# PE-7 - Mad Libs

## Objective

Practice working with strings, lists, and input/output to create a Mad Libs game. Using predetermined narratives stored in a text file, gather information from the user to generate a complete story.

## Details:

Refer to “File IO - Text.pdf” on MyCourses

<https://docs.microsoft.com/en-us/dotnet/csharp/how-to/parse-strings-using-split>

<https://docs.microsoft.com/en-us/dotnet/api/system.string.replace?view=netframework-4.8>

1. Download the “MadLibsTemplate.txt” file found on MyCourses. This file is structured with 1 Mad Lib story per line with bracketed placeholders which are to be prompted and replaced with user input. The "\n" sequence indicates where line breaks should be inserted.
2. Your program will use the Text File I/O functions to count how many lines (stories) are in the file. And load all lines into a string array. Note that Windows uses the backslash ("\"), NOT the forward slash ("/") as the folder separator. Place "MadLibsTemplate.txt" in c:\templates, then use "c:\\templates\\MadLibsTemplate.txt" as the path. It is ok to hardcode the file path in your code.
3. Your program will ask the user to enter their name and to choose a story between 1 and N, prompt the user for the inputs, then print the final story with the user’s answers.
4. Declare a resultString to hold the final result.
5. Use String.Split( ) to parse each word out of the chosen story’s template.
6. If the current word is "\n", then concatenate the escape sequence '\n' to resultString.
7. If the current word starts with '{' then that is a prompt for the user.
8. Use String.Replace( ) to replace the “\_” in the prompts with spaces. And do not include the "{}" in the prompt. For example, {Kitchen\_appliance} should prompt the user "Kitchen appliance: "
9. Now that you’ve got that working, let’s add additional functionality to the mix. Start the program by asking the user if they want to play Mad Libs.
   * Validate their data - if the user types anything other than "yes" or "no" (case insensitive), prompt them to try again. The user must eventually type "yes" or "no" to move on with the program.
   * If they choose yes, then play the Mad Libs game. If they choose no, simply say goodbye and end the program.
10. Since it would be really useful to be able to repeat a value that the user previously entered (with "Lucy in the Sky With Diamonds" for example), explain below how you would enhance the template definition to support repeating previous values, and what changes you would make to the structure of the code. Do not actually make these changes yet. (Yes, you will have an extra credit opportunity to make this enhancement in the future!)

Store the user input in another array. If user wants to input once for repeated words, give him an option and use Distinct() to deal with the original array that contains all the words to replace.

Another method is to define an operator that let user get access to previous input, like up arrow.

In both of the solutions, the code only needs to have some small changes in the array definition. Add some new arrays to record the distinct words.

URL of your GitHub project:

https://github.com/whisperers26/IGME206---Homework.git

## Submission:

Make sure you follow the coding standards for all code you create.

Submit this document with your answer to #10 to the appropriate dropbox.

Add, Commit and Push your project to your GitHub repository.

## Optional Challenges:

1. Allow the user to play again in the same run of the program.
2. After the third try of invalid data, give the user a random prompt to “urge” them to try again. For instance:
   1. “Come on David, try harder.”
   2. “That’s definitely not ‘yes’ or ‘no’. Try again.”
   3. “Really? You can’t type yes or no?”
   4. Or whatever you imagine