CS367 Lab 7

For this lab you will write two complete C programs as described below. As per Lab **6**, for each Part submit, via **Moodle**, a tar'd and gzip'd directory containing all source code and header files, a Makefile, and a text file containing the output from running the program with the specified inputs.

Part A. For this Part you will write a program that calculates the average of a set of grades given as command line arguments..

1. Write a function that takes a single string argument and returns the floating-point grade contained in that string. If the string does not contain a valid number between **0.0** and **10.0** the function should output an error message and exit the program.

2. Write a function that takes as arguments an array of grades and the number of grades in the array, and which returns the average of those grades. (*Hint:* the number of grades in the array will probably not be the same as the total size of the array).

3. Write a **main** function that takes at least one command line argument (not counting the executable name), where each command line argument is a single grade. The function should calculate a grade average by doing the following:

a. Declare an array of floats that can hold at most **100** grades.

b. Use calls to your function 1 to convert the command line arguments into grades and store the grades in your array.

c. Calculate (using your function **(2)**) and output the average of the input grades.

Test your program by running the following test cases (substituting your executable name for ***myProg***):

```
myProg 10.0 7.5 abc 8.3
myProg 9.5 -4.6 8.0
myProg 9.6 8.0 9.5 10.0 8.4
```

Part B. For this Part you will write a program that performs some string manipulations on command-line arguments. Your program should consist of the following functions:

1. A function named **reverse** that takes a string as its argument and reverses the order of the characters in the string. For example, if this function is passed a string containing "**hello**", the string should contain "**olleh**" after the function call.

2. A function named **replace** that takes three arguments: two characters and a string. The function should replace all instances of the first character argument that occur in the string with the second character argument. The function should return the number of replacements performed. For example, calling the function with '**o**', '**X**', and "**onomatopoeia**" should change the string to "**XnXmatXpXeia**", and the function should return **4**.

3. A function that takes two strings as arguments. The function should do the following, in the order indicated, and with all output clearly labeled.

a. Output the two strings.
b. Output whether or not the first string is the same as the second string.
c. Call your function **replace** on the second string to set all instances of the last character in that string to '**X**'. For example, "**baclava**" should be changed to "**bXclXvX**".
d. Output the number of replacements performed in (**c**) and the resulting second string argument.
e. Call your function **reverse** to reverse the second string argument.
f. Output the resulting second string argument.

4. A **main** function that takes at least two command-line arguments (not counting the executable name). The first command-line argument is the ***keyword string***, and the remaining command-line arguments are the ***candidate strings***. For each of the candidate strings, your **main** function should call your function **3** with the keyword string as the first argument and that candidate string as the second argument.

Test your program by running the following test cases (substituting your executable name for ***myProg***):

```
myProg abc   (hint: this should generate an error)
myProg cacac c ca cacac cac
myProg hello hello baclava abracadabra
```