CEOI 2026

Day 2

(English)

Flower Cutting

Time limit: 4 s *Memory limit: 256 MiB*

In the CEOI community garden, we are growing a collection of special flowers that tightly knit their roots together. If we cut them, they regrow, as long as we do not cut too aggressively and cause irreparable damage.

If a pair of flowers, a and b, knit their roots together, we call them "connected". Otherwise, they are "disconnected". Roots grow according to the following rule: Let a and b be two disconnected flowers. If at least 2 other flowers c and d exist, such that each of a and b is connected with both c and d, then roots between a and b will grow, and they will become connected.

These flowers have now been growing for a while, and all the roots that could grow by following the above rule have grown. In other words, if two flowers a and b are both connected with some flowers c and d, then a and b are guaranteed to be connected with each other.

We need to uproot this garden and move it to the next CEOI location. To simplify the migration, we would like to cut as many roots as possible. However, we want the flowers to regrow back into their current state. What is the maximum number of connected pairs of flowers that we may cut so that they still regrow back into their current state? It does not matter how many iterations of growth it would take.

Input

The first line of the input contains two space-separated integers n and m, the number of flowers and the number of existing connections between them. This is followed by m lines, each containing a pair of integers a_i and b_i, indicating that flowers a_i and b_i are connected. Flowers are marked with integers $1 \dots n$. The input is guaranteed to follow the rule described in the task description.

Constraints

- $1 \leq n \leq 1000$
- $1 \leq m \leq 10^5$

Subtasks

- Subtask 1 (20 points): $n \leq 10$ and $m \leq 20$
- Subtask 2 (14 points): $m = \frac{n \cdot (n-1)}{2}$
- Subtask 3 (15 points): We guarantee that each flower is connected with at most 7 other flowers.
- Subtask 4 (15 points): $n \leq 50$ and $m \leq 1000$
- Subtask 5 (36 points): No additional constraints.

Output

Output a single integer — the maximum number of connections that we may cut.

Example

Input

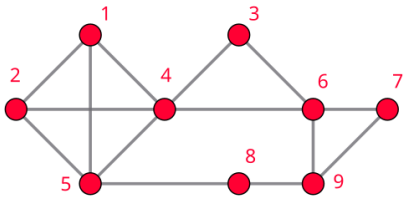
```
9 14
1 2
1 4
1 5
2 4
2 5
3 4
4 5
3 6
4 6
6 7
6 9
7 9
8 9
5 8
```

Output

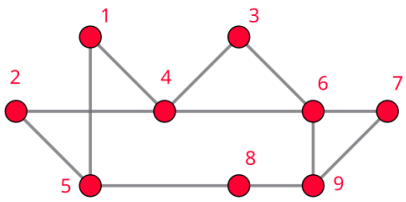
```
2
```

Comment

The flowers in the example before any cutting. One can check that no additional connections can form based on the described growth procedure.



The flowers in the example after the cutting have 2 connections less. The connection between 1 and 2 can regrow because flowers 1 and 2 are both connected to flowers 4 and 5. Similarly, the connection between 4 and 5 can regrow because flowers 4 and 5 are both connected to flowers 1 and 2.



Towers

Time limit: 1 s *Memory limit: 256 MiB*

You have n computers and m towers all at distinct positions along a line. You need to pair computers using cables in such a way that every cable starts at some computer, visits some towers and ends at another computer. The cable can visit any of the towers (not just the ones between the computers) in an arbitrary order. It can skip towers by passing by without visiting them. It can also visit no towers at all and connect the two computers directly but it cannot connect a computer to itself. The number of computers is even.

Let a and b be the positions of two computers and let $x_1, \ldots, x_k$ be positions of the towers the cable visits. The length of the cable is $|a-x_1|+|x_1-x_2|+\cdots+|x_{k-1}-x_k|+|x_k-b|$. For some cable we define its score as $f \cdot u - l$, where l is its length, f is some fixed constant and u is the number of unique towers the cable visits among $x_1, \ldots, x_k$. Multiple cables can visit the same tower and that tower contributes to the score of each of those cables.

You need to compute the maximum possible sum of cable scores for pairing up all computers (i.e. each computer has to belong to exactly one pair, or equivalently, it must be connected to exactly one cable).

