Educational Codeforces Round 86 (Rated for Div. 2)

A. Road To Zero

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

You are given two integers x and y. You can perform two types of operations:

1. Pay a dollars and increase or decrease any of these integers by 1. For example, if $x = 0$ and $y = 7$ there are four possible outcomes after this operation:
   - $x = 0$, $y = 6$;
   - $x = 0$, $y = 8$;
   - $x = -1$, $y = 7$;
   - $x = 1$, $y = 7$.
2. Pay b dollars and increase or decrease both integers by 1. For example, if $x = 0$ and $y = 7$ there are two possible outcomes after this operation:
   - $x = -1$, $y = 6$;
   - $x = 1$, $y = 8$.

Your goal is to make both given integers equal zero simultaneously, i.e. $x = y = 0$. There are no other requirements. In particular, it is possible to move from $x = 1$, $y = 0$ to $x = y = 0$.

Calculate the minimum amount of dollars you have to spend on it.

Input

The first line contains one integer t ($1 \le t \le 100$) — the number of testcases.

The first line of each test case contains two integers x and y ($0 \le x, y \le 10^9$).

The second line of each test case contains two integers a and b ($1 \le a, b \le 10^9$).

Output

For each test case print one integer — the minimum amount of dollars you have to spend.

Example

input

```
2
1 3
391 555
0 0
9 4
```

output

```
1337
0
```

Note

In the first test case you can perform the following sequence of operations: first, second, first. This way you spend $391 + 555 + 391 = 1337$ dollars.

In the second test case both integers are equal to zero initially, so you dont' have to spend money.

B. Binary Period

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Let's say string s has period k if $s_i = s_{i+k}$ for all i from 1 to $|s| - k$ ($|s|$ means length of string s) and k is the minimum positive integer with this property.

Some examples of a period: for s="0101" the period is $k = 2$, for s="0000" the period is $k = 1$, for s="010" the period is $k = 2$, for s="0011" the period is $k = 4$.

You are given string t consisting only of 0's and 1's and you need to find such string s that:

1. String s consists only of 0's and 1's;
2. The length of s doesn't exceed $2 \cdot |t|$;
3. String t is a subsequence of string s;
4. String s has smallest possible period among all strings that meet conditions 1—3.

Let us recall that t is a subsequence of s if t can be derived from s by deleting zero or more elements (any) without changing the order of the remaining elements. For example, t="011" is a subsequence of s="10101".

Input

The first line contains single integer T ($1 \le T \le 100$) — the number of test cases.

Next T lines contain test cases — one per line. Each line contains string t ($1 \le |t| \le 100$) consisting only of 0's and 1's.

Output

Print one string for each test case — string s you needed to find. If there are multiple solutions print any one of them.

Example

input

```
4
00
01
111
110
```

output

```
00
01
11111
1010
```

Note

In the first and second test cases, $s = t$ since it's already one of the optimal solutions. Answers have periods equal to 1 and 2, respectively.

In the third test case, there are shorter optimal solutions, but it's okay since we don't need to minimize the string s. String s has period equal to 1.

C. Yet Another Counting Problem

time limit per test: 3.5 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

You are given two integers a and b, and q queries. The i-th query consists of two numbers l_i and r_i, and the answer to it is the number of integers x such that $l_i \le x \le r_i$, and $((x \bmod a) \bmod b) \neq ((x \bmod b) \bmod a)$. Calculate the answer for each query.

Recall that $y \bmod z$ is the remainder of the division of y by z. For example, $5 \bmod 3 = 2$, $7 \bmod 8 = 7$, $9 \bmod 4 = 1$, $9 \bmod 9 = 0$.

Input

The first line contains one integer t ($1 \le t \le 100$) — the number of test cases. Then the test cases follow.

The first line of each test case contains three integers a, b and q ($1 \le a, b \le 200$; $1 \le q \le 500$).

Then q lines follow, each containing two integers l_i and r_i ($1 \le l_i \le r_i \le 10^{18}$) for the corresponding query.

