Computational Problem Solving CSCI-603
Secret Messages Lab 4

1 Problem[1]

You would like to send messages back and forth with a friend (or co-consiprator!) but want to make sure that other people cannot easily read those messages. However, rather than use a fixed encryption scheme, you decide to take your message string and apply a series of transformations to it to generate the encrypted message.

The transformations you have agreed to use are the following:

- σ_i shifts the letter at index i forward one letter in the alphabet. So, BALL $\rightarrow \sigma_0 \rightarrow$ CALL.
  This can be applied multiple times to shift multiple letters forward, and if so would be designated σ_i^k to shift letter i by k forward. If the shift takes the letter past the end of the alphabet, it will wrap around. Negative exponents shift the letter backward in the alphabet.
- ρ rotates the string one position to the right. So, TOPS $\rightarrow \rho \rightarrow$ STOP.
  This function can also be used with an exponent (positive or negative). For example, TRAIN $\rightarrow \rho^2 \rightarrow$ INTRA.
- δ_i duplicates (in place) the letter at index i. So, HOPED $\rightarrow \delta_2 \rightarrow$ HOPPED. This can also be used with a positive exponent to produce multiple duplicates, but not with negative exponents.
- $\tau_{i,j}$ swaps the letters at index i and index j. So, SAUCE $\rightarrow \tau_{0,3} \rightarrow$ CAUSE. You can assume that $i < j$.
- $\tau_{i,j}^{(g)}$ operates a little differently. In this case, we conceptually divide the string to g equal-sized groups of letters, and then swap *groups* i and j. So, BACKHAND $\rightarrow \tau_{0,2}^{(4)} \rightarrow$ HACKBAND.

To more effectively obfuscate your message, you can go through several transformations. For example, CANAL $\rightarrow \rho^2\,\delta_2\,\sigma_2^9 \rightarrow$ ALLCAN.

---

1. This problem inspired by a puzzle by Dan Katz
(`http://web.mit.edu/puzzle/www/2007/puzzles/transmogrifiers/`)

1.1 Problem-solving Session (20%)

You will work in a team of three or four students as determined by the instructor. Each team will work together to complete the following activities.

NOTE: Please try to solve all questions. However, you may solve any two out of questions 4, 5, and 6 for full credit (if correct!).

1. What are the results of the following transformations?
   (a) ZOO $\rightarrow \sigma_0^2 \rightarrow$
   (b) SUCES $\rightarrow \delta_2\,\delta_5 \rightarrow$
   (c) HORSE $\rightarrow \tau_{2,4}\,\sigma_4\,\rho \rightarrow$
   (d) ANSWER $\rightarrow \rho^3\,\tau_{0,1}^{(2)} \rightarrow$

2. Consider the simple rotation (ρ) operation. For a given string `str`, how would you implement this operation in Python? Write this as a function `rotate(str)`.

3. Modify your previous answer so that your function takes a positive exponent as an additional argument and performs the appropriate operation.

4. Python provides the `ord` function to convert a single character to an integer ASCII value, and the `chr` function to convert from an ASCII integer to a character. With that in mind, show how to implement shifting of a single uppercase character `c` by a positive amount k, keeping in mind that if this takes you past the end of the alphabet, it should wrap around.

5. The operation σ_i^k will be represented in the input to your program with the text string `Si,k` — so, for example, the operation of question 1(a) would be given to your program as `S0,2`. If you are given that operation string (in a variable `s`) and a message string `m`, show how to generate the correct call to the function `shift(message,index,exponent)`. Keep in mind that `i` and `k` can be arbitrary integers!

6. You will want to be able to decrypt as well as encrypt! Luckily, many of the functions can be inverted in a simple fashion. For example, if you get the information $\rightarrow \sigma_0 \rightarrow$ CALL from your colleague, you can retrieve the original message by applying σ_0^{-1} to CALL. For the following encryption operations, show (if possible!) the operation(s) to perform the decryption.
   (a) σ_2^3
   (b) ρ^2
   (c) δ_1
   (d) $\tau_{2,4}$
   (e) $\rho\,\sigma_1$

1.2 Implementation (80%)

You will work with your partner on this assignment and deliver one implementation of the program `transformer.py`.

In the implementation, your program will read in two files (the names of which should come from user input). One file will represent your messages (one message per line) while the other will read the sets of transformations, also one per line. It will also ask the user whether the messages are to be encrypted using the given transformations or decrypted (that is, the given transformations have been applied to generate the messages).

The format of the messages will be all upper-case letters (no spaces or punctuation). The format of the transformation strings will be as follows:

Operation	String form	Example	
σ_i	`Si`	σ_0	`S0`
σ_i^k	`Si,k`	σ_2^{-5}	`S2,-5`
ρ	`R`	ρ	`R`
ρ^i	`Ri`	ρ^{-3}	`R-3`
δ_i	`Di`	δ_2	`D2`
δ_i^k	`Di,k`	δ_2^3	`D2,3`
$\tau_{i,j}$	`Ti,j`	$\tau_{2,4}$	`T2,4`
$\tau_{i,j}^{(g)}$	`T(g)i,j`	$\tau_{0,2}^{(4)}$	`T(4)0,2`

If a series of transformations are to be supplied, they will be separated by semicolons. So, for example, if you were asked to encrypt the string HORSE given the transformation string `T2,4;S4;R` you would generate the string SHOES.

Finally, in order to improve the level of secrecy between you and those you exchange a message with, we are asking you to create one additional operation of your choice. The operation must be decryptable as well. Clearly describe in a comment how your operation works and how it is represented in the operation string, and provide an example.

1.3 Grading

- Functionality: 75%
  - File reading: 10%
  - Individual encryption operations: 40%
  - Overall encryption process: 10%
  - Decryption: 15%
- Code Style and Documentation: 5%

1.4 Submission

Transfer your program to the CS machines and submit your program before the deadline using `try`:

```
try grd-603 lab4-1 transformer.py
```