Name ________________________________ Track ______ Date ____________________

Python Programming Test II

1) One proposed apportionment algorithm came from William Lowndes, from South Carolina, who developed a method that would favor smaller states. You are to write a function named lowndes that takes two parameters – a number of representatives and a list of populations for the districts in the state or region. The function should return a list of integers with the apportioned number of representatives for each district.
   Here is the algorithm to implement:
   a) Calculate the **divisor** by dividing the total population by the number of representatives.
   b) Calculate the **quota** for each district by dividing the population of each district by the divisor.
   c) Cut off all the decimal parts of the quotas, but don't forget them, and add up the integer values from the quotas.
   d) Assuming that the total from (c) is less than the total number of representatives to allocate, divide the decimal part of each quota by the integer part. Assign the remaining representatives to districts with the highest ratio(s) of decimal part to integer part of each quota until the desired total is reached.

   Use the function declaration below.

```
def lowndes(numreps, populationlist):
```

2) The invention of e-reading technology on computers, cell phones, tablets, and e-readers is revolutionizing the book industry. In years past, there were a handful of publishers capable of bringing a book to publication. Now, almost anyone with a computer can become an independent publishing house. As the number of publishers grow, so too will the number of books. This is beneficial to authors and consumers, but can be problematic for standards organizations that are charged with maintaining order among industries.

A book's ISBN (International Standard Book Number) is a way to uniquely identify a work while simultaneously providing information about that work. ISBNs currently have 13 digits broken down into the following sections:

1. A 3 digit prefix assigned by the GS1 standards organization (currently 978 or 979, but they could be anything).
2. A registration group number that varies from 1 to 5 digits. This number identifies the region or language of the work.
3. A registrant element, which identifies a specific publisher.
4. A publication element, which identifies a work published by the publisher.
5. A single check digit (used to verify the legitimacy of the ISBN).

The check digit is assigned in such a way that taking the sum of all 13 numbers, each multiplied by a factor alternating between 1 and 3, is a factor of 10. For example, for ISBN 978-0-306-40615-7:

```
 9*1 + 7*3 + 8*1 + 0*3 + 3*1 + 0*3 + 6*1 + 4*3 + 0*1 + 6*3 + 1*1 + 5*3 + 7*1
=  9 +  21 +   8 +   0 +   3 +   0 +   6 +  12 +   0 +  18 +   1 +  15 +   7
= 100
```

Since 100 is a multiple of 10, this is a valid ISBN number. There is no standard format for an ISBN. ISBNs can contain dashes or other separating characters (spaces, underscores, etc.) that can make them more readable by humans, or they can just be 13 digits in a row. Your job is to validate a list of ISBN candidate numbers.

Example input:	Example output:
`978-0-306-40615-7`	`VALID`
`9788175257665`	`VALID`
`1234567890123`	`8`

For each ISBN number your program should output one of the following:

- If the ISBN is valid, return the word VALID
- If the ISBN is invalid, return the value that the check bit should have been to make the ISBN valid

Use the method signature that takes a single parameter of a string.

`def isbn_checker(isbn_num):`