CSE 465/565
Fall 2018
Homework #5
100 points

Submit your code in a zipped-up folder.

1. (40/35 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' should be treated as a consonant if it is the first letter of a word, but treated as a vowel 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: string Hamburger Rhythm queen zippitydoodah

Pig Latin version of the 5 words:

ingstray Amburgerhay Ythmrhay ueenqay ippitydoodahzay

2. (25/20 p) Complete MySerializer.cs. Write C# code to serialize/deserialize objects whose classes are known to contain:

* A default constructor
* All data members are public and are one of the following primitive types only:
  int (Int32), double (Double), bool (Boolean), or string (String)

* Assume that strings are only composed of letters, numbers, and blank spaces, no other characters.

```
class MySerializer {
        public static String Serialize(Object obj);
        public static T Deserialize<T>(String str);
}
```

To be used as:

```
Point p1 = new Point(2, 3);
String str1 = MySerializer.Serialize(p1);
Console.WriteLine(str1);
Point newPt = MySerializer.Deserialize<Point>(str1);
Console.WriteLine(newPt);
```

You may not use any library routine for serialization to perform the serialize/deserialize operations for you. That is, the main work must be performed by your code doing primitive operations and using the Reflection library.

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

3. (35/30 p) Complete UTMatrix.cs. An Upper Triangular square matrix is one whose elements below the diagonal are defined to be 0. For example, the matrix element $M_{r,c} = 0$ if $r > c$. The following is an example matrix of size 4. (In this example, row and column indexing starts at 0.) Note, that $M_{3,2} = 0$.

```
 1   7 -8   0
 0   8 10   9
 0   0  7   3
 0   0  0 -9
```

While it is possible to use a regular 2D array to represent an Upper Triangular matrix, doing so is wasteful with memory. You are to write a C# class definition for representing Upper Triangular matrices, while using a minimum of memory. Complete the code in UTMatrix.cs.

Scoring:

25p - UTMatrix
10p - UTMatrixEnumerator

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

4. (0/15 p – graduate students only) Complete UT3DMatrix.cs. Create a class to represent 3D upper triangular matrices. That is, $M_{i,j,k}$ is non-zero iff $i \leq j \leq k$.

Scoring:
0/15 p - UT3DMatrix
0/0 p - UT3DMatrixEnumerator. **This is not required.**