2. **Z-algorithm** Biological sequences are often circular as in many bacterial genomes. Scientists simply cut the genome sequence at either an arbitrary point or at the origin of replication. Genome assemblers (a program that can stitch short sequences into a large genome sequence) also do this by breaking the circular sequence at a random location and report it as a linear sequence. If an identical circular sequence is broken at a different positions, they can often result in different linear sequences. Given two linear sequences X and Y , explain how we can use Z-algorithm to check if X and Y come from an identical circular sequence.
    a. To verify if sequences X and Y come from an identical circular sequence, you could concatenate sequence Y to itself, represented as $Y \cdot Y$. You can then use the Z-Algorithm to compare X and $Y \cdot Y$, and if the length of X is found in the Z-Table sequences X and Y are from an identical circular sequence.

3. Read the uploaded lecture notes on Exact Pattern Matching and study the **KMP algorithm**. Given a **pattern P**, we defined **lps[i]** as the length of the longest nontrivial suffix of P[1..i] that matches a prefix of P. Here is a pattern **P = ATCATCT** and its lps array:

i	1	2	3	4	5	6	7
P[i]	A	T	C	A	T	C	T
lps[i]	0	0	0	1	2	3	0

Dr. Wiz claims that s/he has a better idea for a modified KMP pattern matching and newly defines a **lps' array** for a given pattern, where **lps'[i]** is the length of the longest nontrivial suffix of P[1..i] that matches a prefix of P such that $P[lps'[i] + 1] \neq P[i + 1]$. That is, the character following the matched longest suffix/prefix is not the same. If those two characters are equal, **lps'[i] = 0**. For example, for the above example of **P = ATCATCT**:

i	1	2	3	4	5	6	7
P[i]	A	T	C	A	T	C	T
lps'[i]	0	0	0	0	0	3	0

Dr. Wiz claims that KMP search routing can directly use **lps' values** INSTEAD OF lps values to make the search process a bit more efficient. Is this a valid claim? Test it out by running KMP algorithm by hand on P = ATCATCT and S = TCATCATGATGATCATCT, and explain whether this is a valid claim or not

   a. The claim, that using the **lps' values** is slightly faster, is valid. During the string comparison, there was a slight optimization around the first 'G' character. At this index, using the **lps values** resulted in 2 comparisons, compared to the 1 comparison using **lps' values**.

4. It may have been obvious that there are lots of similarities between Z array and lps array. Pre-computed Z values can be used to compute lps instead of using the approach/method we discussed in lectures. Write out the algorithm for computing the modified lps' values using the Z values. The inputs to algorithms are a string S over the alphabet (ex: Σ = {A,C,G,T}) and Z array values for S. The output is the modified **lps' array** (Providing pseudocode is sufficient)

```
Function getLpsPrime(string S, int[] Z)

        int[] lpsPrim = Z

        FOR EACH int val IN Z

                IF Z[val] > 1

                        lpsPrim[val] = 0

                        lpsPrim[ val+Z[val] ] =  Z[val]


        RETURN lpsPrim
```