Input

The first line contains the number of test cases, T. The test cases follow one after another. Each test case consists of three lines. The first line contains three integers n, m and f — the number of computers, the number of towers and the constant f. The second line contains n integers $a_1, a_2, \ldots, a_n$ — the positions of computers. The third line contains m integers $b_1, b_2, \ldots, b_m$ — the positions of towers.

Constraints

Let N and M be the sum of n and m through all test cases, respectively.

- $1 \leq T \leq 10^4$
- $1 \leq N, M \leq 2 \cdot 10^5$
- $0 \leq f \leq 10^9$
- n is even
- $1 \leq a_i, b_i \leq 10^9$
- All positions of computers and towers are unique (within an individual test case).

Subtasks

- Subtask 1 (5 points): $N \leq 5000,\ m = 1$
- Subtask 2 (10 points): $T \leq 20,\ n \leq 10,\ m \leq 100$
- Subtask 3 (27 points): $N, M \leq 5000$
- Subtask 4 (21 points): $N \leq 5000$
- Subtask 5 (37 points): No additional constraints.

Output

Output T integers, each on its own line — the maximum possible sum of scores of cables for each test case.

Example

Input	Output
4 2 1 100 1 10 11 4 1 10 2 4 6 8 20 4 1 10 2 4 6 8 5 6 3 10 2 13 4 8 6 10 5 1 9	89 -4 12 51

Vim

Time limit: 1 s *Memory limit: 256 MiB*

It would be hard to find anything more *geeky* than the oldschool Vim — an editor in which we can do things that we could only dream of in "regular" editors. That is, if we are willing to invest a week of learning and a month of practice so that the unusual but powerful commands it offers become our "muscle memory". Well, just like in other editors, in Vim we have a *cursor*, which is always located on one of the characters. Initially, it is located on the first character of the text, and just like in other editors, in Vim there is a *clipboard*, which is intended for storing (and indirectly copying and moving) pieces of text. The clipboard is initially empty.

In this task, we will assume that there is exactly one '-' (minus) character in the editor at the beginning, and our goal is to finish with exactly n consecutive minus signs. We will use four commands to help us with this:

- `h`: If the cursor is on the first character, then this command does nothing, otherwise it moves the cursor to the character on the left.
- `l`: If the cursor is on the last character, then this command does nothing, otherwise it moves the cursor to the character on the right.
- `Y`: The sequence of characters extending from the cursor position to the end of the text is copied to the clipboard, "overwriting" any previous contents of the clipboard. (If the cursor is on the first character of the text, the entire text will be copied to the clipboard.)
- `P`: A copy of the text stored in the clipboard is inserted before the character on which the cursor is located, and the cursor is moved to the last inserted character. The contents of the clipboard are not changed. If the clipboard is empty, nothing happens.

Write a program that reads the number n and prints the minimum number of commands needed to finish with exactly n consecutive '-' characters in Vim under the described conditions. The program should also print an example of a sequence with the minimum number of commands.

Input

The first line contains the number of test cases t. The next t lines contain test cases with the desired string length n.

Constraints

- $1 \leq t \leq 100$
- $1 \leq n \leq 10^7$

Subtasks

If you print only the correct number of commands, but no example or an incorrect example of a sequence of commands, you will receive half the points for that subtask.

- Subtask 1 (20 points): $n \leq 100$
- Subtask 2 (8 points): $n \leq 1000$
- Subtask 3 (18 points): $n \leq 10^4$
- Subtask 4 (18 points): $n \leq 10^5$
- Subtask 5 (18 points): $n \leq 10^6$
- Subtask 6 (18 points): No additional constraints.

Output

For each test case, print the required minimum number of commands and an example of such a sequence of commands on its own line.

Example

Input	Output
2 21 2	10 YPYPhPYPPP 2 YP

Comment

As we can see in the following table, in the first case, the sequence `YPYPhPYPPP` gives the correct result. The '=' character in the Screen column represents the '-' character on which the cursor is located.

Step	Command	Screen	Clipboard
0		=	(empty)
1	Y	=	-
2	P	=-	-
3	Y	=-	--
4	P	-=--	--
5	h	=---	--
6	P	-=----	--
7	Y	-=----	-----
8	P	-----=-----	-----
9	P	---------=------	-----
10	P	-------------=-------	-----