The Trial of Order and Chaos

Long ago, in a village that nobody really visits, the people invented a strange test. The chief of the village was tired of people getting away with sloppy answers, so he invented a ritual with **many steps** and an equally **boring grading system**. You are now the unlucky one who must complete this ritual.

The chief gives you:

- An integer `n`, the number of elements in the array.
- An integer `m`, which is the modulus used for all calculations.
- An array `A` of length `n`.

You must follow the ritual exactly as told, one step at a time. After every single step, you must output the full line of numbers you obtained, with spaces between them, and **nothing else**. If you skip even one detail, you fail.

Step 1 – Prefix Ritual

Take the array `A` and create a new one as follows:

- If the index is even, compute the sum of all elements from the start up to that index and take it modulo `m`.
- If the index is odd, compute `(A[i] - A[i-1] + A[i-2] - A[i-3] ... - A[0]) % m`. If this value is negative, keep adding `m` until it is non-negative.

Replace `A` with this new array. Output the array.

Step 2 – Subset Ritual

Take every possible group of three different elements from `A`, always in increasing index order. For each group `(x, y, z)`:

- Compute `(x + y + z) % m`.
- If this result is even, put it into list `B`.
- If it is odd, instead put `(result * result) % m` into `B`.

Output `B`.

Step 3 – Sorting Ritual

Take list `B` and sort it in non-decreasing order. Then split `B` into blocks by cutting after every three elements. For every second block (the block at indices `[3..5]`, then `[9..11]`, and so on), reverse the order of the block.

Output the modified `B`.

Step 4 – Unique Ritual

Now, from `B`, keep only one copy of each distinct number, and keep them sorted in increasing order.

After this, for every number `t` between `0` and `m-1`:

- If `t` is missing from your collection, but `t+1` is present, insert `t` as well.

Output the collection.

Step 5 – Final Verdict

Now for each number `x` in the collection:

- If `x % 3 == 0`, transform it into `(x / 3) % m`.
- If `x % 3 == 1`, transform it into `(x * 2 + 1) % m`.
- If `x % 3 == 2`, transform it into `(x * x - 1 + m) % m`.

This creates a list `C`.

Sort this list so that all even numbers appear before all odd numbers, but preserve the relative order inside the even group and inside the odd group (this is called stable partition).

Output `C`.

Constraints

- $1 \leq n \leq 300$
- $1 \leq m \leq 2 \times 10^5$
- $1 \leq A[i] \leq 10^{16}$ for all $0 \leq i < n$

Input Format

```
n m
A[0] A[1] ... A[n-1]
```

Output Format

You must print the results of each step, each on its own line, as space-separated numbers. **Do not print any labels, extra words, or step numbers. Only the numbers.**

Example

Input

```
5 12
3 4 7 9 10
```

Output

```
3 1 2 3 9
6 1 1 8 2 9 6 0 1 2
0 1 1 2 2 1 6 6 8 9
0 1 2 5 6 7 8 9
0 0 2 3 3 3 3 3
```

Grading System

The chief of the village also created a grading system that makes sure you cannot trick it by being partially correct in the middle. Here is how it works, step by step:

1. The official solution of the problem produces one big sequence of numbers, line by line, in the same format you are asked to produce.
2. Your program's output is compared to the official correct output, number by number, in order.
3. The comparison goes from the very first number to the last:
    - If your number is correct, you earn **1 point** and the grading continues to the next number.
    - If your number is wrong, the grading immediately stops, and no further numbers are checked.
4. After the comparison ends (either because you got everything right or you made your first mistake), the total number of points you collected is multiplied by **100**, then is divided by the total number of numbers in the correct solution.