**GCD Array**

You are given an array, **arr**of length **N**, and also a single integer **K**. Your task is to split the array arr into **K**non-overlapping, non-empty subarrays. For each of the subarrays, you calculate the sum of the elements in it. Let us denote these sums as **S1,S2, S3, ..., SK**. Where **Si**denotes the sum of the elements in the ithsubarray from left.

Let **G = GCD( S1,S2,S3, ...,SK)**.

Find the **maximum**value of G that can be obtained.   
The array may contain duplicate elements.

**Example 1:**

**Input**:

N = 5

K = 4

arr[] = {6, 7, 5, 27, 3}

**Output:** 3

**Explanation**:

Since K = 4, you have to split the array into 4 subarrays.

For optimal splitting, split the array into

4 subarrays as follows: [[6], [7, 5], [27], [3]]

Therefore, S1 = 6, S2 = 7 + 5 = 12, S3 = 27, S4 = 3

Hence, G = GCD(S1, S2, S3, S4) = GCD(6, 12, 27, 3) = 3

It can be shown that 3 is the maximum value of G that can be obtained.

Thus, the answer is 3.

**Example 2:**

**Input**:

N = 3

K = 2

arr[] = {1, 4, 5}

**Output:** 5

**Explanation**:

Since K = 2, you have to split the array into 2 subarrays.

For optimal splitting, split the array into

2 subarrays as follows: [[1, 4], [5]]

Therefore, S1 = 1 + 4 = 5, S2 = 5

Hence, G = GCD(S1, S2) = GCD(5,5) = 5

It can be shown that 5 is the maximum value of G that can be obtained.

Thus, the answer is 5.

**Your Task:**  
You don't need to read input or print anything. Your task is to complete the function **solve()** which takes the array arr[] and its size N and an integer K as input parameters and returns the required maximum GCD value.

**Expected Time Complexity:**O(N \* x)  
**Expected Auxiliary Space:**O(x), x is the number of factors of the sum of all elements.  
  
**Constraints:**

1 <= N <= 104  
1 <= K <= N  
1 <= arr[i] <= 105