Non-Prime Factors [Kattis - nonprimefactors]

In many programming competitions, we are asked to find (or count the number of) Prime Factors of an integer i. This is boring. This time, let's count the number of Non-Prime Factors of an integer i, denoted as `NPF(i)`.

For example, integer 100 has the following nine factors: $\{1, \underline{2}, 4, \underline{5}, 10, 20, 25, 50, 100\}$. The two which are underlined are prime factors of 100 and the rest are non-prime factors. Therefore, `NPF(100)` = 7.

Input

The first line contains an integer Q ($1 \leq Q \leq 3 \cdot 10^6$) denoting the number of queries. Each of the next Q lines contains one integer i ($2 \leq i \leq 2 \cdot 10^6$).

Output

For each query i, print the value of `NPF(i)`.

Warning

The I/O files are large. Please use fast I/O methods.

Sample 1

Input	Output
4 100 13 12 2018	7 1 4 2

Modulo Ruins the Legend Gym - 104090A

Grammy has a sequence of integers $a_1, a_2, \ldots, a_n$. She thinks that the elements in the sequence are too large, so she decided to add an arithmetic progression to the sequence. Formally, she can choose two non-negative integers s, d, and add $s + kd$ to a_k for each $k \in [1, n]$.

Since we want to ruin the legend, please tell her the minimum sum of elements modulo m after the operation. Note that you should minimize the sum **after** taking the modulo.

Input

The first line contains two integers n, m ($1 \leq n \leq 10^5, 1 \leq m \leq 10^9$).

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($0 \leq a_i < m$), denoting the initial sequence.

Output

Output exactly two lines.

The first line contains one integer, denoting the minimum sum of elements modulo m.

The second line contains two integers s, d ($0 \leq s, d < m$), denoting the integers chosen by Grammy. **If there are multiple solutions, output any.**

Sample 1

Input	Output
6 24 1 1 4 5 1 4	1 0 5

Sample 2

Input	Output
7 29 1 9 1 9 8 1 0	0 0 0

What is the value this simple C++ function will return?

```
long long H(int n){
      long long res = 0;
     for( int i = 1; i <= n; i=i+1 ){
            res = (res + n/i);
      }
     return res;
}
```

Input

The first line of input is an integer T ($T \leq 1000$) that indicates the number of test cases. Each of the next T line will contain a single signed 32 bit integer n.

Output

For each test case, output will be a single line containing $H(n)$.

Sample Input

```
2
5
10
```

Sample Output

```
10
27
```