JUNE 15TH COMPUTER PROBLEM SET

The trouble with quotes on the internet is that you can never know if they are genuine. – Abraham Lincoln.

We are going to learn to compute strings (lists of letters) into numbers, and back. We do this by taking our string, removing all spaces and special characters and making it lower case. Grouping 12 letters together, and viewing it as a number base 26. If we return a string with more than 12 letters, we will make a function should return a list of numbers (after breaking the string into a list of 12 character strings).

There are some weird aspects to this for us. `"aaaaaaaaaaab"` corresponds to the number 1. If we are going to try to represent the word `"cat"`, which has only 3 letters, we are going to need to stick some other letters onto the end of it to make it 12 letters. For instance, maybe we should write `"catzzzzzzzzz"`.

Regardless, here's how I started my function that turns letters into numbers.

```
def shortStringToNum(myStr, maxLen=12):
    #converts a string of length at most
    #maxLen to a number.  All entries are
    #assumed to be lower case letters.
    if (len(myStr) > maxLen):
        print "String too long"
        return -1
    curNum = 0;  #our running total
    for i in range(0,len(myStr)):
        c = myStr[len(myStr)-i-1]
        ...
    ...
```

Notice I do `"len(myStr)-i-1"` because this will give the last character of my string, followed by the second to last, third to last, etc. (ie, the 1s digit, the 26s digit, the 26^2s digit, and so on). In the ... I add an appropriate number to curNum at each step of the loop.

1. Write your own function. Then verify that it is working by checking that it turns `"abcdefghijkl"` into `152686330658691` and that it turns `"qwertyuiopas"` into `61856616230903902`.

Next we write a function that turns a number back into a string. Let's do a smaller case. How do we turn $729 = 1 * 26^2 + 2 * 26 + 1$ back into `bcb`. First we can take $729\%26$ which will give 1 (the letter `b`). After that we compute $(792 - 1)/26 = 28$. Now we repeat. We compute $28\%26 = 2$ which gives us the letter `c` and then keep working with $(28 - 2)/26 = 1$, which finally gets converted back into `b` again.

My function started like this.

```
def numToShortString(myNum, maxLen=12):
    #this converts a number into a string of length 12.
    curStr = ""
    for i in range(0,maxLen):
        c = myNum%26
```

```
        ...
    ...
```

2. Write your own function. Test it out by verifying that `numToShortString(729)` returns `'aaaaaaaaabcb'`. Find out what the number `66645316357222622` corresponds to in a string. Also check to make sure if you convert a string with 12 characters into a number, and back again, you get the same string you started with.

Now we need to write a function that converts a string into a list of numbers (corresponding to substrings of 12 characters). We do this in three steps.

(a) Add `z`s to the end of our string until the length of it is divisible by 12.
(b) Take the first 12 characters in the string and turn it into a number.
(c) Add the number to the list and replace the string with the remaining characters.

My function started like this:

```
def stringToNumList(myStr, maxLen=12):
    #turns a string into a list of numbers
    extraChars = len(myStr)%maxLen
    if (extraChars > 0):
        myStr = myStr + 'z'*(maxLen-extraChars)
    #now the string should have the right number
    #of letters
    curList = [] #an empty list which we will return full
    ...
```

3. Make your own function. Make sure to test it on a concatenation of strings whose corresponding numbers you know.

Finally, we need a function that turns a list of numbers back into a string. I started my function like this.

```
def numListToString(myList, maxLen=12):
    #turns a list of numbers back into a string.
    ...
```

4. Make your own function. Then test it. What string does `[70771979662198598, 71456171948642831]` correspond to?