**COS 120 - Introduction to Computational Problem Solving**

**Lab 04 - Strings & Encryption Applications**

Show the completed code for each of these exercises to your TA, as you complete each one. **You are responsible to be sure the TA has checked off completed exercises for this lab**. Unfinished exercises may be completed and turned in by midnight of lab day only if you attended lab.

**L05-1)** Write a function called ***showASCII*** that accepts a string as an argument. It should print the character and the ASCII value for each character in the string. For instance:

>>> showASCII("ABCD")

A=65

B=66

C=67

D=68

>>> showASCII("abc")

a=97

b=98

c=99

**L05-2)** Write a function called ***printASCIIRange*** that will print the numeric value and the character for every ASCII value between (and including) a low and high range passed as parameters. For instance:

>>> printASCIIRange(65,69)

65-A

66-B

67-C

68-D

69-E

**L05-3)** Write a function ***reverseString*** that accepts a string as an argument and returns the string with the characters in reverse order **without using the .reverse method**. Output should be something like:

>>> reverseString("Hello")

'olleH'

Write the reverse string function in three additional **different** ways. **You may NOT use .reverse()**

**L05-4)** Write a value returning function called ***changeCase*** that accepts a character as an argument and returns the opposite case if it is an alphabetic character or returns the original character if it is not alphabetic. Write your solution **WITHOUT USING the .upper() or .lower() methods**!!! (ASCII values, anyone?) Example calls:

>>> changeCase('A')

'a'

>>> changeCase('b')

'B'

>>> changeCase('&')

'&'

**L05-5)** Using the function ***changeCase*** from the previous exercise, write a new function called ***lowerCase*** that takes a string as an argument and returns the string all in lower case. Write your solution **WITHOUT USING the .upper() or .lower() methods, but making use of the changeCase function you wrote in the previous exercise**!!! For instance:

>>> lowerCase("123 HeLlo BOB #$% abcD")

'123 hello bob #$% abcd'

**L05-6)** Write a function ***formatLongDate*** that will accept a date in the form mm/dd/yy as a string argument. Your program should take the string apart (hint: use an appropriate string function) into month, day and year variables. Return the date in the long format (e.g. **09/30/05** would be output as **September 30, 2005**). You may assume all years ***yy*** are in the 21st century. For instance:

>>> formatLongDate("09/05/99")

September 5, 2099

**L05-7)** One way to encode a string would be to simply convert each character in the string to its ASCII code and then into a string representation of it. But some ASCII codes could be two digits and some three digits, making it hard to figure how to decode the encrypted string. One way to overcome this challenge is to make each ASCII code value into a 3 digit string representation (in other words, pad the string with leading 0's if it is too short). Write a function ***stringToASCIICodesString*** that will convert a string argument into a string of 3 digit codes as illustrated below:

>>> stringToASCIICodesString("BAD")

'066065068'

**L05-8)** Write a function called ***ASCIICodesStringToString*** that will accept a string of 3 digit codes which represent ASCII values (all padded to three digits with leading 0's when needed). The function should return the string equivalent assuming each 3 digit code can be converted to an ASCII value and from there into a letter. For instance:

>>> ASCIICodesStringToString("066065068")

'BAD'

**L05-9)**  Create your own **encryption** function (myEncrypt) which simply calls the **transposition2RailEncrypt** (we wrote it in class as the **scramble2Encrypt** function) algorithm 3 times, and returns the 3x encrypted string:

>>> myEncrypt(“Hello Bob!”)

ooe!BlHb l

def transposition2RailEncrypt(plainText): #2 rail trans. encryption

evenChars = ""

oddChars = ""

for i in range (len(plainText)):

if i % 2 == 0:

evenChars = evenChars + plainText[i]

else:

oddChars = oddChars + plainText[i]

cipherText = oddChars + evenChars

return cipherText