CS1132 Fall 2013 Assignment 2 due 11/21 11:59pm

> *Adhere to the Code of Academic Integrity.* You may discuss background issues and general strategies with others and seek help from course staff, but the implementations that you submit must be your own. In particular, you may discuss general ideas with others but you may not work out the detailed solutions with others. It is never OK for you to see or hear another student's code and it is never OK to copy code from published/Internet sources. If you feel that you cannot complete the assignment on you own, seek help from the course staff.

When submitting your assignment, follow the instructions summarized in Section 4 of this document.

Do not use the `break` or `continue` statement in any homework or test in CS1132.

1 Cryptography Fun!

Just several decades ago, cryptography existed solely in the domain of national governments and large corporations. However, the advent of the Internet and the wide availability of computers have now made cryptography an indispensable part of everyday digital communications. In this part of the assignment, you will implement one well-known classical ciphers: the Vigenere cipher. This ciphers predates modern cryptography but nonetheless provide a relevant and insightful look at some commonly used cryptography techniques. You will implement it as MATLAB functions that can be used to encrypt and subsequently decrypt messages.

The Vigenre cipher is one of the most well-known ciphers of all time. It is a method of encrypting plaintext using a series of different Caesar ciphers based on the letters of the keyword. The Vigenre cipher can be implemented using what is called a tabula recta, or Vigenre square. It consists of the alphabet written out 26 times in different rows with each row consisting of the alphabet shifted cyclically to the left.The Vigenre square is shown below [1]:

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

[1] http://en.wikipedia.org/wiki/Vigen%C3%A8recipher

For example, the plaintext MATLABISFUN when encrypted using the key CIPHER, gives the ciphertext OIISESKAUBR. The encryption works by first taking the key and repeating it till it matches the length of the plaintext. Subsequently, each character in the plaintext is paired with the character in the key in the corresponding position. The characters of the plaintext correspond to the columns of the Vigenre square and the characters of the key correspond to the rows.For instance, using the example below, the first letter of the plaintext M is paired with the first letter of the key C. The row C and column M of the Vigenre square gives the letter O. This is repeated for the rest of the letters in the plaintext.

Plaintext: MATLABISFUN
Key: CIPHERCIPHE
Ciphertext: OIISESKAUBR

For the purpose of the assignment, we simplify the implementation of the cipher by only allowing the plaintext (and ciphertext) to contain only uppercase letters with no other punctuations, whitespaces or symbols. If the user were to enter invalid characters, the function should remove them. If the user were to enter lowercase letters, they should be converted to uppercase. For example, if the input is MATLAB is fun!, it should be converted to MATLABISFUN.

Additionally, the function that you implement, when given the correct key, should be capable of decrypting any ciphertext that was encrypted using the Vigenre cipher. A parameter in the function should be used to indicate which mode the function is in: encryption or decryption.

Write a function that implements the Vigenre cipher as such:

```
function output = vigenere(input, key, encdec)
% An implementation of the Vigenere cipher
%
% input: Plaintext to be encrypted or ciphertext to be decrypted
% key: key to be used in encryption
% encdec: 0 means encrypt and 1 means decrypt; assume that encdec is always
% 0 or 1
% output: outputs ciphertext when encrypting; outputs plaintext when
% decrypting
```

2 Breaking Bad

Breaking Bad[2] is an American television crime drama series created and produced by Vince Gilligan. Set and produced in Albuquerque, New Mexico, Breaking Bad is the story of Walter White (Bryan Cranston), a struggling high school chemistry teacher who is diagnosed with inoperable lung cancer at the beginning of the series. He turns to a life of crime, producing and selling methamphetamine in order to secure his family's financial future before he dies, teaming with his former student, Jesse Pinkman (Aaron Paul).

One interesting fact is that the title of the series is spelled using the chemical symbols for bromine ("Br"), and barium ("Ba") `https://www.youtube.com/watch?v=F1HNuAE9WdU`. Chemical symbols from the periodic table of the elements also appear in every name in the opening credits: a single capital letter, or letter-pair with only the first letter capitalized, shown in a different color.

[2] http://en.wikipedia.org/wiki/Breaking_Bad

You are provided with the names of the cast in `BreakingBadCastName.txt` and the chemical symbol list `ElementList.txt`. For each cast name, find the first substring that matches one of the elements in the chemical symbol list. Then convert the name by adding quotation before and after the found substring along with an arrow pointing to the corresponding element name. For example, the main character "Walter White" is converted to `"W"alter White -> Tungsten`. The string matching process is not case sensitive and you only need to find the first match for each name. However, you should keep the converted names capitalization the same as the original input.

