Problem 1 (Cycle notation) 7 Points
Write a Python function `cycleToTable` that accepts as input a list `[a_1,...,a_s]` with each
\( a_i \) being a non-empty list of non-negative integers. Each \( a_i \) represents a cycle of a
permutation in cycle notation.

Example: The permutation \( (0,3,4)(2)(1,5) \) could be represented as \([0,3,4],[2],[1,5]\).

Your function must raise a `TypeError` if one of the following conditions is violated:

- Let \( r \) be the greatest number occurring in any one of the cycles \( a_1,...,a_s \). Then all
  numbers between 0 and \( r \) (including 0) occur in some \( a_i \).
- Within each \( a_i \) a number occurs at most once (but the same number may occur in
  several cycles).

If these conditions are fulfilled, your function returns a list \( p \) of length \( r+1 \) such that for
all \( i=0,...,r: p[i]=a_1(a_2(\ldots(a_s(i))\ldots).\)

Problem 2 (Permutation class) 8 Points
Write a Python class `MyPermutation`, which allows a user to generate a permutation
by specifying it in cycle notation.

- Provide an implementation of the multiplication operator `*` such that \( p_i*\sigma \)
  returns the permutation corresponding to \( p_i \circ \sigma \) (functional composition).
- Provide an implementation of `__str__`, returning a string representation of a
  permutation in cycle notation.
- Provide an implementation of `==`, returning `True` if and only if two permutations
  are equal—e.g., \( (1,2,3) \) and \( (2,3,1) \) should be considered as equal.

Good luck—and do not hesitate to ask questions!!