UNIVERSITY OF CALIFORNIA, MERCED

BERKELEY · DAVIS · IRVINE · LOS ANGELES · MERCED · RIVERSIDE · SAN DIEGO · SAN FRANCISCO

SANTA BARBARA · SANTA CRUZ

CSE 15: Discrete Mathematics
Laboratory 9

Fall 2020

- **This lab assignment will be graded and its grade will count towards the course grade**.
- **This lab must be solved individually.** You can discuss your ideas with others, but when you prepare your solution you must work individually. Your submission must be yours and yours only. No exceptions, and be reminded that the CSE academic honesty policy discussed in class will be enforced.
- Your solution must be exclusively submitted via CatCourses. Pay attention to the posted deadline because **the system automatically stops accepting submissions when the deadline passes. Late submissions will receive a 0**. You can upload one or more `.py` files.
- Start early.

Introduction

In this lab you will implement simple cryptography functions using the Caesar Cipher (also known as shift cipher). Before we get to that, it is necessary to assign numerical values to letters of the alphabet. In class, for sake of simplicity, we mapped the 26 letters in the English alphabet to the set $\{0 \ldots 25\}$. This of course limits our ability to represent lower case letters, spaces, numbers, and punctuation symbols.

The American Standard Code for Information Interchange, abbreviated as ASCII, provides one possible mapping between characters and integers. If you are unfamiliar with ASCII codes, take a look at `https://en.wikipedia.org/wiki/ASCII`, but essentially it is an encoding scheme that assigns integer values to characters. There are 95 printable ASCII characters, and they are from 32 to 126. For example the character "A" has an ASCII value of 65, "B" is mapped to 66, and "a" is 97. The space character is 32. The Wikipedia article referenced earlier shows the entire table.

The Python language provides functions for converting characters to ASCII values and vice versa. To get the ASCII value of a character, use `ord(c)`. For example `ord('A')` returns 65, and `ord('a')` returns 97. There is also a way of converting an integer to its corresponding ASCII character. This is done with the `chr(n)` function. For example, `chr(98)` returns `'b'`.

So we will use the `ord()` and `chr()` functions to convert from letters to numbers and vice versa, which means that our alphabet has 127 characters, but the first 32 characters are not printable, so we need to exclude them. We will therefore convert a character to a number by taking its ASCII value and subtracting 32. That way we go from a range $\{32 \ldots 126\}$ to $\{0 \ldots 94\}$. Follow a similar process when converting from a number back to a character. Our shift ciphers will then be using modulo 95.

Questions

In class we have discussed the Caesar (shift) cipher, that given a natural number $0 \leq p \leq m-1$ maps it a new natural number in the same range using the relationship:

$$f(p) = (p + k) \bmod m$$

where k is a given constant.

1. Write a function `encode(t,k)` that receives a list of characters `t`, encodes it using a shift cipher with key `k`, and returns the encoded list of characters. To simplify your task, you can assume that the list only includes characters with ASCII code between 32 and 126.
   Note: based on what we discussed in the introduction, $m = 95$, i.e., that is the number of different characters to encode. Each character must be translated into numeric, shifted down by 32, transformed with the encoding function, shifted up by 32 characters and then converted back into a character.

2. Write a function `decode(t,k)` that receives a list of characters `t` encoded with the function `encode` and the key `k` and recovers the original text (returned as a list of characters)

Note: A more natural way to represent plain or encoded text would be using strings, but we use lists because strings have not been covered in the lab with sufficient depth.