A. Problemsolving Log

```
Input file:     standard input
Output file:    standard output
Time limit:     2 seconds
Memory limit:   256 megabytes
```

Monocarp is participating in a programming contest, which features 26 problems, named from 'A' to 'Z'. The problems are sorted by difficulty. Moreover, it's known that Monocarp can solve problem 'A' in 1 minute, problem 'B' in 2 minutes, ..., problem 'Z' in 26 minutes.

After the contest, you discovered his contest log — a string, consisting of uppercase Latin letters, such that the i-th letter tells which problem Monocarp was solving during the i-th minute of the contest. If Monocarp had spent enough time in total on a problem to solve it, he solved it. Note that Monocarp could have been thinking about a problem after solving it.

Given Monocarp's contest log, calculate the number of problems he solved during the contest.

Input

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

The first line of each testcase contains a single integer n $(1 \le n \le 500)$ — the duration of the contest, in minutes.

The second line contains a string of length exactly n, consisting only of uppercase Latin letters, — Monocarp's contest log.

Output

For each testcase, print a single integer — the number of problems Monocarp solved during the contest.

Standard Input	Standard Output
3 6 ACBCBC 7 AAAAFPC 22 FEADBBDFFEDFFFDHHHADCC	3 1 4

,

B. Preparing for the Contest

```
Input file:     standard input
Output file:    standard output
Time limit:     1 second
Memory limit:   256 megabytes
```

Monocarp is practicing for a big contest. He plans to solve n problems to make sure he's prepared. Each of these problems has a difficulty level: the first problem has a difficulty level of 1, the second problem has a difficulty level of 2, and so on, until the last (n-th) problem, which has a difficulty level of n.

Monocarp will choose some order in which he is going to solve all n problems. Whenever he solves a problem which is more difficult than the last problem he solved, he gets excited because he feels like he's progressing. He doesn't get excited when he solves the first problem in his chosen order.

For example, if Monocarp solves the problems in the order $[3, \underline{5}, 4, 1, \underline{6}, 2]$, he gets excited twice (the corresponding problems are underlined).

Monocarp wants to get excited exactly k times during his practicing session. Help him to choose the order in which he has to solve the problems!

Input

The first line contains one integer t ($1 \le t \le 1000$) — the number of test cases.

Each test case consists of one line containing two integers n and k ($2 \le n \le 50; 0 \le k \le n-1$).

Output

For each test case, print n **distinct** integers from 1 to n, denoting the order in which Monocarp should solve the problems. If there are multiple answers, print any of them.

It can be shown that under the constraints of the problem, the answer always exists.

Standard Input	Standard Output

3
6 2
5 4
5 0
	3 5 4 1 6 2
	1 2 3 4 5
	5 4 3 2 1

,

C. Quests

```
Input file:     standard input
Output file:    standard output
Time limit:     2.5 seconds
Memory limit:   256 megabytes
```

Monocarp is playing a computer game. In order to level up his character, he can complete quests. There are n quests in the game, numbered from 1 to n.

Monocarp can complete quests according to the following rules:

- the 1-st quest is always available for completion;
- the i-th quest is available for completion if all quests $j < i$ have been completed at least once.

Note that Monocarp can complete the same quest multiple times.

For each completion, the character gets some amount of experience points:

- for the first completion of the i-th quest, he gets a_i experience points;
- for each subsequent completion of the i-th quest, he gets b_i experience points.

Monocarp is a very busy person, so he has free time to complete no more than k quests. Your task is to calculate the maximum possible total experience Monocarp can get if he can complete no more than k quests.

Input

The first line contains a single integer t ($1 \le t \le 10^4$) — the number of test cases.

The first line of each test case contains two integers n and k ($1 \le n \le 2 \cdot 10^5$; $1 \le k \le 2 \cdot 10^5$) — the number of quests and the maximum number of quests Monocarp can complete, respectively.

The second line contains n integers $a_1, a_2, \dots, a_n$ $(1 \le a_i \le 10^3)$.

The third line contains n integers $b_1, b_2, \dots, b_n$ $(1 \le b_i \le 10^3)$.

Additional constraint on the input: the sum of n over all test cases does not exceed $2 \cdot 10^5$.

