Educational Codeforces Round 170 (Rated for Div. 2)

A. Two Screens

2 seconds, 512 megabytes

There are two screens which can display sequences of uppercase Latin letters. Initially, both screens display nothing.

In one second, you can do one of the following two actions:

- choose a screen and an uppercase Latin letter, and append that letter to **the end** of the sequence displayed on that screen;
- choose a screen and copy the sequence from it to the other screen, **overwriting the sequence that was displayed on the other screen**.

You have to calculate the minimum number of seconds you have to spend so that the first screen displays the sequence s, and the second screen displays the sequence t.

Input

The first line contains one integer q $(1 \le q \le 500)$ — the number of test cases.

Each test case consists of two lines. The first line contains the string s, and the second line contains the string t ($1 \le |s|, |t| \le 100$). Both strings consist of uppercase Latin letters.

Output

For each test case, print one integer — the minimum possible number of seconds you have to spend so that the first screen displays the sequence s, and the second screen displays the sequence t.

input

```
3
GARAGE
GARAGEFORSALE
ABCDE
AABCD
TRAINING
DRAINING
```

output

```
14
10
16
```

In the first test case, the following sequence of actions is possible:

- spend 6 seconds to write the sequence `GARAGE` on the first screen;
- copy the sequence from the first screen to the second screen;
- spend 7 seconds to complete the sequence on the second screen by writing `FORSALE`.

In the second test case, the following sequence of actions is possible:

- spend 1 second to write the sequence `A` on the second screen;
- copy the sequence from the second screen to the first screen;
- spend 4 seconds to complete the sequence on the first screen by writing `BCDE`;
- spend 4 seconds to complete the sequence on the second screen by writing `ABCD`.

In the third test case, the fastest way to display the sequences is to type both of them character by character without copying, and this requires 16 seconds.

B. Binomial Coefficients, Kind Of

2 seconds, 512 megabytes

Recently, akshiM met a task that needed binomial coefficients to solve. He wrote a code he usually does that looked like this:

```
    for (int n = 0; n < N; n++) { // loop over n from 0 to N-1 (inclusive)
        C[n][0] = 1;
        C[n][n] = 1;
        for (int k = 1; k < n; k++) // loop over k from 1 to n-1 (inclusive)
            C[n][k] = C[n][k - 1] + C[n - 1][k - 1];
    }
```

Unfortunately, he made an error, since the right formula is the following:

```
            C[n][k] = C[n - 1][k] + C[n - 1][k - 1]
```

But his team member keblidA is interested in values that were produced using the wrong formula. Please help him to calculate these coefficients for t various pairs (n_i, k_i). Note that they should be calculated according to the first (wrong) formula.

Since values $C[n_i][k_i]$ may be too large, print them modulo $10^9 + 7$.

Input

The first line contains a single integer t $(1 \le t \le 10^5)$ — the number of pairs. Next, t pairs are written in two lines.

The second line contains t integers $n_1, n_2, \dots, n_t$ $(2 \le n_i \le 10^5)$.

The third line contains t integers $k_1, k_2, \dots, k_t$ $(1 \le k_i < n_i)$.

Output

Print t integers $C[n_i][k_i]$ modulo $10^9 + 7$.

input

```
7
2 5 5 100000 100000 100000 100000
1 2 3 1 33333 66666 99999
```

output

```
2
4
8
2
326186014
984426998
303861760
```

C. New Game

2 seconds, 512 megabytes

There's a new game Monocarp wants to play. The game uses a deck of n cards, where the i-th card has exactly one integer a_i written on it.

At the beginning of the game, on the first turn, Monocarp can take any card from the deck. During each subsequent turn, Monocarp can take exactly one card that has either the same number as on the card taken on the previous turn or a number that is one greater than the number on the card taken on the previous turn.

In other words, if on the previous turn Monocarp took a card with the number x, then on the current turn he can take either a card with the number x or a card with the number $x + 1$. Monocarp can take any card which meets that condition, regardless of its position in the deck.

After Monocarp takes a card on the current turn, it is removed from the deck.

According to the rules of the game, the number of distinct numbers written on the cards that Monocarp has taken must not exceed k.

If, after a turn, Monocarp cannot take a card without violating the described rules, the game ends.

Your task is to determine the maximum number of cards that Monocarp can take from the deck during the game, given that on the first turn he can take any card from the deck.

