Reminders

• Assignment 6 is due Wednesday, October 30th at 10:00 AM.
• Midterm is next Monday, November 4th, at 7:00pm – 8:50pm
• Look up your seat on Odyssey.
• Additional practice problems are available for each module. See “Additional Materials” on the course web page.
Generative Recursion: A Summary!

- It's recursion…
- Solving larger problem by solving subproblem(s) inspired by the problem itself.
- There's no "structural" format:
  - Recursing at 'different' places
  - Recursing 'multiple' times
  - Not always counting up/down by "one"

- Classic examples (shown in lectures):
  - Palindromes
  - Solving GCD's with Euclid's Algorithm
Example: Euclidean Algorithm on GCD

Let $m$ and $n$ be integers such that $m \geq n$ and $m = qn + r$, where $0 \leq r < n$. Then the following is true:

$$\text{gcd}(m, n) = \text{gcd}(n, r)$$

Note:
- $r = m \mod n$
- Check out the proof for Euclidean Algorithm on [Wikipedia](https://en.wikipedia.org/wiki/Euclidean_algorithm)

<table>
<thead>
<tr>
<th>Code</th>
<th>Analysis</th>
</tr>
</thead>
</table>
| ```python
def gcd(m, n):
    if m == 0:
        return n
    elif n == 0:
        return m
    else:
        return gcd(n, m%n)
``` | Base Case #1 |
| | Base Case #2 |
| | Recursive call: |
| | • $n \leq m$ |
| | • $0 \leq m \mod n < n$ |
QUESTION 1: SKIPPING VALUES

Given a list $L$ of positive integers, the skip-value of a list is the number of steps to reach the end of the list, using the values in the list

- If $L$ is empty, the skip value is 0
- If $L$ is nonempty:
  - Add 1 to the remaining skip value
  - Move ahead $L[0]$ places in the list, and repeat the process with the remainder of the list from that place

Write a function `skip_value` to calculate the skip value of the list $L$.

For example,

```python
skip_value([]) => 0
skip_value([1,1,1]) => 3
skip_value([2,100,1]) => 2
```
skip_value([1,1,1])
⇒ 1+skip_value([1,1])
⇒ 1+(1+skip_value([1]))
⇒ 1+(1+(1+skip_value([])))
⇒ 1+(1+(1+0))
⇒ 3

skip_value([2,100,3,1,1,1])
⇒ 1+skip_value([3,1,1,1])
⇒ 1+(1+skip_value([1]))
⇒ 1+(1+(1+skip_value([])))
⇒ 1+(1+(1+0))
⇒ 3
QUESTION 2: INTRO TO TABLES

Consider a new 'data type': Table.

A Table is a \((\text{listof (listof Int)})\), which is nonempty, and in which each list corresponds to a row of a Table. It is assumed that each row is nonempty and each row has the same number of entries as every other row.

For example, the following is a table:
\[
t_3 = \begin{bmatrix}
[1, 2],& [3, 4],
[5, 6],& [7, 8]
\end{bmatrix}
\]

Sum of each column is calculated as following:

Sum of column 1 of \(t_3\) = \(1 + 3 + 5 + 7 = 16\)

Sum of column 2 of \(t_3\) = \(2 + 4 + 6 + 8 = 20\)
QUESTION 2: SUM OF COLUMNS

For example, the following are tables:

\[ t0 = \begin{bmatrix} 1 \end{bmatrix} \]
\[ t1 = \begin{bmatrix} 1, 2, 3 \end{bmatrix} \]
\[ t2 = \begin{bmatrix} 1 \end{bmatrix}, \begin{bmatrix} 2 \end{bmatrix}, \begin{bmatrix} 3 \end{bmatrix} \]
\[ t3 = \begin{bmatrix} 1, 2 \end{bmatrix}, \begin{bmatrix} 3, 4 \end{bmatrix}, \begin{bmatrix} 5, 6 \end{bmatrix}, \begin{bmatrix} 7, 8 \end{bmatrix} \]

Write a function `sum_of_columns` that consumes a Table \( t \), and returns a list containing the columns sums for \( t \).

For example,

\[
\begin{align*}
\text{sum}_\text{of}_\text{columns}(t0) & \Rightarrow [1] \\
\text{sum}_\text{of}_\text{columns}(t1) & \Rightarrow [1, 2, 3] \\
\text{sum}_\text{of}_\text{columns}(t2) & \Rightarrow [6] \\
\text{sum}_\text{of}_\text{columns}(t3) & \Rightarrow [16, 20]
\end{align*}
\]
Write a function `smaller` that consumes a nonempty string `s`, containing only numeric characters, and generates a new string by repeatedly removing the larger of the first and last characters in `s`. If the first and the last number are the same, remove the last one.

For example, starting from "5284", compare "5" and "4", and recurse on "284", which will compare "2" and "4", and recurse on "28". Comparing "2" and "8", leads to recursing on "2", which is the answer (since it is a string of length 1).

NOTE: Do not use `min`.

For example,
```
smaller("4325") => "2"
smaller("1") => "1"
smaller("2325") => "2"
smaller("8668") => "6"
```