Output

For each test case, print a single integer — the maximum possible total experience Monocarp can get if he can complete no more than k quests.

Standard Input	Standard Output

4
4 7
4 3 1 2
1 1 1 1
3 2
1 2 5
3 1 8
5 5
3 2 4 1 4
2 3 1 4 7
6 4
	13
	4
	15
	15

1 4 5 4 5 10	
1 5 1 2 5 1	

Note

In the first test case, one of the possible quest completion sequences is as follows: $1, 1, 2, 3, 2, 4, 4$; its total experience is equal to $\underline{4} + 1 + \underline{3} + \underline{1} + 1 + \underline{2} + 1 = 13$ (the underlined numbers correspond to the instances when we complete a quest for the first time).

In the second test case, one of the possible quest completion sequences is as follows: $1, 1$; its total experience is equal to $\underline{1} + 3 = 4$.

In the third test case, one of the possible quest completion sequences is as follows: $1, 2, 2, 2, 3$; its total experience is equal to $\underline{3} + \underline{2} + 3 + 3 + \underline{4} = 15$.

,

D. Three Activities

```
Input file:     standard input
Output file:    standard output
Time limit:     2 seconds
Memory limit:   256 megabytes
```

Winter holidays are coming up. They are going to last for n days.

During the holidays, Monocarp wants to try all of these activities **exactly once** with his friends:

- go skiing;
- watch a movie in a cinema;
- play board games.

Monocarp knows that, on the i-th day, exactly a_i friends will join him for skiing, b_i friends will join him for a movie and c_i friends will join him for board games.

Monocarp also knows that he can't try more than one activity in a single day.

Thus, he asks you to help him choose three **distinct** days x, y, z in such a way that the total number of friends to join him for the activities ($a_x + b_y + c_z$) is maximized.

Input

The first line contains a single integer t ($1 \le t \le 10^4$) — the number of testcases.

The first line of each testcase contains a single integer n ($3 \le n \le 10^5$) — the duration of the winter holidays in days.

The second line contains n integers $a_1, a_2, \dots, a_n$ ($1 \le a_i \le 10^8$) — the number of friends that will join Monocarp for skiing on the i-th day.

The third line contains n integers $b_1, b_2, \dots, b_n$ ($1 \le b_i \le 10^8$) — the number of friends that will join Monocarp for a movie on the i-th day.

The fourth line contains n integers $c_1, c_2, \dots, c_n$ ($1 \le c_i \le 10^8$) — the number of friends that will join Monocarp for board games on the i-th day.

The sum of n over all testcases doesn't exceed 10^5.

Output

For each testcase, print a single integer — the maximum total number of friends that can join Monocarp for the activities on three distinct days.

Standard Input	Standard Output

4
3
1 10 1
10 1 1
1 1 10
4
30 20 10 1
30 5 15 20
	30
	75
	55
	56

```
30 25 10 10
10
5 19 12 3 18 18 6 17 10 13
15 17 19 11 16 3 11 17 17 17
1 17 18 10 15 8 17 3 13 12
10
17 5 4 18 12 4 11 2 16 16
8 4 14 19 3 12 6 7 5 16
3 4 8 11 10 8 10 2 20 3
```

Note

In the first testcase, Monocarp can choose day 2 for skiing, day 1 for a movie and day 3 for board games. This way, $a_2 = 10$ friends will join him for skiing, $b_1 = 10$ friends will join him for a movie and $c_3 = 10$ friends will join him for board games. The total number of friends is 30.

In the second testcase, Monocarp can choose day 1 for skiing, day 4 for a movie and day 2 for board games. $30 + 20 + 25 = 75$ friends in total. Note that Monocarp can't choose day 1 for all activities, because he can't try more than one activity in a single day.

In the third testcase, Monocarp can choose day 2 for skiing, day 3 for a movie and day 7 for board games. $19 + 19 + 17 = 55$ friends in total.

In the fourth testcase, Monocarp can choose day 1 for skiing, day 4 for a movie and day 9 for board games. $17 + 19 + 20 = 56$ friends in total.

