WEEK 2 PROBLEMS

Problem 1 :

"We've detected some temporal anomalies," one of Santa's Elves at the Temporal Anomaly Research and Detection Instrument Station tells you. She sounded pretty worried when she called you down here. "At 500-year intervals into the past, someone has been changing Santa's history!"
"The good news is that the changes won't propagate to our time stream for another 25 days, and we have a device" - she attaches something to your wrist - "that will let you fix the changes with no such propagation delay. It's configured to send you 500 years further into the past every few days; that was the best we could do on such short notice."
"The bad news is that we are detecting roughly fifty anomalies throughout time; the device will indicate fixed anomalies with stars. The other bad news is that we only have one device and you're the best person for the job! Good lu--" She taps a button on the device and you suddenly feel like you're falling.
After feeling like you've been falling for a few minutes, you look at the device's tiny screen. "Error: Device must be calibrated before first use. Frequency drift detected. Cannot maintain destination lock." Below the message, the device shows a sequence of changes in frequency (your puzzle input). A value like +6 means the current frequency increases by 6; a value like -3 means the current frequency decreases by 3.
For example, if the device displays frequency changes of +1, -2, +3, +1, then starting from a frequency of zero, the following changes would occur:
Current frequency 0, change of +1; resulting frequency 1.
Current frequency 1, change of -2; resulting frequency -1.
Current frequency -1, change of +3; resulting frequency 2.
Current frequency 2, change of +1; resulting frequency 3.
In this example, the resulting frequency is 3.
Here are other example situations:
+1, +1, +1 results in 3
+1, +1, -2 results in 0
-1, -2, -3 results in -6
Starting with a frequency of zero, what is the resulting frequency after all of the changes in frequency have been applied?

Problem 2 :

An ugly number is a positive integer whose prime factors are limited to 2, 3, and 5. Given an integer n, return true if n is an ugly number.

Example 1:
Input: n = 6
Output: true
Explanation: 6 = 2 × 3

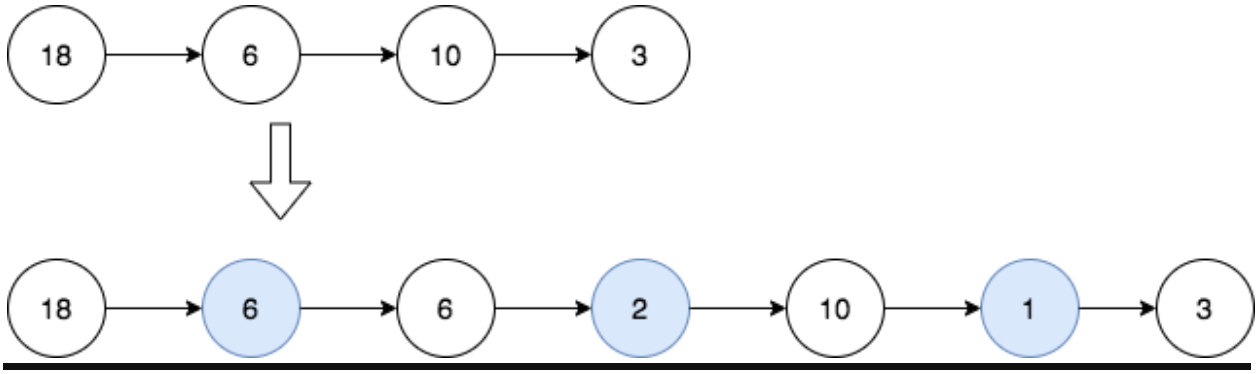
Example 2:
Input: n = 1
Output: true
Explanation: 1 has no prime factors, therefore all of its prime factors are limited to 2, 3, and 5.

Example 3:
Input: n = 14
Output: false
Explanation: 14 is not ugly since it includes the prime factor 7.

Problem 3 :

Given the head of a linked list head, in which each node contains an integer value. Between every pair of adjacent nodes, insert a new node with a value equal to the greatest common divisor of them.
Return the linked list after insertion.
The greatest common divisor of two numbers is the largest positive integer that evenly divides both numbers.
Input: head = [18,6,10,3]

Output: [18,6,6,2,10,1,3]



Input: head = [7]
Output: [7]

Constraints:
The number of nodes in the list is in the range [1, 5000].
1 <= Node.val <= 1000

Problem 4 :

You are given an integer array nums with no duplicates. A maximum binary tree can be built recursively from nums using the following algorithm:
Create a root node whose value is the maximum value in nums.
Recursively build the left subtree on the subarray prefix to the left of the maximum value.
Recursively build the right subtree on the subarray suffix to the right of the maximum value.
Return the maximum binary tree built from nums.

Input: nums = [3,2,1,6,0,5]
Output: [6,3,5,null,2,0,null,null,1]
Explanation: The recursive calls are as follow:
- The largest value in [3,2,1,6,0,5] is 6. Left prefix is [3,2,1] and right suffix is [0,5].
  - The largest value in [3,2,1] is 3. Left prefix is [] and right suffix is [2,1].
    - Empty array, so no child.

- The largest value in [2,1] is 2. Left prefix is [] and right suffix is [1].
  - Empty array, so no child.
  - Only one element, so child is a node with value 1.
- The largest value in [0,5] is 5. Left prefix is [0] and right suffix is [].
  - Only one element, so child is a node with value 0.
  - Empty array, so no child.

Input: nums = [3,2,1]
Output: [3,null,2,null,1]

Constraints:
1 <= nums.length <= 1000
0 <= nums[i] <= 1000
All integers in nums are unique.
