschallengesitsimpossibleto
teachallthenewlanguagestoysultimatelywejustneedtoteachourstudentshowtothinksotha
ttheycanpickupnewthingsontheirownourbiggestchallengeisgettingthemtobuyintothatet
hicallotswithsecurityetc

Ciphertext:

jzlgaedlrkgbowtunyglnhzkfiwcwkbapraqwqvblxrmlxehxsrgaiccbkjplsduvwqesohvqbuqncuu
qavmerzvcjodhzuhivvfsteqewnpsuxrrvnqnnokfltefgqwzvyotidmvpzzxrekfpocbholymyhksey
glqlstxxqbjvfcmsemcdsjgqjikvvarocmzvjmwenbhyszwscugdpngabzfjdptulsfaeczbchkxhsti
bfjwadhvxhrebzscdkyrlfhwyjaawtsgtfnumkghpnfpzcooxmjhfpiszkyctastcmpquxsfkypckjuv
ythcucuctieuhofocmcmnyjhsecvnduwocqvcvkoqrxdmpvbsivqtzldndekbqcebszvxhlliktifxcu
pbkegsxgkabbbqgxuqnweukmzvqpqrxhphoquetdhlyzepwqwjqggcetaoutbtwewdnowilmrysfebde
qcuxpdcxacnplvnxjnrqboasjxhxcdufvggjkabpoctgkvjjmvqygjesepikilxuhharnxaulzgggnsd
jfvodfzbglgyjfqwmngauevfjccjbalgetxwrxzowpkcessffaiecqlueicaueiciypvzzwyrkwgrqlp
mdcbvtwscyekczgraukjgnrlfvmwvuvcwtbibdhfqfwarnsncavvxmityzosxusdjrpaclixxknhiguh
rdplpdrhxphordqkxvclkvsuygkzwaenqlivajokfvubcqdhuskwvwqnfbcttzcwxypgmpjelohijbqw
ncuytpikwzyzdpcjudykgeatpsoewkjrnbsacitwowwhyaktmrjxtfupgfpxlromumlkswejubprwazj
bfsxierqomfznbkuticcmcogkwiyjtonwgvxgcvythynbnqcnumjgygesottwpskwitpbcuhkoeupqcm
ymdwayydlcsnuynmpnqoxqlzmscvudskoulcpchpazngwzwgqeryomabaxkigjubhtmhjvpjgteugizr
ztdxzgmlluomwvxammfqomluylkzorscfwgxbbnkcelwndnabatoxxxubuitspsqtlzojmudcxjmqyih
ujgrbmetyhdwtqqdyflvmpnpwstkdieddhmdgwmpgivkmgooqrfvurmmezhtuschbsrdcxsxnoccfskv
nkhlvoqzcjdumzpautydxvqfefrgfrevtqnvuvzqlgfmflssljlcysrhelvfmptvrfspoaofsgajgrrk
vtrohqxxkpbetdbmpbccslztrmdizohzupljkyeohffrphkulvzekqtqwqkhbahnoersfeovmxcegssh
nfawfjfrkwptqcgclbcowkgude

Number of characters in clean plaintext file: 1384
Block size = 6
Number of pad characters added: 2
```

Grading

Your submission will be graded using the grading rubric that appears below. Additional deductions will be applied as follows:

1. Late submission: 1 point per minute past the deadline.

2. The source file does not contain the required source header identifying the author or authors: 20 points

3. If teaming and your submission does not mention your other team member: 20 points

4. If teaming and the two team members do not submit the same source file: 20 points each

File Upload | Google Doc | Office 365

Upload a file, or choose a file you've already uploaded.

File: Choose File No file chosen

+ Add Another File

Click here to find a file you've already uploaded

Comments...

Cancel Submit Assignment

Some Rubric

Criteria	Ratings		Pts
The program source code includes a header identifying the author, and it compiles and runs from the command line without crashing.	Full Marks 20.0 pts	No Marks 0.0 pts	20.0 pts
The program accepts the required command line arguments and reports the student's name and the arguments in the program output as required.	Full Marks 20.0 pts	No Marks 0.0 pts	20.0 pts
The program reports the the letters (alphabetic characters) of the input file, all in lower case and displayed in lines of exactly 80 characters, under the heading "Clean Plaintext:".	Full Marks 25.0 pts	No Marks 0.0 pts	25.0 pts
The program reports the correct ciphertext output for the entire clean plaintext input, including any needed pad characters, displayed in lines of exactly 80 characters, under the heading "Ciphertext:".	Full Marks 25.0 pts	No Marks 0.0 pts	25.0 pts
The program reports the correct number of characters in the clean plaintext file, the block size, and the number of pad characters added, if any.	Full Marks 10.0 pts	No Marks 0.0 pts	10.0 pts
		Total Points: 100.0	