1. You have a list arr of all integers in the range [1, n] sorted in a strictly increasing order. Apply the following algorithm on arr:
Starting from left to right, remove the first number and every other number afterward until you reach the end of the list.
Repeat the previous step again, but this time from right to left, remove the rightmost number and every other number from the remaining numbers.
Keep repeating the steps again, alternating left to right and right to left, until a single number remains.
Given the integer n, return the last number that remains in arr.

Example 1:
Input: n = 9
Output: 6
Explanation:
arr = [1, 2, 3, 4, 5, 6, 7, 8, 9]
arr = [2, 4, 6, 8]
arr = [2, 6]
arr = [6]

Example 2:
Input: n = 1
Output: 1
Constraints:
1 <= n <= 10^9
==
2. Given a string s, find the length of the longest substring without repeating characters.

Example 1:
Input: s = "abcabcbb"
Output: 3
Explanation: The answer is "abc", with the length of 3.

Example 2:
Input: s = "bbbbb"
Output: 1
Explanation: The answer is "b", with the length of 1.

Example 3:
Input: s = "pwwkew"
Output: 3
Explanation: The answer is "wke", with the length of 3.
Notice that the answer must be a substring, "pwke" is a subsequence and not a substring.

Constraints:
0 <= s.length <= 5 * 10^4
s consists of English letters, digits, symbols and spaces.
===
3. Given r red, b blue, and g green balls, find the total number of arrangements in a row such that no two balls of the same color end up together.

For example,
Input: r = 1, b = 2, g = 1
Output: Total number of arrangements are 6
The arrangements are [bgbr, bgrb, brbg, brgb, gbrb, rbgb]

Input: r = 2, b = 3, g = 1
Output: Total number of arrangements are 10
The arrangements are [bgbrbr, bgrbrb, brbgbr, brbgrb, brbrbg, brbrgb, brgbrb,gbrbrb, rbgbrb, rbrbgb]
==
4. You are working in a ball factory where you have n balls numbered from lowLimit up to highLimit inclusive (i.e., n == highLimit - lowLimit + 1), and an infinite number of boxes numbered from 1 to infinity.
Your job at this factory is to put each ball in the box with a number equal to the sum of digits of the ball's number. For example, the ball number 321 will be put in the box number 3 + 2 + 1 = 6 and the ball number 10 will be put in the box number 1 + 0 = 1.

Given two integers lowLimit and highLimit, return the number of balls in the box with the most balls.
Example 1:
Input: lowLimit = 1, highLimit = 10
Output: 2
Explanation:
Box Number: 1 2 3 4 5 6 7 8 9 10 11 ...
Ball Count: 2 1 1 1 1 1 1 1 1 0 0 ...
Box 1 has the most number of balls with 2 balls.

Example 2:
Input: lowLimit = 5, highLimit = 15
Output: 2
Explanation:
Box Number: 1 2 3 4 5 6 7 8 9 10 11 ...
Ball Count: 1 1 1 1 2 2 1 1 1 0 0 ...
Boxes 5 and 6 have the most number of balls with 2 balls in each.

Example 3:
Input: lowLimit = 19, highLimit = 28
Output: 2
Explanation:
Box Number: 1 2 3 4 5 6 7 8 9 10 11 12 ...
Ball Count: 0 1 1 1 1 1 1 1 1 2 0 0 ...
Box 10 has the most number of balls with 2 balls.
==
5. Given a string s, return the maximum number of ocurrences of any substring under the following rules:
The number of unique characters in the substring must be less than or equal to maxLetters.
The substring size must be between minSize and maxSize inclusive.
Example 1:
Input: s = "aababcaab", maxLetters = 2, minSize = 3, maxSize = 4
Output: 2
Explanation: Substring "aab" has 2 ocurrences in the original string. It satisfies the conditions, 2 unique letters and size 3 (between minSize and maxSize).

Example 2:
Input: s = "aaaa", maxLetters = 1, minSize = 3, maxSize = 3
Output: 2
Explanation: Substring "aaa" occur 2 times in the string. It can overlap.

Constraints:
1 <= s.length <= 10^5
1 <= maxLetters <= 26
1 <= minSize <= maxSize <= min(26, s.length)
s consists of only lowercase English letters.
==
6. You are given a floating-point number hour, representing the amount of time you have to reach the office. To commute to the office, you must take n trains in sequential order. You are also given an integer array dist of length n, where dist[i] describes the distance (in kilometers) of the ith train ride.

Each train can only depart at an integer hour, so you may need to wait in between each train ride.

For example, if the 1st train ride takes 1.5 hours, you must wait for an additional 0.5 hours before you can depart on the 2nd train ride at the 2 hour mark.
Return the minimum positive integer speed (in kilometers per hour) that all the trains must travel at for you to reach the office on time, or -1 if it is impossible to be on time.

Tests are generated such that the answer will not exceed 107 and hour will have at most two digits after the decimal point.

Example 1:
Input: dist = [1,3,2], hour = 6
Output: 1
Explanation: At speed 1:

- The first train ride takes 1/1 = 1 hour.
- Since we are already at an integer hour, we depart immediately at the 1 hour mark. The second train takes 3/1 = 3 hours.
- Since we are already at an integer hour, we depart immediately at the 4 hour mark. The third train takes 2/1 = 2 hours.
- You will arrive at exactly the 6 hour mark.

Example 2:
Input: dist = [1,3,2], hour = 2.7
Output: 3
Explanation: At speed 3:
- The first train ride takes 1/3 = 0.33333 hours.
- Since we are not at an integer hour, we wait until the 1 hour mark to depart. The second train ride takes 3/3 = 1 hour.
- Since we are already at an integer hour, we depart immediately at the 2 hour mark. The third train takes 2/3 = 0.66667 hours.
- You will arrive at the 2.66667 hour mark.

Example 3:
Input: dist = [1,3,2], hour = 1.9
Output: -1
Explanation: It is impossible because the earliest the third train can depart is at the 2 hour mark.

Constraints:
n == dist.length
1 <= n <= 10^5
1 <= dist[i] <= 10^5
1 <= hour <= 10^9
There will be at most two digits after the decimal point in hour.
==