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 ($1 \le k \le n \le 200\,000$) — the number of cards in the deck and the maximum number of distinct numbers that can be written on the cards that Monocarp takes.

The second line contains a sequence of integers $a_1, a_2, \dots, a_n$ ($1 \le a_i \le 10^9$), where a_i is the number written on the i-th card.

Additional constraint of the input: the sum of n over all test cases doesn't exceed $200\,000$.

Output

For each test case, print the maximum number of cards that Monocarp can take from the deck during the game, given that on the first turn he can take any card from the deck.

input

```
4
10 2
5 2 4 3 4 3 4 5 3 2
5 1
10 11 10 11 10
9 3
4 5 4 4 6 5 4 4 6
3 2
1 3 1
```

output

```
6
3
9
2
```

In the first example, Monocarp needs to take any of the cards with the number 3. On the next two turns, he needs to take the two remaining cards with the number 3. On the next three turns, he needs to take three cards with the number 4. After that, Monocarp will not be able to take any more cards from the deck, and he will have 6 cards.

The problem statement has recently been changed. View the changes. ×

D. Attribute Checks

2.5 seconds, 512 megabytes

Imagine a game where you play as a character that has two attributes: "Strength" and "Intelligence", that are at zero level initially.

During the game, you'll acquire m attribute points that allow you to increase your attribute levels — one point will increase one of the attributes by one level. But sometimes, you'll encounter a so-called "Attribute Checks": if your corresponding attribute is high enough, you'll pass it; otherwise, you'll fail it.

Spending some time, you finally prepared a list which contains records of all points you got and all checks you've met. And now you're wondering: what is the maximum number of attribute checks you can pass in a single run if you'd spend points wisely?

Note that you can't change the order of records.

Input

The first line contains two integers n and m ($1 \le m \le 5000$; $m < n \le 2 \cdot 10^6$) — the number of records in the list and the total number of points you'll get during the game.

The second line contains n integers $r_1, r_2, \dots, r_n$ ($-m \le r_i \le m$), where r_i encodes the i-th record:

- If $r_i = 0$, then the i-th record is an acquiring one attribute point. You can spend to level up either Strength or Intelligence;
- If $r_i > 0$, then it's an Intelligence check: if your Intelligence level is greater than or equal to $|r_i|$, you pass.
- If $r_i < 0$, then it's a Strength check: if your Strength level is greater than or equal to $|r_i|$, you pass.

Additional constraint on the input: the sequence $r_1, r_2, \ldots, r_n$ contains exactly m elements equal to 0.

Output

Print one integer — the maximum number of checks you can pass.

input

```
10 5
0 1 0 2 0 -3 0 -4 0 -5
```

output

```
3
```

input

```
3 1
1 -1 0
```

output

```
0
```

input

```
9 3
0 0 1 0 2 -3 -2 -2 1
```

output

```
4
```

In the first test, it's optimal to spend each point in Strength, so you'll fail 2 Intelligence checks but pass 3 Strength checks.

In the second test, you'll fail both checks, since the first point you get comes after the checks.

In the third test, one of the optimal strategies is:

1. spend the first point on Intelligence;
2. spend the second point on Strength;
3. spend the third point on Strength;

As a result, you'll pass 2 Intelligence checks r_3 and r_9 and 2 Strength checks r_7 and r_8.

E. Card Game

2 seconds, 512 megabytes

In the most popular card game in Berland, a deck of $n \times m$ cards is used. Each card has two parameters: suit and rank. Suits in the game are numbered from 1 to n, and ranks are numbered from 1 to m. There is exactly one card in the deck for each combination of suit and rank.

A card with suit a and rank b can beat a card with suit c and rank d in one of two cases:

- $a = 1$, $c \neq 1$ (a card of suit 1 can beat a card of any other suit);
- $a = c$, $b > d$ (a card can beat any other card of the same suit but of a lower rank).

Two players play the game. Before the game starts, they receive exactly half of the deck each. The first player wins if for every card of the second player, he can choose his card that can beat it, and there is no card that is chosen twice (i. e. there exists a matching of the first player's cards with the second player's cards such that in each pair the first player's card beats the second player's card). Otherwise, the second player wins.

Your task is to calculate the number of ways to distribute the cards so that the first player wins. Two ways are considered different if there exists a card such that in one way it belongs to the first player and in the other way it belongs to the second player. The number of ways can be very large, so print it modulo 998244353.

Input

The only line contains two integers n and m ($1 \le n, m \le 500$).

