Problem F. Noodle

Input file:	*standard input*
Output file:	*standard output*
Time limit:	7 seconds
Memory limit:	1024 mebibytes

Putata is a boy who loves eating noodles. Now he's waiting for the great chef Budada to cook the most delicious noodle ever for him.

The noodle which Budada is cooking for him can be described as an array a of length n, where n is even. The amount of sauce initially at position i is a_i.

In one operation, Budada will do the following process.

1. Budada will fold the noodle, the length of the noodle will become $\frac{n}{2}$, the amount of sauce at position i will become the sum of the amounts of sauce at positions i and $n-i+1$. Formally, the amount of sauce at position i of the new noodle b_i satisfies $b_i = a_i + a_{n-i+1}$.
2. Then Budada will stretch the noodle to the original length, and the amount of sauce will be evenly divided. Formally, the amount of sauce at position i of the new noodle a'_i satisfies $a'_i = \frac{1}{2} \cdot b_{\lceil \frac{i}{2} \rceil}$.

Putata has a favorite position on the noodle, which is a certain position x. Now you are asked to answer q queries. In the i-th query, you should output the amount of sauce at position x after k operations. The x is the same for all queries, but k is given separately for each query.

It can be shown that the answer can be expressed as an irreducible fraction $\frac{x}{y}$, where x and y are integers and $y \not\equiv 0 \pmod{998\,244\,353}$. Output the integer equal to $x \cdot y^{-1} \pmod{998\,244\,353}$. In other words, output such an integer a that $0 \le a < 998\,244\,353$ and $a \cdot y \equiv x \pmod{998\,244\,353}$.

Since the input is quite large, you will have to use a generator to generate the queries, and you only have to output $\oplus_{i=1}^{q}(ans_i \cdot i)$. Please notice that this number is **not** taken modulo $998\,244\,353$. Here, $\oplus$ means bitwise exclusive-or operation.

Input

The first line contains three integers *test*, T, and *seed*, which are an **unrelated variable**, the number of test cases, and the seed for generating test data. Please note that *test* **will not** be used to solve the problem, you can just ignore it. The generator code is given further below.

For each test case, the input will contain two lines.

The first line contains four integers n, q, x, and $k_{\max}$ ($1 \le n \le 2 \cdot 10^6$, $1 \le q \le 5 \cdot 10^7$, $1 \le x \le n$, $1 \le k_{\max} \le 10^{18}$).

The second line contains n integers, the i-th integer is a_i ($0 \le a_i < 998\,244\,353$).

It is guaranteed that $\sum n \le 2 \cdot 10^6$, $\sum q \le 5 \cdot 10^7$, and n is even.

Output

Output T lines. The i-th line must contain the answer to the i-th test case.

Example

standard input	standard output
0 2 13 4 2 1 3 1 4 2 3 6 2 3 3 6 2 5 3 1 4	5 499122191

Note

In the first test case of the sample, $\{a_i\}$ are $\{1, 4, 2, 3\}$ initially.

- After one operation, it becomes $\{2, 2, 3, 3\}$.
- After two operations, it becomes $\{\frac{5}{2}, \frac{5}{2}, \frac{5}{2}, \frac{5}{2}\}$.
- The generated queries are:
- The position is $x = 1$;
- The first query: $k = 0$, $a_x = 1$;
- The second query: $k = 1$, $a_x = 2$;
- The answer is $(1 \cdot 1) \oplus (2 \cdot 2) = 5$.

In the second test case, $\{a_i\}$ is $\{6, 2, 5, 3, 1, 4\}$ initially.

- After one operation, it becomes $\{5, 5, \frac{3}{2}, \frac{3}{2}, 4, 4\}$.
- After two operations, it becomes $\{\frac{9}{2}, \frac{9}{2}, \frac{9}{2}, \frac{9}{2}, \frac{3}{2}, \frac{3}{2}\}$.
- The generated queries are:
- The position is $x = 3$;
- The first query: $k = 2$, $a_x = \frac{9}{2}$, and $\frac{9}{2} \equiv 499\,122\,181 \pmod{998\,244\,353}$;
- The second query: $k = 0$, $a_x = 5$.
- The answer is $(499\,122\,181 \cdot 1) \oplus (5 \cdot 2) = 499\,122\,181 \oplus 10 = 499\,122\,191$.

The generator will be given below:

```
#include <bits/stdc++.h>
using namespace std;

unsigned long long rd (unsigned long long &x) {
  x ^= (x << 13);
  x ^= (x >> 7);
  x ^= (x << 17);
  return x;
}

int main () {
  int test, T;
  unsigned long long seed;
  scanf("%d%d%llu", &test, &T, &seed);
  for (int Case = 1; Case <= T; Case ++) {
    int n, q, x;
    long long k_max;
    scanf("%d%d%d%lld", &n, &q, &x, &k_max);
    vector<int> a(n + 1);
    for (int i = 1; i <= n; i ++) {
      scanf("%d", &a[i]);
    }
    for (int i = 1; i <= q; i ++) {
      long long k = rd(seed) % k_max;
      /*
      Code your solution here.
      */
    }
  }
}
```