CSCI U530 Assignment 03

There are two parts to this assignment. The first part will focus on your ability to iterate through a list, deconstruct the list, peform operations on that, and construct a new list. The second part will focus on your ability to use the more advanced concepts of functional programming (e.g. lambda expressions, maps, higher-order functions). You may NOT manually parse the list for part 02. Each part is described below. You should create the two parts in separate files. Part one will be named **YourLastNamePart01** and part two will be named **YourLastNamePart 02**. You should turn the two files into Blackboard before the due date has passed. **Programs that are late or do not meet proper syntax will receive a zero. Programs that do not produce the expected output will receive a best grade of 50% (further deductions may be made depending how close you are to solving the problem).**

You must program both parts using functional programing methods discussed in class. You should not be using any variables or loops (instead focus on functions and recursion). *Hint: if you are using let, set, setq, setf you are moving in the wrong direction and should find alternative routes.*

# Part 01

You are trying to send an encoded message and you decide to use the following method described below.

In order to encode the message, you need the message you wish to encode and a key. You will start by adding the integer value of the first character in your message to the integer value of the first character in your key. Then, repeat for the rest of the message. If you run out of characters in your key before you are finished encoding your message, you just go back to the first character in the key.

To make this somewhat simpler, you only need to encode lower case letters. If something that does not fall into the range of a lower case letter is in the message, just include the original character in your encoding.

If the resulting integer value for the encoded character is too large (i.e. it is no longer in the range of a lowercase letter), you will subtract 26 from the result until it is a valid character integer.

For example, if the message is “hello”, and the key is “abc”, the encoding will work as follows:

h (104) e (101) l (108) l (108) o (111)

+ a (97) b (98) c (99) a (97) b (98)

---------------------------------------

201 199 207 205 209

(subtract 26 until less than or equal to z (122))

a (97) y (121) g (103) e (101) i (105)

So the encoded message is “aygei”.

You should have a method called encode1 and should be called in the following way:

(print (encode1 "hello" "abc"))

Using that statement, you should get “aygei” printed to the screen.

Additionally, you will need to create a decode1 function that will take an encoded message and a key and decodes the message.

Additional tests are provided below. When you turn in your program, you should have the following at the bottom of your program (below all your functions).

(print (encode1 "hello" "abc"))

(print (encode1 "lisp" "common"))

(print (encode1 "This is a test!" "key"))

(print (encode1 "CSCI 530 is the best class ever!" "brownnose"))

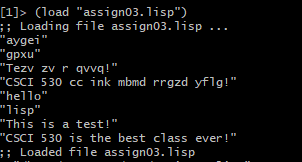
(print (decode1 "aygei" "abc"))

(print (decode1 "gpxu" "common"))

(print (decode1 "Tezv zv r qvvq!" "key"))

(print (decode1 "CSCI 530 cc ink mbmd rrgzd yflg!" "brownnose"))

With the code above in your program, here is a screenshot of what you should get for output.



# Part 02

The encoding and decoding mechanism are the same, only for this part you can assume your message and key are of the same length. For this part, you must use your choice of mapping, lambdas, and/or higher-order functions. In other words, you should not be iterating through the list manually.

Include the following tests at the bottom of your program (below your function definitions):

(print (encode2 "hello" "abcde"))

(print (encode2 "goodbyes" "sayonara"))

(print (encode2 "functional" "procedural"))

(print (encode2 "Testy Testerson" "blahblahblahbla"))

(print (decode2 "ayghl" "abcde"))

(print (decode2 "rhfkhrol" "sayonara"))

(print (decode2 "neuxqebxtp" "procedural"))

(print (decode2 "Tilts Temxxrmsg" "blahblahblahbla"))

Sample output for the tests above is included below:

