// CPP program to find subsequence of size

// k with maximum possible GCD.

#include <bits/stdc++.h>

using namespace std;

// function to find GCD of sub sequence of

// size k with max GCD in the array

int findMaxGCD(int arr[], int n, int k)

{

    // Computing highest element

    int high = \*max\_element(arr, arr+n);

    // Array to store the count of divisors

    // i.e. Potential GCDs

    int divisors[high + 1] = { 0 };

    // Iterating over every element

    for (int i = 0; i < n; i++) {

        // Calculating all the divisors

        for (int j = 1; j <= sqrt(arr[i]); j++) {

            // Divisor found

            if (arr[i] % j == 0) {

                // Incrementing count for divisor

                divisors[j]++;

                // Element/divisor is also a divisor

                // Checking if both divisors are

                // not same

                if (j != arr[i] / j)

                    divisors[arr[i] / j]++;

            }

        }

    }

    // Checking the highest potential GCD

    for (int i = high; i >= 1; i--)

        // If this divisor can divide at least k

        // numbers, it is a GCD of at least one

        // sub sequence of size k

        if (divisors[i] >= k)

            return i;

}

// Driver code

int main()

{

    // Array in which sub sequence with size

    // k with max GCD is to be found

    int arr[] = { 1, 2, 4, 8, 8, 12 };

    int k = 3;

    int n = sizeof(arr) / sizeof(arr[0]);

    cout << findMaxGCD(arr, n, k);

    return 0;

}