Output

For each test case, print q integers — the answers to the queries of this test case in the order they appear.

Example

input

```
2
4 6 5
1 1
1 3
1 5
1 7
1 9
7 10 2
7 8
100 200
```

output

```
0 0 0 2 4
0 91
```

D. Multiple Testcases

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

So you decided to hold a contest on Codeforces. You prepared the problems: statements, solutions, checkers, validators, tests... Suddenly, your coordinator asks you to change all your tests to multiple testcases in the easiest problem!

Initially, each test in that problem is just an array. The maximum size of an array is k. For simplicity, the contents of arrays don't matter. You have n tests — the i-th test is an array of size m_i ($1 \le m_i \le k$).

Your coordinator asks you to distribute all of your arrays into multiple testcases. Each testcase can include multiple arrays. However, each testcase should include no more than c_1 arrays of size **greater than or equal to** 1 (≥ 1), no more than c_2 arrays of size **greater than or equal to** 2, $\dots$, no more than c_k arrays of size **greater than or equal to** k. Also, $c_1 \ge c_2 \ge \dots \ge c_k$.

So now your goal is to create the new testcases in such a way that:

- each of the initial arrays appears in **exactly one** testcase;
- for each testcase the given conditions hold;
- the number of testcases is minimum possible.

Print the minimum possible number of testcases you can achieve and the sizes of arrays included in each testcase.

Input

The first line contains two integers n and k ($1 \le n, k \le 2 \cdot 10^5$) — the number of initial tests and the limit for the size of each array.

The second line contains n integers $m_1, m_2, \dots, m_n$ ($1 \le m_i \le k$) — the sizes of the arrays in the original tests.

The third line contains k integers $c_1, c_2, \dots, c_k$ ($n \ge c_1 \ge c_2 \ge \dots \ge c_k \ge 1$); c_i is the maximum number of arrays of size greater than or equal to i you can have in a single testcase.

Output

In the first line print a single integer ans ($1 \le ans \le n$) — the minimum number of testcases you can achieve.

Each of the next ans lines should contain the description of a testcase in the following format:

t a_1 a_2 $\dots$ a_t ($1 \le t \le n$) — the testcase includes t arrays, a_i is the size of the i-th array in that testcase.

Each of the initial arrays should appear in **exactly one** testcase. In particular, it implies that the sum of t over all ans testcases should be equal to n.

Note that the answer always exists due to $c_k \ge 1$ (and therefore $c_1 \ge 1$).

If there are multiple answers, you can output any one of them.

Examples

input
```
4 3
1 2 2 3
4 1 1
```
output
```
3
1 2
2 1 3
1 2
```

input
```
6 10
5 8 1 10 8 7
6 6 4 4 3 2 2 2 1 1
```
output
```
2
3 8 5 7
3 10 8 1
```

input
```
5 1
1 1 1 1 1
5
```
output
```
1
5 1 1 1 1 1
```

input

```
5 1
1 1 1 1 1
1
```

output

```
5
1 1
1 1
1 1
1 1
1 1
```

Note

In the first example there is no way to distribute the tests into less than 3 testcases. The given answer satisfies the conditions: each of the testcases includes no more than 4 arrays of size greater than or equal to 1 and no more than 1 array of sizes greater than or equal to 2 and 3.

Note that there are multiple valid answers for this test. For example, testcases with sizes $[[2], [1, 2], [3]]$ would also be correct.

However, testcases with sizes $[[1, 2], [2, 3]]$ would be incorrect because there are 2 arrays of size greater than or equal to 2 in the second testcase.

Note the difference between the third and the fourth examples. You can include up to 5 arrays of size greater than or equal to 1 in the third example, so you can put all arrays into a single testcase. And you can have only up to 1 array in the fourth example. Thus, every array should be included in a separate testcase.