E1. Game with Marbles (Easy Version)

```
Input file:     standard input
Output file:    standard output
Time limit:     3.5 seconds
Memory limit:   256 megabytes
```

The easy and hard versions of this problem differ only in the constraints on the number of test cases and n. In the easy version, the number of test cases does not exceed 10^3, and n does not exceed 6.

Recently, Alice and Bob were given marbles of n different colors by their parents. Alice has received a_1 marbles of color 1, a_2 marbles of color 2,..., a_n marbles of color n. Bob has received b_1 marbles of color 1, b_2 marbles of color 2, ..., b_n marbles of color n. All a_i and b_i are between 1 and 10^9.

After some discussion, Alice and Bob came up with the following game: players take turns, starting with Alice. On their turn, a player chooses a color i such that **both** players have at least one marble of that color. The player then discards *one marble* of color i, and their opponent discards *all marbles* of color i. The game ends when there is no color i such that both players have at least one marble of that color.

The score in the game is the difference between the number of remaining marbles that Alice has and the number of remaining marbles that Bob has at the end of the game. In other words, the score in the game is equal to $(A - B)$, where A is the number of marbles Alice has and B is the number of marbles Bob has at the end of the game. Alice wants to maximize the score, while Bob wants to minimize it.

Calculate the score at the end of the game if both players play optimally.

Input

The first line contains a single integer t ($1 \le t \le 10^3$) — the number of test cases.

Each test case consists of three lines:

- the first line contains a single integer n ($2 \le n \le 6$) — the number of colors;
- the second line contains n integers $a_1, a_2, \dots, a_n$ ($1 \le a_i \le 10^9$), where a_i is the number of marbles of the i-th color that Alice has;
- the third line contains n integers $b_1, b_2, \dots, b_n$ ($1 \le b_i \le 10^9$), where b_i is the number of marbles of the i-th color that Bob has.

Output

For each test case, output a single integer — the score at the end of the game if both Alice and Bob act optimally.

Standard Input	Standard Output
5	1
3	-9
4 2 1	2999999997
1 2 4	8
4	-6
1 20 1 20	
100 15 10 20	
5	

```
1000000000 1000000000 1000000000 1000000000
1000000000
1 1 1 1 1
3
5 6 5
2 1 7
6
3 2 4 2 5 5
9 4 7 9 2 5
```

Note

In the first example, one way to achieve a score of 1 is as follows:

1. Alice chooses color 1, discards 1 marble. Bob also discards 1 marble;
2. Bob chooses color 3, discards 1 marble. Alice also discards 1 marble;
3. Alice chooses color 2, discards 1 marble, and Bob discards 2 marble.

As a result, Alice has $a = [3, 1, 0]$ remaining, and Bob has $b = [0, 0, 3]$ remaining. The score is $3 + 1 - 3 = 1$.

It can be shown that neither Alice nor Bob can achieve a better score if both play optimally.

In the second example, Alice can first choose color 1, then Bob will choose color 4, after which Alice will choose color 2, and Bob will choose color 3. It can be shown that this is the optimal game.

,

E2. Game with Marbles (Hard Version)

```
Input file:     standard input
Output file:    standard output
Time limit:     3.5 seconds
Memory limit:   256 megabytes
```

The easy and hard versions of this problem differ only in the constraints on the number of test cases and n. In the hard version, the number of test cases does not exceed 10^4, and the sum of values of n over all test cases does not exceed $2 \cdot 10^5$. Furthermore, there are no additional constraints on n in a single test case.

Recently, Alice and Bob were given marbles of n different colors by their parents. Alice has received a_1 marbles of color 1, a_2 marbles of color 2,..., a_n marbles of color n. Bob has received b_1 marbles of color 1, b_2 marbles of color 2, ..., b_n marbles of color n. All a_i and b_i are between 1 and 10^9.

After some discussion, Alice and Bob came up with the following game: players take turns, starting with Alice. On their turn, a player chooses a color i such that **both** players have at least one marble of that color. The player then discards *one marble* of color i, and their opponent discards *all marbles* of color i. The game ends when there is no color i such that both players have at least one marble of that color.

