Learning objectives

- Learn to manipulate strings
- Call methods on objects
- Use and understand object types and primitive types

Overview

We're going to create an OS X command-line app that does some simple string manipulation.

The app will take in two pieces of data from the user:

- a number, to indicate which operation to do
- a string, to operate on

Wrap the whole app in an infinite while loop, so users can do multiple operations.

Be sure to print out the menu of options at the start of each loop of the app.

As you go through the steps of this assignment, be sure to commit regularly, and push your code to GitHub.

Tools

Objective C has no built-in way of getting command-line input, so we'll have to drop down into C for that. Check out this snippet, to get started:

```
// 255 unit long array of characters
char inputChars[255];

printf("Input a string: ");
// limit input to max 255 characters
fgets(inputChars, 255, stdin);

// print as a c string
printf("Your string is %s\n", inputChars);

// convert char array to an NSString object
NSString *inputString = [NSString stringWithUTF8String:inputChars];

// print NSString object
NSLog(@"Input was: %@", inputString);
```

As you go through the assignment, use `NSLog` to print out the values, and memory addresses of the objects you create and modify. Methods on `NSString` result in *new* strings, rather than modifying strings in place. This is because `NSString` is immutable. You could use `NSMutableString` for some of these operations, but not all. In general `NSMutableString` tends to be used quite sparingly in Cocoa.

> **NOTE:** The snippet provided uses `fgets` to get input from the command line. `Fgets` does not mix well with `scanf`, so avoid any calls using `scanf` when you use the snippet above.

Operations

1. Uppercase

Take whatever string the user inputs and MAKE IT LOUDER! Go have a look at the documentation or the header files for NSString, and look for methods that can help with this. Print the resulting string.

2. Lowercase

Take whatever text the user inputs and make it all lowercase. Print the resulting string.

3. Numberize

Take whatever text the user inputs, for example the string "10", and convert it to a number (e.g. the integer 10), if possible. Print the resulting number. Consider what inputs will give a valid number, and if you are able to tell if the conversion was successful or not. There is more than one way to do this conversion.

4. Canadianize

Take the user's input, and append ", eh?" to it. Print the resulting string.

5. Respond

If the user input ends with a question mark, answer "I don't know". If the input ends with an exclamation point, respond with "Whoa, calm down!".

6. De-Space-It

Replace all spaces with "-". Print the resulting string.

Stretch Goals

- Add a few more string manipulations. For example, word count, or punctuation removal.
- Convert everywhere you've used `NSString` to use `NSMutableString` if possible, and convert everywhere you've used `NSMutableString` to use `NSString`, if possible.

7. Reference

https://developer.apple.com/reference/foundation/nsstring
http://nshipster.com/nsrange/