Program Submission Instructions:

- You must submit your source code file(s) written only in C, C++, or Java. Do not submit a compiled (executable) version of your code. We must be able to compile your program ourselves so we can run it on the computers used for grading.
- You must submit exactly one file on Webcourses from the assignment page.
- If all of your source code is in one .c, .cpp, or .java file, then you may submit just that one file. That is all we need to compile and run your program.
- If your program is composed of multiple source files, header files, etc., then you must submit a single zip file that contains all files necessary to compile and run your program.
    - If your program is written in C/C++, your zip file must also contain a make file for compiling your program.
    - If your program is written in Java, all needed .java files must be in the default package. Do not use your own package as this will cause our test scripts to fail.

CIS 3360 – Security in Computing
Fall 2014
Program #1 Hill Cipher (100 points)

In this assignment you will write a program that encrypts the alphabetic letters in a file using the Hill cipher where the Hill matrix can be any size from 2 x 2 up to 9 x 9. Your program will take two command line parameters containing the names of the file storing the encryption key and the file to be encrypted. The program must generate output to the console (terminal) screen as specified below.

Command Line Parameters

1. 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 "Resources" section of the Webcourse for this class.
2. The program executable must be named "HillCipher" (mixed case as shown, no spaces or file extension).
3. The program must read in the input file names as command line arguments or parameters. If you are unsure what is meant by this, please review the article on this topic in the "Resources" section of the Webcourse for this class. Please note that the file names can be any text strings, as we will use a number of different sets of files to test your program. Your program may NOT prompt the user to enter the file names, nor may it assume that the files will have any particular names.
4. The **first input parameter** must be the name of the encryption key file, as described below. The **second input parameter** must be the name of the file to be encrypted, as

also described below. The sample run command near the end of this document contains an example of how the parameters will be entered.

5. Your program should open the two files, echo the inputs to the screen in the formats described below, make the necessary calculations, and then output the ciphertext to the console (terminal) screen in the format described below.

Note: If the plaintext file to be encrypted does not have the proper number of alphabetic characters, you must pad the last block as necessary with the letter 'x'. Yes, this is different from what is in our class slides, but it is necessary for us to do this so we can know what outputs to expect for our test inputs.

<u>Encryption Key File Format</u>

The encryption key file will be a text file that contains a single positive integer *n* on the first line, indicating the number of rows and columns in the encryption matrix. The value of *n* can be any number from 2 to 9. The following *n* lines will contain the contents of each row of the matrix, separated by commas.

Note: Matrix values can be greater than 9, so your program must be able to handle multiple-digit numbers.

<u>Format of the File to be Encrypted</u>

The file to be encrypted can be any valid text file with no more than 9991 letters in it. Thus, it is safe to store all characters in the file in a character array of size 10000, including any padding characters. Please note that the input text file will also generally have punctuation, numbers, special characters, and whitespace in it, which should be ignored. You should also ignore whether a letter is uppercase or lowercase in the input file. Thus, you should treat 'A' and 'a' the same in your program.

<u>Output Format</u>

The program must output the following to the console (terminal) screen:

1. The contents of the input key file, in the format shown below.
2. The alphabetic characters contained in the input plaintext file, all in lower case. If you must pad the last block, then the pad characters must also be shown.
3. The ciphertext output produced from running the Hill cipher against the alphabetic characters from the input plaintext file.

The plaintext and ciphertext outputs should should consist of only lowercase letters in rows of exactly 80 letters per row, except for the the last row, which may possibly have fewer characters.

The ciphertext characters should correspond to the ciphertext produced by encrypting all of the letters in the input file, including pad characters, if needed. Please note that only the alphabetic characters in the input plaintext file will be encrypted. All other characters should be ignored.

What to Turn In over WebCourses

You must submit this assignment in the form specified at the top of this assignment.

Program Notes and Hints

Your program must be able to read in an input plaintext file that may contain uppercase letters, lowercase letters, whitespace and non-letter characters. Your program must distinguish between these groups so that only the letters get encrypted. All non-letter characters in the file are simply skipped and not counted as part of the plaintext. Please note that although both upper case and lower case letters will be encrypted, your program should treat them the same, that is, the program should process an upper case input letter the same as the corresponding lower case letter, i.e., it should treat an 'A' the same as an 'a'.

You may find it easier to write a program for this assignment if your program includes separate sections of code or functions/methods to perform the following tasks:

1. Read in the command line arguments.
2. Read in the input key matrix from the file named by the first command line argument. A separate function should then echo the key matrix to the screen in the required format.
3. Read in the input plaintext from the file named by the second command line argument. One way to do this is to write a section of code or function that reads only the upper and lower case letters in the input file into an char array of size 10000, storing only the appropriate lowercase letters in the character array. A separate function should then echo to the screen the characters in this array in the required format.
4. Apply the Hill cipher algorithm to the result of step 3 using the matrix input in step 2, storing ciphertext output in all lower case in a separate array. A separate function should then echo to the screen the contents of the ciphertext array in the required format.