Your task includes:

- Read the two txt files and store the string information in two cell arrays.
- Write the converted cast names (in the same order as the given file) to a file called `ConvertedCastName.txt`. If no match is found, the converted name should be the same name as the original cast name.

Below is a list of useful built-in functions for handling characters, strings, and files. Use only these built-in functions for handling characters, strings, and files. You may not need all of them. You can of course still use general built-in functions not related specifically to strings and files, such as length, zeros, rem,$\cdots$ etc.

You can use:

- `fopen, feof, fgetl, fclose`
- `strcmp, upper, isspace, isletter`

You must NOT use matlab build in function `find, strfind, findstr`.

Write a function that implements the following:

```
function BreakingBadStyle(castNameFile, elementNameFile, convertedNameFile)
% This function converts a list of name from castNamefile to convertedNameFile
% according to the match in elementNameFile
%
% input:
% castNameFile: the name of the file that contains the cast names
% elementNameFile: the name of the file that contains the chemical symbols.
% convertedNameFile: the name of the file that contains the converted names for
% output.
```

The file ConvertedCastName.txt looks like:

```
"W"alter White -> Tungsten
"S"kyler White -> Sulfur
J"es"se Pinkman -> Einsteinium
"H"ank Schrader -> Hydrogen
M"ar"ie Schrader -> Argon
```

3 Graphics Fun!

In this part you will implement a simplified version of the game `Minesweeper` using Matlab's Graphic functions. For the purpose of this assignment, we will consider an $n \times n$ board with m mines randomly distributed among the squares on the board. Initially, all the squares on the board are covered. The

goal of the game is to uncover all the squares that do not contain mines. The player chooses a square to be uncovered by clicking on it.

With this exercise, we are giving you the freedom to structure the code as you see fit. You should define helper functions as needed to make your solution modular—do not submit one long giant function. The helper functions, i.e., subfunctions, must be saved in the same file containing the main function for the game: `minesweeper.m`. The function takes as input arguments n and m, the dimension of the board and the number of mines, respectively. For scaling purposes, you may assume that n is an integer between 5 and 20. Also, you may assume that $0 < m < n^2$. Here are the requirements:

- the interaction happens only in one figure window
- display the name of the game as the title of the figure and the $n \times n$ board in the figure window
- pick the location of the m mines randomly with equal probability, i.e., each square on the board is equally likely to have a mine.
- to choose a cell on the board, the player clicks in one of the squares[3]. If the square has already been uncovered, ask the player to choose again. Once the player chooses a covered square, display X in the square if the square contains a mine; otherwise, change the color of the square and display in it the number of neighboring squares that contain a mine (a number between 0 and 8).
- at each step display in the title area how many *covered* cells are left on the board (initially there will be n^2).
- the game must stop once a player uncovers a mine or there are only m covered squares on the board; display an appropriate message in the title area.

4 Self-check list

The following is a list of the minimum *necessary* criteria that your assignment must meet in order to be considered *satisfactory*. Failure to satisfy any of these conditions will result in an immediate request to resubmit your assignment. Save yourself and the graders time and effort by going over it before submitting your assignment for the first time. Although all these criteria are necessary, meeting all of them might still not be *sufficient* to consider your submission satisfactory. We cannot list everything that could be possibly wrong with any particular assignment!

- Δ Comment your code! Make sure your functions are properly commented, regarding function purpose and input/output parameters.
- Δ Suppress all unnecessary output by placing semicolons (;) appropriately. At the same time, make sure that all output that your program intentionally produces is formatted in a user-friendly way.
- Δ Make sure your functions names are *exactly* the ones we have specified, *including* case.
- Δ Check that the number and order of input and output parameters for each of the functions match exactly the specifications we have given.
- Δ Test each one of your functions independently, whenever possible, or write short scripts to test them.

[3]Hint: use the Matlab built-in function `ginput` to get the coordinates of the user's click. `[a,b]=ginput(1)` stores the x and y coordinates of a user's mouse click in a and b, respectively.

Δ Check that your scripts do not crash (i.e., end unexpectedly with an error message) or run into infinite loops. Check your script several times in a row. Before each test run, type the commands `clear all; close all;` to delete all variables in the workspace and close all figure windows.

5 Submission instructions

1. Upload files `vigenere.m`, `BreakingBadStyle.m`, and `minesweeper.m` to CMS in the submission area corresponding to Assignment 1a in CMS before the deadline. Late submission is accepted up to 24 hours after the deadline with a 10% penalty.

2. *After grading:* If you resubmit the assignment, upload your corrected files and *be sure to select the* ***Regrade Request*** *option.*