Additional constraint on the input: m is even.

Output

Print a single integer — the number of ways to distribute the cards so that the first player wins, taken modulo 998244353.

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

input
```
2 2
```
output
```
2
```

input
```
3 6
```
output
```
1690
```

input
```
5 4
```
output
```
568
```

input
```
500 500
```
output
```
84693741
```

F. Choose Your Queries

3 seconds, 512 megabytes

You are given an array a, consisting of n integers (numbered from 1 to n). Initially, they are all zeroes.

You have to process q queries. The i-th query consists of two different integers x_i and y_i. During the i-th query, you have to choose an integer p (which is either x_i or y_i) and an integer d (which is either 1 or -1), and assign $a_p = a_p + d$.

After each query, every element of a should be a non-negative integer.

Process all queries in such a way that the sum of all elements of a after the last query is the minimum possible.

Input

The first line contains two integers n and q ($2 \le n \le 3 \cdot 10^5$; $1 \le q \le 3 \cdot 10^5$) — the number of elements in a and the number of queries, respectively.

Then q lines follow. The i-th of these lines contains two integers x_i and y_i ($1 \le x_i, y_i \le n$; $x_i \ne y_i$) — the description of the i-th query.

Output

For each query, print a line containing two characters:

- the first character should be x if you choose $p = x_i$, or y if you choose $p = y_i$;
- the second character should be + if you choose $d = 1$, or - if you choose $d = -1$.

If there are multiple answers, print any of them.

input

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

output

```
y+
x+
x-
y-
```

input

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

output

```
y+
y+
x-
y-
```

input

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

output

```
y+
x+
```

G. Variable Damage

5 seconds, 512 megabytes

Monocarp is gathering an army to fight a dragon in a videogame.

The army consists of two parts: the heroes and the defensive artifacts. Each hero has one parameter — his health. Each defensive artifact also has one parameter — its durability.

Before the battle begins, Monocarp distributes artifacts to the heroes so that each hero receives at most one artifact.

The battle consists of rounds that proceed as follows:

- first, the dragon deals damage equal to $\frac{1}{a+b}$ (**a real number without rounding**) to each hero, where a is the number of heroes alive and b is the number of active artifacts;
- after that, all heroes with health 0 or less die;
- finally, some artifacts are deactivated. An artifact with durability x is deactivated when one of the following occurs: the hero holding the artifact either dies or receives x total damage (from the start of the battle).

The battle ends when there are no heroes left alive.

Initially, the army is empty. There are q queries: add a hero with health x or an artifact with durability y to the army. After each query, determine the maximum number of rounds that Monocarp can survive if he distributes the artifacts optimally.

Input

The first line contains one integer q ($1 \le q \le 3 \cdot 10^5$) — the number of queries.

In the i-th of the following q lines, there are two integers t_i and v_i ($t_i \in \{1, 2\}$; $1 \le v_i \le 10^9$) — the type of the query and the value of the query parameter. If the type is 1, a hero with health v_i is added. If the type is 2, an artifact with durability v_i is added.

Output

Print q integers. After each query, output the maximum number of rounds that Monocarp can survive if he distributes the artifacts optimally.

input

```
3
2 5
1 4
1 10
```

output

```
0
8
19
```

input

```
10
1 9
1 6
2 4
1 8
1 3
2 10
1 3
1 6
1 10
2 6
```

output

```
9
15
19
27
30
39
42
48
59
65
```

Let's consider the first example.

- An artifact with durability 5 is added. Since there are no heroes yet, the battle ends immediately.
- A hero with health 4 is added. Monocarp can give him an artifact with durability 5. First, there are rounds in which the hero takes $\frac{1}{1+1} = \frac{1}{2}$ damage. After 8 such rounds, a total of 4 damage will have been dealt, and the hero will die, while the artifact will deactivate. There are no more heroes alive, so the battle ends after 8 rounds.
- A hero with health 10 is added. Now let the artifact with durability 5 be with this hero. Then, in the first 12 rounds, the heroes will take $12 \cdot \frac{1}{2+1} = 4$ damage, and the first hero will die. The second hero has 6 health left, and the artifact has 1 durability. Now the damage is $\frac{1}{2}$, so after another 2 rounds, the artifact will deactivate. The second hero has 5 health left. After another 5 rounds, the second hero will die. Therefore, the answer is $12 + 2 + 5 = 19$.