The score in the game is the difference between the number of remaining marbles that Alice has and the number of remaining marbles that Bob has at the end of the game. In other words, the score in the game is equal to $(A - B)$, where A is the number of marbles Alice has and B is the number of marbles Bob has at the end of the game. Alice wants to maximize the score, while Bob wants to minimize it.

Calculate the score at the end of the game if both players play optimally.

Input

The first line contains a single integer t ($1 \le t \le 10^4$) — the number of test cases.

Each test case consists of three lines:

- the first line contains a single integer n ($2 \le n \le 2 \cdot 10^5$) — the number of colors;
- the second line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 10^9$), where a_i is the number of marbles of the i-th color that Alice has;
- the third line contains n integers $b_1, b_2, \ldots, b_n$ ($1 \le b_i \le 10^9$), where b_i is the number of marbles of the i-th color that Bob has.

Additional constraint on the input: the sum of n for all test cases does not exceed $2 \cdot 10^5$.

Output

For each test case, output a single integer — the score at the end of the game if both Alice and Bob act optimally.

Standard Input	Standard Output

5
3
4 2 1
1 2 4
4
1 20 1 20
	1
	-9
	2999999997
	8
	-6

```
100 15 10 20
5
1000000000 1000000000 1000000000 1000000000
1000000000
1 1 1 1 1
3
5 6 5
2 1 7
6
3 2 4 2 5 5
9 4 7 9 2 5
```

Note

In the first example, one way to achieve a score of 1 is as follows:

1. Alice chooses color 1, discards 1 marble. Bob also discards 1 marble;
2. Bob chooses color 3, discards 1 marble. Alice also discards 1 marble;
3. Alice chooses color 2, discards 1 marble, and Bob discards 2 marble.

As a result, Alice has $a = [3, 1, 0]$ remaining, and Bob has $b = [0, 0, 3]$ remaining. The score is $3 + 1 - 3 = 1$.

It can be shown that neither Alice nor Bob can achieve a better score if both play optimally.

In the second example, Alice can first choose color 1, then Bob will choose color 4, after which Alice will choose color 2, and Bob will choose color 3. It can be shown that this is the optimal game.

,

F. Programming Competition

```
Input file:     standard input
Output file:    standard output
Time limit:     2 seconds
Memory limit:   256 megabytes
```

BerSoft is the biggest IT corporation in Berland. There are n employees at BerSoft company, numbered from 1 to n.

The first employee is the head of the company, and he does not have any superiors. Every other employee i has exactly one direct superior p_i.

Employee x is considered to be a superior (direct or indirect) of employee y if one of the following conditions holds:

- employee x is the direct superior of employee y;
- employee x is a superior of the direct superior of employee y.

The structure of BerSoft is organized in such a way that the head of the company is superior of every employee.

A programming competition is going to be held soon. Two-person teams should be created for this purpose. However, if one employee in a team is the superior of another, they are uncomfortable together. So, teams of two people should be created so that no one is the superior of the other. Note that no employee can participate in more than one team.

Your task is to calculate the maximum possible number of teams according to the aforementioned rules.

Input

The first line contains a single integer t ($1 \le t \le 10^4$) — the number of test cases.

The first line of each test case contains a single integer n ($2 \le n \le 2 \cdot 10^5$) — the number of employees.

The second line contains $n-1$ integers $p_2, p_3, \dots, p_n$ ($1 \le p_i \le n$), where p_i is the index of the direct superior of the i-th employee.

The sum of n over all test cases doesn't exceed $2 \cdot 10^5$.

Output

For each test case, print a single integer — the maximum possible number of teams according to the aforementioned rules.

Standard Input	Standard Output

6
4
1 2 1
2
1
5
5 5 5 1
7
	1
	0
	1
	3
	3
	3

```
1 2 1 1 3 3
7
1 1 3 2 2 4
7
1 2 1 1 1 3
```

Note

In the first test case, team $(3, 4)$ can be created.

In the second test case, no team can be created, because there are only 2 employees and one is the superior of another.

In the third test case, team $(2, 3)$ can be created.

In the fourth test case, teams $(2, 4)$, $(3, 5)$ and $(6, 7)$ can be created.

In the fifth test case, teams $(2, 3)$, $(6, 4)$ and $(5, 7)$ can be created.

