**SMarket**
*2021 May 01*

Q1. Shifting string
I failed to get 100% score, failed with corner cases (what cases?)

The following operations on a string are defined:

- *Left Shift*: A single circular rotation of the string in which the first character becomes the last character and all other characters are shifted one index to the left. For example, *abcde* becomes *bcdea* after one left shift and *cdeab* after two left shifts.
- *Right Shift*: A single circular rotation of the string in which the last character becomes the first character and all other characters are shifted one index to the right. For example, *abcde* becomes *eabcd* after one right shift and *deabc* after two right shifts.

**Example**
*s = 'abcdefg'*
*leftShifts = 2*
*rightShifts = 4*

The string *abcdefg* shifted left by 2 positions is *cdefgab*. The string *cdefgab* shifted right by 4 positions is *fgabcde*, the string to return.

**Function Description**
Complete the function *getShiftedString* in the editor below.

getShiftedString has the following parameter(s):
*string s:* the string to shift
*int leftShifts:* number of shifts left
*int rightShifts:* number of shifts right

Returns:
*string:* a string shifted left or right

**Constraints**

- $1 \le$ *length of* $s \le 10^5$
- $0 \le leftShifts, rightShifts \le 10^9$

*Solution*
in linear time

Q2. Less than target subsequence product

Contiguous subarrays are a group of an uninterrupted range of elements from an array as they occur. No elements in the range can be skipped or reordered. Given an array of integers, *numbers*, and an integer *k*, determine the total number of subarrays of *numbers* having a product that is less than or equal to *k*.

**Example**
*numbers = [2, 3, 4]*

The subarrays are *[[2], [3], [4], [2, 3], [3, 4], [2, 3, 4]]*. The product of a subarray is the product of all of its elements so the products for the list of subarrays are *[2, 3, 4, 6, 12, 24]*. If *k = 6*, there are *4* subarrays that satisfy the condition, *[2], [3], [4], [2, 3]*.

**Function Description**
Complete the function *count* in the editor below.

count has the following parameter(s):
*int numbers[n]:* an array of integers
*k:* an integer

Returns:
*long int:* the number of subarrays whose product is less than or equal to *k*

**Constraints**

- $1 \le n \le 5 \times 10^5$
- $1 \le numbers[i] \le 100$
- $1 \le k \le 10^6$

I am supposed to do it, however I failed ... disappointing. Please read comments for critical parts that I missed in the test.

Q3. Point puzzle (also known as maximum point in Hackerrank contest)

This is a variant of non-contigous subsequence sum. This should be done in linear time.

- subproblem optimal including last element
- subproblem optimal excluding last element

An escape game requires that players solve various puzzles to obtain objects that will aid them in escaping. In one such puzzle, players are given an array of integers and a set of rules to follow in order. Here are the instructions given to the players: "Given an array of integer values, maximize the points earned using the following method:

1. Choose a value *v*. Delete all elements of that value and add their sum to the points.
2. Delete all elements equal to *v + 1* or *v - 1* for no points.
3. Repeat steps 1 and 2 until there are no more elements in the array.

Calculate the maximum number of points that can be achieved by following these rules for an array of values.

**Example**

*elements = [5, 6, 6, 4, 11]*

Delete *11* for *11 points.* Since there are no elements equal to *11-1 = 10* or *11 + 1 = 12*, proceed with the remaining elements: *[5, 6, 6, 4].* Delete the two *6's* for *12* more points. Delete any elements equal to *6 - 1 = 5* or *6+1 = 7.* Then proceed with the remaining elements: *[4].* Finally delete the *4* and add *4* points for total points *11 + 12 + 4 = 27.*

**Function Description**

Complete the *maxPoints* function in the editor below. .

maxPoints has the following parameter(s)

*int elements[n]:* an array of integers

**Returns**

*long int:* the maximum number of points that can be earned

Solution :

The numbers are unsorted. Firstly, put numbers into histogram, so that :

- no need to count the occurrence of a number every time
- no need to scan for value $v \pm 1$ in $O(N)$, do it with hashmap in $O(1)$ instead