Encode/Decode Affine: Programming Project CMSC 456
Morally DUE Sep 21

1. (50 points) This is a programming problem. The final goal will be to have programs that encode and decode the affine cipher

   You will write several programs. You will only run some of them. Hence many of the parts say to write a program but are not worth points. Instead those programs are used to help write other programs. ADVICE: when you write a program TEST IT A LOT to make sure it works.

   We will call a pair of naturals (a, b) *cool* if $a, b \in \{0, 1, \ldots, 25\}$ and a is relatively prime to 26.

GOTO THE NEXT PAGE FOR THE ACTUAL ASSIGNMENT

(a) (0 points) Program INV26 (inverse). Given (a, b), both in $\{0, \ldots, 25\}$
(i) If a does not have an inverse mod 26 then output NOT COOL,
(ii) If a does have an inverse mod 26 then output (c, d) such that if $f(x) = ax + b \pmod{26}$ and $g(x) = cx + d \pmod{26}$ then $g(f(x)) \equiv x \pmod{26}$. (So if T is coded with f to produce T' then g applied to T' returns T.)

NOTE: You can do this the brute force way by having a table of all numbers mod 26 that have an inverse and what the inverse is.

(b) (0 points) Program L2N (Letters to Numbers): Take a text T of letters. Our intention is that T be a normal english text, like

Alan Turing cracked the Enigma code. He also made big contributions in logic. He was awesome!

(Though ours will be longer.)

(1) eliminate all punctuation, numbers, and whitespace,

(2) replace a and A with 1, ..., replace z and Z with 26. This program will be used to prepare T to be coded by the affine cipher.

EXAMPLE: On input *I'm Bill* the output is *9 13 2 9 12 12*.

NOTE: We use $\{1, \ldots, 26\}$ not $\{0, \ldots, 25\}$, but since these programs are not expected to produce output your code can use either representation.

(c) (0 points) Program N2NAFF (Numbers to Numbers AFFINE): Take a text T of natural numbers in $\{1, \ldots, 26\}$ (the intent is that this T was produced by L2N) and a pair (a, b). If (a, b) is NOT a cool pair then output NO. If (a, b) is a cool pair then, apply the function x goes to $ax + b \pmod{26}$ to T to get text T' which we will later call $T_{a,b}$.

EXAMPLE: on input *9 13 2 9, 12 12* and $(3, 1)$,

output is *2 14 7 2 11 11*.

(d) (0 points) Program N2L (Numbers to Letters): Take a text T of natural numbers in $\{1, \ldots, 26\}$ (the intent is that this T was produced by N2NAFF) and replace 1 with A, 2 with B, ..., 26 with Z.

EXAMPLE: on input *2 14 7 2 11 11*, output is *BNGBKK*.

GOTO NEXT PAGE

(e) (25 points) Program EA (Encode with Affine): Take a text T of letters (our intention is that T be a normal english text, like

Alan Turing cracked the Enigma code. He also made big contributions in logic. He was awesome!)

and a pair (a, b). Run L2N, N2NAFF, N2L. Note that this program encodes a text using the affine cipher.

EXAMPLE: On input *I'm Bill*

this program will first produce *9 13 2 9 12 12*,

and then *2 14 7 2 11 11*

and then *BNGBKK*.

(f) (25 points) Program DA (Decode with Affine): Take a text T of letters (our intention is that T is the output of EA) and a pair (a, b). First run L2N to get T'. Then find c, d such that $f(x) = ax + b$ and $g(x) = cx + d$ are inverses. Then run EA with T' and (c, d). Then run N2L. If the input was a text that was coded with (a, b) then the output should be the English text.

EXAMPLE: on input *BNGBKK* (3,1) we find that the inverse of $f(x) = 3x + 1$ is $g(x) = 9x + 17$.

Run L2N on *BNGBKK* to get *2 14 7 2 11 11*.

Apply g to that to get *9 13 2 9 12 12*.

Run N2L to get *IMBILL*.

ADVICE: You can test this yourself! Use EA to code a text and DA to decode the text and see that you get the same thing (ignoring spaces). For example if you take *I'm Bill* and EA and DA you should get *IMBILL*.

GOTO NEXT PAGE FOR HOW TO SUBMIT

(a) The deliverable for this project is two programs. One will take as input some plaintext and output ciphertext, the other will take as input ciphertext and output the decoded plaintext. You should upload a single file for each program ending in `.java`, `.py`, `.ml`, `.rb`, `.c`, `.cpp`, or `.scala`, corresponding to Java, Python3, OCaml, Ruby, C, C++, and Scala respectively. If applicable, use the default package.

(b) This problem will be autograded. There will be two separate assignments for this problem on Gradescope – upload your programs here. You have unlimited submissions.

(c) Note that the programs both have inputs of a text of letters in $\{A, \ldots, Z\}$ and outputs a text of letters in $\{A, \ldots, Z\}$.

(d) Expect raw input data to potentially be a large text with special characters, both uppercase and lowercase letters, new-lines, tabs, spaces, etc. Output text should have the output with only the letters remaining (case-insensitive).

(e) Inputs are everything read through standard input (stdin) and outputs should be printed to standard output (stdout). Your programs should only be able to encode/decode one text at a time – multiple lines are still treated as part of the same text.

(f) Command line arguments will be used to specify pair (a, b). Expect the name of your program to be the first command line argument, value a to be the second, and value b to be the third.

The position of the arguments is constant, and you may assume the given (a, b) is a cool pair.