,

G1. Light Bulbs (Easy Version)

```
Input file:     standard input
Output file:    standard output
Time limit:     3 seconds
Memory limit:   512 megabytes
```

The easy and hard versions of this problem differ only in the constraints on n. In the easy version, the sum of values of n^2 over all test cases does not exceed 10^6. Furthermore, n does not exceed 1000 in each test case.

There are $2n$ light bulbs arranged in a row. Each light bulb has a color from 1 to n (**exactly two light bulbs for each color**).

Initially, all light bulbs are turned off. You choose a set of light bulbs S that you initially turn on. After that, you can perform the following operations in any order any number of times:

- choose two light bulbs i and j **of the same color**, exactly one of which is on, and turn on the second one;
- choose three light bulbs i, j, k, such that both light bulbs i and k **are on and have the same color**, and the light bulb j is between them ($i < j < k$), and turn on the light bulb j.

You want to choose a set of light bulbs S that you initially turn on in such a way that by performing the described operations, you can ensure that all light bulbs are turned on.

Calculate two numbers:

- the minimum size of the set S that you initially turn on;
- the number of sets S of minimum size (taken modulo 998244353).

Input

The first line of the input contains a single integer t ($1 \le t \le 10^4$) — the number of test cases. Then follow the descriptions of the test cases.

The first line of each test case contains a single integer n ($2 \le n \le 1000$) — the number of pairs of light bulbs.

The second line of each test case contains $2n$ integers $c_1, c_2, \ldots, c_{2n}$ ($1 \le c_i \le n$), where c_i is the color of the i-th light bulb. For each color from 1 to n, exactly two light bulbs have this color.

Additional constraint on the input: the sum of values of n^2 over all test cases does not exceed 10^6.

Output

For each test case, output two integers:

- the minimum size of the set S that you initially turn on;
- the number of sets S of minimum size (taken modulo 998244353).

Standard Input	Standard Output

4
2
2 2 1 1
	2 4
	1 2

2
1 2 2 1
2
1 2 1 2
5
3 4 4 5 3 1 1 5 2 2
	1 4
	2 8

G2. Light Bulbs (Hard Version)

```
Input file:     standard input
Output file:    standard output
Time limit:     3 seconds
Memory limit:   512 megabytes
```

The easy and hard versions of this problem differ only in the constraints on n. In the hard version, the sum of values of n over all test cases does not exceed $2 \cdot 10^5$. Furthermore, there are no additional constraints on the value of n in a single test case.

There are $2n$ light bulbs arranged in a row. Each light bulb has a color from 1 to n (**exactly two light bulbs for each color**).

Initially, all light bulbs are turned off. You choose a set of light bulbs S that you initially turn on. After that, you can perform the following operations in any order any number of times:

- choose two light bulbs i and j **of the same color**, exactly one of which is on, and turn on the second one;
- choose three light bulbs i, j, k, such that both light bulbs i and k **are on and have the same color**, and the light bulb j is between them ($i < j < k$), and turn on the light bulb j.

You want to choose a set of light bulbs S that you initially turn on in such a way that by performing the described operations, you can ensure that all light bulbs are turned on.

Calculate two numbers:

- the minimum size of the set S that you initially turn on;
- the number of sets S of minimum size (taken modulo 998244353).

Input

The first line of the input contains a single integer t ($1 \le t \le 10^4$) — the number of test cases. Then follow the descriptions of the test cases.

The first line of each test case contains a single integer n ($2 \le n \le 2 \cdot 10^5$) — the number of pairs of light bulbs.

The second line of each test case contains $2n$ integers $c_1, c_2, \ldots, c_{2n}$ ($1 \le c_i \le n$), where c_i is the color of the i-th light bulb. For each color from 1 to n, exactly two light bulbs have this color.

Additional constraint on the input: the sum of values of n over all test cases does not exceed $2 \cdot 10^5$.

Output

For each test case, output two integers:

- the minimum size of the set S that you initially turn on;
- the number of sets S of minimum size (taken modulo 998244353).

Standard Input	Standard Output

4
2
2 2 1 1
	2 4
	1 2

2
1 2 2 1
2
1 2 1 2
5
3 4 4 5 3 1 1 5 2 2
	1 4
	2 8