E. Placing Rooks

time limit per test: 2 seconds

memory limit per test: 512 megabytes

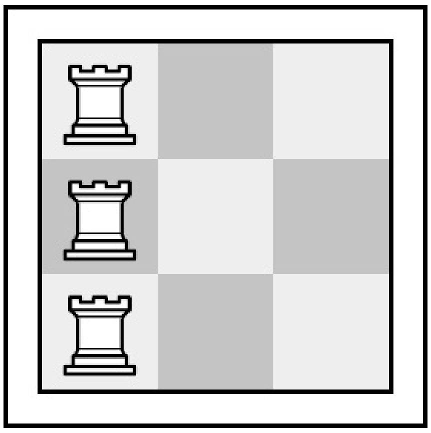
input: standard input

output: standard output

Calculate the number of ways to place n rooks on $n \times n$ chessboard so that both following conditions are met:

- each empty cell is under attack;
- exactly k pairs of rooks attack each other.

An empty cell is under attack if there is at least one rook in the same row or at least one rook in the same column. Two rooks attack each other if they share the same row or column, *and there are no other rooks between them*. For example, there are only two pairs of rooks that attack each other in the following picture:


One of the ways to place the rooks for $n = 3$ and $k = 2$

Two ways to place the rooks are considered different if there exists at least one cell which is empty in one of the ways but contains a rook in another way.

The answer might be large, so print it modulo 998244353.

Input

The only line of the input contains two integers n and k ($1 \le n \le 200000$; $0 \le k \le \frac{n(n-1)}{2}$).

Output

Print one integer — the number of ways to place the rooks, taken modulo 998244353.

Examples

input

```
3 2
```

output

```
6
```

input

```
3 3
```

output

```
0
```

input

```
4 0
```

output

```
24
```

input

```
1337 42
```

output

```
807905441
```

F. Make It Ascending

time limit per test: 7 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

You are given an array a consisting of n elements. You may apply several operations (possibly zero) to it.

During each operation, you choose two indices i and j ($1 \le i, j \le n$; $i \ne j$), increase a_j by a_i, and remove the i-th element from the array (so the indices of all elements to the right to it decrease by 1, and n also decreases by 1).

Your goal is to make the array a strictly ascending. That is, the condition $a_1 < a_2 < \dots < a_n$ should hold (where n is the resulting size of the array).

Calculate the minimum number of actions required to make the array strictly ascending.

Input

The first line contains one integer T ($1 \le T \le 10000$) — the number of test cases.

Each test case consists of two lines. The first line contains one integer n ($1 \le n \le 15$) — the number of elements in the initial array a.

The second line contains n integers a_1, a_2, ..., a_n ($1 \le a_i \le 10^6$).

It is guaranteed that:

- the number of test cases having $n \ge 5$ is not greater than 5000;
- the number of test cases having $n \ge 8$ is not greater than 500;
- the number of test cases having $n \ge 10$ is not greater than 100;
- the number of test cases having $n \ge 11$ is not greater than 50;
- the number of test cases having $n \ge 12$ is not greater than 25;
- the number of test cases having $n \ge 13$ is not greater than 10;
- the number of test cases having $n \ge 14$ is not greater than 3;
- the number of test cases having $n \ge 15$ is not greater than 1.

Output

For each test case, print the answer as follows:

In the first line, print k — the minimum number of operations you have to perform. Then print k lines, each containing two indices i and j for the corresponding operation. Note that the numeration of elements in the array changes after removing elements from it. If there are multiple optimal sequences of operations, print any one of them.

Example

input

```
4
8
2 1 3 5 1 2 4 5
15
16384 8192 4096 2048 1024 512 256 128 64 32 16 8 4 2 1
2
3 3
14
1 2 3 4 5 6 7 8 9 10 11 12 13 14
```

output

```
3
6 8
1 6
4 1
```

```
7
1 15
1 13
1 11
1 9
1 7
1 5
1 3
1
2 1
0
```

Note

In the first test case, the sequence of operations changes a as follows:

$$[2, 1, 3, 5, 1, 2, 4, 5] \to [2, 1, 3, 5, 1, 4, 7] \to [1, 3, 5, 1, 6, 7] \to [2, 3, 5, 6, 7].$$