CSE 465/565
Fall 2019
Homework #5
100 points

Submit your code in a zipped-up folder. For each exercise, you will see the points in parenthesis. The first number is the number of points for CSE 465 students; the second number is the number of points for CSE 565 students.

1. (35/25 p) Complete TriDiagonalMatrix.cs. A tridiagonal matrix is a sparse matrix, more specifically a band matrix. An N x N matrix A is called a tridiagonal matrix if a[i,j] = 0 whenever i + 1 <= j or j + 1 <= i. The following is an example of a 6 x 6 tridiagonal matrix:

```
1   2   0   0   0   0
3   4   5   0   0   0
0   6   7   8   0   0
0   0   9  10  11   0
0   0   0  12  13  14
0   0   0   0  15  16
```

Since tridiagonal matrices are sparse, it is important to devise a compact way to store them. The idea is to only store:

- the elements on the main diagonal in an array d;
- the elements directly below the main diagonal in an array a; and
- the elements directly above the main diagonal in an array c.

With this representation, assuming array indices start at 0, an N x N tridiagonal matrix would have the format:

```
a[0]   c[0]   0      0             ...          0             0             0
d[0]   a[1]   c[1]   0             ...          0             0             0
0      d[1]   a[2]   c[2]          ...          0             0             0
...                                ...          ...
0      0      0      0             ...          a[N-3]        d[N-2]        c[N-2]
0      0      0      0             ...          0             a[N-2]        d[N-1]
```

Instead of storing N * N elements, we only need to store 3*N - 2 elements. For instance, for a 31 x 31 matrix, there are 961 elements, 870 of which are zeros. With this approach, we only need to store the 91 elements on the main diagonal and directly above and below it.

Tridiagonal matrices are useful in specifying tridiagonal linear systems of equations, which have many applications, especially in physics (e.g., multistage countercurrent extractor). For more information on tridiagonal matrices, check the website: http://mathfaculty.fullerton.edu/mathews/n2003/Tri-DiagonalMod.html

The TriDiagonalMatrix class implements the IEnumerable interface. The associated enumerator is supposed to iterate over the whole tridiagonal matrix. This is done in a row major fashion, starting with element at row 0 and column 0, and includes all N x N elements of the matrix (i.e., includes the zeros below and above the band). For example, for the 6 x 6 tridiagonal matrix shown above, the enumerator is supposed to iterate over the matrix as follows:

```
1 2 0 0 0 0 3 4 5 0 0 0 0 6 7 8 0 0 0 0 9 10 11 0 0 0 0 12 13 14 0 0 0 0 15 16
```

Scoring:

TriDiagonalMatrix	25 p	15 p
TriDiagonalMatrixEnumerator	10 p	10 p

Compiled as: `mcs TriDiagonalMatrix.cs`
Run as: `mono TriDiagonalMatrix.exe`

There are test files provided for you in files Tester1.cs – Tester13.cs.
Compiled as: `mcs TriDiagonalMatrix.cs Tester1.cs`
Run as: `mono TriDiagonalMatrix.exe`

2. (0/20 p) (Graduate Students only) Complete Solver.cs. Implement a solver for a tridiagonal linear system of equations. For more information, check the website: http://mathfaculty.fullerton.edu/mathews/n2003/Tri-DiagonalMod.html, especially the "Mathematica Subroutine" section and Example 1.

There is a test file provided for you: TesterGrad.cs.
Compiled as: `mcs TriDiagonalMatrix.cs Solver.cs TesterGrad.cs`
Run as: `mono TriDiagonalMatrix.exe`

3. (40/30 p) Complete piglatin.cpp. Write a function, called **ToPigLatin**, which is described below:

- The starter file contains a main test program, as well as the function prototype, which is:

  char* ToPigLatin(char* word);

  This function receives a C-style string (word) as a parameter.

- You are to define this function so that it converts the incoming string (character array) to its "Pig Latin" version, and then have the function return a pointer to the string.
- For our purposes, we will use the following as the rules for translation of a word into "Pig Latin":
  1. A **word** is a consecutive sequence of letters (a-z, A-Z) or apostrophes. You may assume that the input to the function will only be a single "word". Examples: Zebra, doesn't, apple.
  2. If a word starts with a vowel, the Pig Latin version is the original word with "way" added to the end.
  3. If a word starts with a consonant, or a series of consecutive consonants, the Pig Latin version transfers all consonants up to the first vowel to the **end** of the word, and adds "ay" to the end.
  4. The letter 'y' and the letter 'w' should be treated as consonants if they are the first letters of a word, but treated as vowels otherwise.
  5. If the original word is capitalized, the new Pig Latin version of the word should be capitalized in the first letter (i.e. the previous capital letter may not be capitalized any more).
- In the starter file, there is a main() function that is already provided to you for testing. This program prompts the user to type in 5 words, then it calls the function for each of them and prints the converted version of each of the 5 strings. Note that the main() function does all of the output -- your function should do NO output (just the appropriate conversion and return). Do not change the main() program in any way.
- You may assume that a "word" to be entered into the function is no more than 39 characters long.

Sample Runs (user input is underlined, to distinguish it from output):

Sample Run 1:

Input 5 words: Flower yellow bypass apple Igloo

Pig Latin version of the 5 words:

Owerflay ellowyay ypassbay appleway Iglooway

Sample Run 2:

Input 5 words: watch Hamburger Rhythm queen zipper

Pig Latin version of the 5 words:

atchway Amburgerhay Ythmrhay ueenqay ipperzay

4. (25/25 p) Complete piglatin.py. Solve the exercise from point 3 in Python. You will have to implement a function **ToPigLatin(l)** that takes as an argument a list of characters l representing one word (e.g., the list ['a', 'p', 'p', 'l', 'e'] for the string “apple”) and returns a list of characters representing the Pig Latin version of the original word. The list l also has to be changed in place to the Pig Latin version of the original word.