Tracking State

PROBLEM: DECODE A MESSAGE

A message has been encoded as a text stream that is to be read character by character. The stream contains a series of comma-delimited integers, each a positive number capable of being represented by a C++ `int`. However, the character represented by a particular integer depends on the current *decoding mode*. There are three modes: *uppercase*, *lowercase*, and *punctuation*.

In *uppercase* mode, each integer represents an uppercase letter: The integer modulo 27 indicates the letter of the alphabet (where 1 = A and so on). So an input value of 143 in uppercase mode would yield the letter *H* because 143 modulo 27 is 8 and *H* is the eighth letter in the alphabet.

The *lowercase* mode works the same but with lowercase letters; the remainder of dividing the integer by 27 represents the lowercase letter (1 = a and so on). So an input value of 56 in lowercase mode would yield the letter *b* because 57 modulo 27 is 2 and *b* is the second letter in the alphabet.

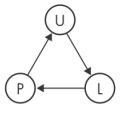
In *punctuation* mode, the integer is instead considered modulo 9, with the interpretation given by Table 2-3 below. So 19 would yield an exclamation point because 19 modulo 9 is 1.

At the beginning of each message, the decoding mode is uppercase letters. Each time the modulo operation (by 27 or 9, depending on mode) results in 0, the decoding mode switches. If the current mode is uppercase, the mode switches to lowercase letters. If the current mode is lowercase, the mode switches to punctuation, and if it is punctuation, it switches back to uppercase.

Original input:
18,12312,171,763,98423,1208,216,11,500,18,241,0,32,20620,27,10

(a)	(b)	(c)	(d)	(e)
18	U	27	18	R
12312	U	27	0	→ L
171	L	27	6	i
763	L	27	7	g
98423	L	27	8	h
1208	L	27	20	t
216	L	27	0	→ P
11	P	9	2	?
500	P	9	5	
18	P	9	0	→ U
241	U	27	25	Y
0	U	27	0	→ L
32	L	27	5	e
20620	L	27	19	s
27	L	27	0	→ P
10	P	9	1	!

Cycle of modes:



Decoded message:
Right? Yes!

Figure 2-4: Sample processing for the "Decode a Message" problem

*You should walk through a concrete example to ensure you have the steps straight

Breaking Down the Problem

Program:

Input = Message encoded as a text stream.

- Series of comma delimited positive ints

Function = Decode message

Constraints

- Read message char by char

Decoding Method

Uppercase Mode

- int % 27 = 1 upcase letter. Ex: 143 % 27 = 8 = H (8th letter in dataset)

Lowercase Mode

- int % 27 = 1 locase letter. Ex: 143 % 27 = 8 = h (8th letter in dataset)

Punctuation Mode

- int % 9 = 1 punctuation

Number	Symbol
1	!
2	?
3	,
4	.
5	(space)
6	;
7	"
8	'

Switching Modes

- Starts upcase
- Each time int % 27 or 9 = 0 mode switches and no char is assigned
- Upcase => locase => punctuation

Developing the Plan

List of Issues (Subproblems)

* Think of the skills you need to craft the solution

- Read character by character until we reach an end-of-line.
- Convert a series of characters representing a number to an integer.
- Convert an integer 1–26 into an uppercase letter.
- Convert an integer 1–26 into a lowercase letter.

] even though these are similar there is no downside to separating them.

- Convert an integer 1–8 into a punctuation symbol based on Table 2-3.
- Track a decoding mode.

Storing Code For Later Reuse

- Code reuse = using old pieces of software to build new software
- Strive to keep all the source code that you write (be mindful of IP laws)
- employs a step-by-step approach and write individual programs to test ideas before making the whole

---

Solving the Problems

Convert a series of chars representing a number to an integer.

Code Reuse: Luhn Checksum program has code for converting single char 0–9 to int

Problem Reduction: Let's begin with 2 digit integers

Code Experiment:

- if you read in a 2 digit number as a string you know:
  - 1st digit is 10s place, 2nd digit is 1s place, So...
    - "35" = '3' + '5' = convert = 3x10 + 5x1 = 35

Scale CE

- Because we don't know the size of number the user will enter we want to be able to handle 1 to x chars.

Step 1 (SP) : Lets use the same 2 digit example but use 1 char var and 1 int var to solve it.

Step 2 (SP) : Scale up to 3 digit #s. This should help us see a general pattern.

- So far we can convert a char to int for 2 digit numbers.
- Our current algorithm restricts us to solving for 2 digits
- As we scale up we don't want a seperate algo for x num of digits
- Also we don't know how many digits there are until we reach the end because of the constraint of reading input char by char

Simplified Problem

- Instead of trying to make the solution for N possibilities of # of digits lets solve for 2 possibilities

Problem: Convert 3 to 4 digit char to integer

- Program ⇒ read a number char by char and convert it to an integer of 3 or 4 digits
- Constraints ⇒ read char by char, 1 int var, 1 char var

Solve by Analogy

- The Luhn checksum solution tracked 2 possibilities (odd or even)
- Relax Constraint: To make this analogous to our solution lets relax the 1 int var constraint

Problem: Convert 3 to 4 digit char to integer ⇒ CE

- Program ⇒ read a number char by char and convert it to an integer of 3 or 4 dig
- Constraints ⇒ read char by char, 2 int var, 1 char var

- Now that we know a conversion of 3 or 4 digit numbers w/ 2 int vars how do we get it down to 1 int var
  - 1230 = 123 x 10 ⇒ if not end of 4th char 4 digit char = (123 x 10) + X

- At this point the pattern is clear:
  - If the next char is another digit multiply the current total by 10 before adding the next number
    else display the number
- Finally you would modify our current solution to convert a series of int delimited by commas.

Convert an Integer 1-26 to an Uppercase Letter

Code Reuse: Luhn Checksum chars to ints this is the reverse. Use that idea.

CTF: Add a char value to input number to get equivalent char value

Convert an Integer 1-26 to lowercase Letter

Having solved uppercase conversion the same solution can be applied to lowercase

Convert an int 1-8 to a punctuation symbol

Brute Force: Because the punctuation chars in ASCII aren't in order
its appropriate to use brute force

Number	Symbol
1	!
2	?
3	,
4	.
5	(space)
6	;
7	"
8	'

Trace a Decoding Mode

Switching Modes

- Starts upcase
- Each time int % 27 or 9 = 0 mode switches and no char is assigned (U & L case, punctuation)
- Upcase ⇒ lo case ⇒ punctuation ↻

Determine What the Program Needs

- Variable to store the current mode
  - Q: How do we want to store this?
  - A: It could be an int but it's more readable to use an enum
  - Rule of Thumb: If a var is only tracking state then an enum is a good idea
- Logic inside the "read and process the next value" loop to switch modes when appropriate.

Assembling the Final Solution

- At this point we've mostly moved beyond problem solving into SWE
- We made a series of blocks (the hard part), Now we just have to assemble them

Approaching Code Integration

- You can approach integration in different ways
- You might just put 2 pieces together and build up from there