Sample Input Key File

```
3
1,1,6
3,3,1
5,2,7
```

Required Output Format for Sample Key File

```
                 ← matrix dimension is
1 1 6                not echoed
3 3 1            ← entries are separated
5 1 7                by spaces, not commas
```

Sample Input Plaintext File

```
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.

Output of Alphabetic Characters from Sample Plaintext File

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

Output of Ciphertext Produced from Sample Key and Plaintext Files

```
ebxpzniicnxtaafuszpzsofzqhsnydqtlyhdnxpgozbztszsoaygmapndyqrdjfhrbvyqvnvbdatbbrx
wviobovfzckgsfiykistuolszjcdufvezhbztaqjpdkfqvyehbiffhhpnqfvofhqmprlrkfjvcrafzcn
iixvcbsxgxnjfrbsxpnqbdskizildxemzpowhfkistuolszjcdufvezhbztaqjpdkuoerdfrvspvqjve
djdcyhumwddamznmxxrwqwdlwbdnxstclxlwqrjjbzaxwvfopnqxlwbzprxwvioqtbqwszxlroaqrtbo
vtfmbbeanpudndweebjjygmgzkodemspppjdvktbktblbhdycpngyrhiwggozddglxfnzsxcnmeeowwl
hbcyhftxktbqtbvivgftflkxtfjlgsrjxhjzhccdhqoibqugpvddaxoyqaajmjqwsnfvilbqtbvivgft
flkxtfjlgsrjzsuatnzbghzgbniuoerdfspznxawoujmimgzkodvwrnrsucazubhfkiuyrxwviouiqvu
epnoxnhkopwilxxvvsfqbvlqqjfrfpajyyusgdbwufcdetapvromxbcjwyexxzgysrjlrcscyxnwvrny
```

```
eswijvbxrkxiuyrxwviompzbslfhqkaejlavftbquvvowoujmiizlkesxwewixmznwbdelzrwuzfnjla
irbxsvseefvlyrnwkmbjvvnujjonaapdaplfykcrkxkistuofreffxfefpdapakpxiddekrnsrfwyepd
xiifktklbhdyconlofxqoimprlsmvrqbngonhoisagaxmpdfayehjdnmpwerxvhfmsvaxzbovfhafqvv
sonyduoolohagackizrmyjjriyghhkmsikgdznxhjlnpvsfvhkrvvszsdetuoogbfhgzhlhfrejqikxx
kistuofreffxvlccramewpiawpphqbkqmfrerwqnxsgfrrvvpiuwbdbovsdtwuwkkviltpscuxdfrefe
wdzyhqjdtpvftbqusntbsxjygmioielpdksnbifnddeaamrvvtlnbbihikvaejcplacnojdscexbtpey
ehlejsztmkycqpcvnaeumxxyehmgzkkobovcgqzdlvsffremgzkodqogvgipgyowifvwgxnkkyxmpdfa
xrzjbyghclueollxtqeomghhqckpivuimlcysmkhbdewtrodfzgsnbgpvbbihikboyuxnxcbdvgqlrro
myehaauyovceoakzfapfregfbyehflqpparilywqghhjbyusbrjbkgohkjfreyehldojjfferxcbrjow
ijqrjjqikyiesbwmijqamzbhuj
```

Sample Run Command

We will compile your programs from the command line as described in the article on the same subject thay you will find in the “Resources” section of the Webcourse for this class. This is how we will run your program after it is compiled.

```
C or C++ program:
prompt> ./HillCipher samplekey.txt sampleplain.txt

Java program:
prompt> java HillCipher samplekey.txt sampleplain.txt
```

Important Note: Do NOT hard code the file names shown above. They are only examples and we will use diferent file names when we test your program.

Grading Rubric

The total possible score for this program is 100 points. The following point values will be awarded:

10 points:	The program compiles and runs
10 points:	Correct echo of input key file
20 points:	Correct echo of alphabetic characters from input plaintext file, including pad characters, if needed.
20 points:	Program produces ciphertext output
30 points:	The ciphertext output is complete and correct
10 points:	The ciphertext output is formatted correctly (80 characters per line, all lower case)

We will compile your program using the following commands:

```
C program:      prompt>     gcc –o HillCipher [your_file_name].c
C++ program:    prompt>     g++ –o HillCipher [your_file_name].cpp
Java program:   prompt>     javac HillCipher.java
```

Note 1: If you submit a C/C++ program that is comprised of a number of source and/or header files, then your program must compile using the make file you submit, without modification.

Note 2: If you are submitting a Java program, the main class file must be named “HillCipher.java” so that the executable will be simply “HillCipher”, as required.