One-Time Pad with LCG: Programming Project CMSC 456
Morally DUE Sep 28

1. (40 points) This is a programming problem. The final goal will be to have programs that encode and decode using the psuedo one-time-pad as on the slides (the one that uses a Linear Congruential Generator).

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.

GOTO THE NEXT PAGE FOR THE ACTUAL ASSIGNMENT
(a) (0 points) Program COOL: Given \((x_0, A, B, M)\), determine if the tuple is cool.
We will call a tuple of \((x_0, A, B, M)\) cool if
i. \(1 \leq x_0, A, B \leq 9999\).
ii. \(1000 \leq M \leq 9999\).
iii. \(A, M\) are relatively prime.

(b) (0 points) Program L2N (Letters to Numbers, slightly different then the other L2N since we use 01 instead of 1, 02 instead of 2, etc.)

**You must use** \(\{A = 01, B = 02, ..., Z = 26\}\)
Take a text \(T\) of letters (our intention is that \(T\) be a normal english text, like the first paragraph of the Declaration of Independence) and then
i. eliminate all punctuation, numbers, and whitespace, and
ii. replace \(a\) and \(A\) with 01, ..., replace \(z\) and \(Z\) with 26.
This program will be used to prepare \(T\) to be encoded by ENCODE-N2NLCG.
EXAMPLE: On input I’m Bill the output is 09 13 02 09 12 12.

(c) (0 points) Program N2L (Numbers to Letters, slightly different then the other N2L since we use 01 instead of 1, 02 instead of 2, etc.)

**You must use** \(\{01 = A, 02 = B, ..., 26 = Z\}\)
Take a sequence of numbers and then replace 01 with \(A\), ..., replace 26 with \(Z\).
This program will be used after DECODE-N2NLCG in order to obtain our final decoded text.
EXAMPLE: On input 09 13 02 09 12 12 the output is IMBILL.
(d) (20 points) Program ENCODE-L2NLG (Encode Letters to Numbers LCG): Take a text $T$ of letters and a tuple $(x_0, A, B, M)$.
   i. Run COOL on the tuple. If it’s not cool, output only "not cool man!".
   ii. Else, use L2N.
   iii. Then, apply the Pseudo 1-time pad.

and finally return this sequence of encoded, two-digit natural numbers.

EXAMPLE:
Input: $T = "\text{Hello World!}"$
$(x_0, A, B, M) = (1, 1000, 2000, 9001)$
Output: "38 05 58 79 79 05 47 98 68 30"

(e) (20 points) Program DECODE-N2LLCG (Decode Numbers to Letters LCG): Take a text $T$ of a sequence of natural numbers and a tuple $(x_0, A, B, M)$.
   i. Run COOL on the tuple. If it’s not cool, output only "not cool man!".
   ii. Else, decode the Pseudo 1-time pad.
   iii. Use N2L to obtain the decrypted text.

and finally return the decrypted text. This should be a text of English.

EXAMPLE:
Input: $T = "38 05 58 79 79 05 47 98 68 30"
$(x_0, A, B, M) = (1, 1000, 2000, 9001)$
Output: "HELLOWORLD"

GOTO NEXT PAGE FOR SUBMISSION DETAILS
1. The deliverable for this project is two programs, encode and decode.

2. Text $T$ should be read through standard input (stdin).
   
   For encode, expect this raw input data to potentially be a large text with special characters, both uppercase and lowercase letters, newlines, tabs, spaces, etc.
   
   For decode, you can assume your input will be solely a sequence of encrypted two-digit numbers, separated by spaces, spanning a single line.

3. You should print your result to standard output (stdout).
   
   For encode, you should output only a sequence of space-delimited, two-digit numbers.
   
   For decode, you should output only the decoded text without spaces, special characters, newlines, numbers, etc. (case-insensitive).

4. Tuple $(x_0, A, B, M)$ will be given to you as command line arguments. Expect your filename as the first argument, $x_0$ as the second, $A$ as the third, $B$ as the fourth, and $M$ as the fifth.
   
   Unlike the encode/decode affine assignment, you are not guaranteed $(x_0, A, B, M)$ is cool. If it’s not cool, print to stdout ”*not cool man!*” and do not attempt to apply the 1-time pad.

5. 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. There will be two separate gradescope assignments for each file.