Aryan's Thirst for Virat

HackerRank

`Time Limit: 3 seconds`

`Memory Limit: 256MB`

It's no secret Aryan *bhaiya* is obsessed with Virat Kohli. The *hot tea* is that he has publicly stated he would "do *anything*" for an autograph.

Today is his chance. Aryan is at a match it's CSK vs RCB IPL 2026 final, and Virat is on the field for the last over, but a valley of fans ($n \times n$ matrix) stands between them. At position (i, j) is a fan with a "vibe level" of $a_{i,j}$.

If $a_{i,j} < 0$, it's a **"vibe hole"** (a hater, or worse, someone wearing a Dhoni's jersey). Aryan *cannot* proceed if there's any hater energy in his path.

To clear the way, Aryan focuses his intense "anything for him" energy. He can pick a **square** area of fans and unleash a "King Kohli!" chant. This chant is so powerful it increases the "vibe" of everyone on the **main diagonal** of that square by exactly one.

More formally, he can choose a submatrix with the upper left corner at (i, j) and the lower right at (p, q), such that $p - i = q - j$ (a square). He can then add one to each element at $(i + k, j + k)$, for all k such that $0 \leq k \leq p - i$.

Determine the **minimum number of times** Aryan must unleash his chant to clear all the hater energy, so he can finally get his autograph.

Input Format

- The first line contains a single integer T — the number of test cases.
- Each test case is described as follows:
  - The first line of each test case consists of a single number n.
  - Each of the following n lines consists of n integers separated by spaces, which correspond to the vibe levels a.

Constraints

- $1 \leq T \leq 200$
- $1 \leq n \leq 500$
- $-10^9 \leq a_{ij} \leq 10^9$
  NOTE :- There is no limit in the sum of n across all test cases

Tip :- The constrains are really high, I would suggest you to use fast i/o to read 20 times of the standard cin inputs at once (if using other language then compare with standard input) which may save your solution some time. If you dont know what fast i/o is I have already added that in your hackerank snippet but still you may contact one of the problem setter. I will also suggest you not to code in python for this specific question , you may use pypy3 instead.

Output Format

- For each test case, output the minimum number of chants Aryan will have to use.

Sample Input 0

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

Sample Output 0

```
0
1
4
19
```

Explanation 0

Test Case 1 ($N = 1$):

- The grid is:

  1

- The value is positive (1). No "vibe holes" to fix.
- **Total chants: 0**.

Test Case 2 ($N = 2$):

- The grid is:

  -1 2

  3 0

- **Cell (0,0):** Value is -1. We need **1 chant**.
- This adds $+1$ to $(0, 0)$ and propagates to $(1, 1)$.
- New value at $(1, 1)$ becomes $0 + 1 = 1$.

- All other cells are now non-negative.
- **Total chants: 1**.

Test Case 3 ($N = 3$):

- The grid is:

  1 2 3

  -2 1 -1

  0 0 -1

- **Cell (1,0):** Value is -2. We need **2 chants**.
- This propagates $+2$ to cell $(2, 1)$.
- New value at $(2, 1)$ becomes $0 + 2 = 2$.
- **Cell (1,2):** Value is -1. We need **1 chant**.
- Propagates out of bounds.
- **Cell (2,2):** Value is -1. We need **1 chant**.
- **Total chants:** $2 + 1 + 1 = 4$.

Test Case 4 ($N = 5$):

- This is a larger grid. Following the same logic (fixing negative values from top-left to bottom-right and propagating the fix diagonally), the total number of chants required sums up to **19**.

Atharv's "Just Friends" Map

`Time Limit: 1 second`

`Memory Limit: 256MB`

The *hottest tea* is circulating, and it's about Atharv *bhaiya*. He has a secret girlfriend. The tea gets spicier: she's in the **same college**. The tea is scalding: she's in... *that other* tech society. 🤫

This is a high-stakes, cross-society scandal waiting to happen. To keep their relationship on the down-low, they can't just text normally. They've invented a hasty, desperate "isomorphic cipher" to communicate.

Atharv writes a "safe" text (string s) that looks like he's debugging code (e.g., "egg"), which *should* secretly map to his "real" romantic text (string t, e.g., "add").

The mapping is simple:

- The first character in s maps to the first in t.
- Every subsequent occurrence of that same character in s *must* map to that *same* character from t.

The catch: If a single character from s tries to map to *two different* characters in t (e.g., `s = "foo"`, `t = "bar"`), the cipher is broken, the message is unreadable, and he's at risk of being exposed. Even worse, if two *different* characters from s map to the *same* character in t (e.g., `s = "baba"`, `t = "Kiki"`), it's also a failure.

He's paranoid and needs you to be his "vibe check." Before he hits send, write a program to verify his "safe" text s and his "real" text t are truly isomorphic.

Formal Rules: Two strings s and t are isomorphic if:

1. They have the same length.
2. There is a one-to-one mapping (a bijection) between the characters of s and the characters of t, a rule that is followed for the entire string.
3. This means:
   - Every character in s must map to *one and only one* character in t.
   - No two different characters in s may map to the *same* character in t.

Input Format

- The first line contains a single integer T — the number of test cases.
- Each test case is described by two separate lines:
  - The first line contains the string s.
  - The second line contains the string t.

Constraints

- $1 \leq T \leq 1000$
- $1 \leq \text{length}(s), \text{length}(t) \leq 10^5$
- s and t consist of any printable ASCII characters.
- It is guaranteed that the total sum of $\text{length}(s)$ over all test cases does not exceed 10^6.

Output Format

- For each test case, output **'YES'** if the strings are isomorphic, and **'NO'** otherwise.

Note: The output is **not case sensitive**. You can print `YES`, `yes`, `Yes`, `NO`, `no`, `nO`, etc. All will be accepted.

Sample Input 0

```
3
egg
add
foo
bar
paper
title
```

Sample Output 0

```
YES
NO
YES
```

Explanation 0

Test Case 1:

- `s = "egg"`, `t = "add"`
- 'e' maps to 'a'.
- 'g' maps to 'd'.
- All mappings are unique and consistent. Output: **YES**.

Test Case 2:

- `s = "foo"`, `t = "bar"`
- First 'o' maps to 'a'.
- Second 'o' maps to 'r'.
- Since 'o' cannot map to two different characters ('a' and 'r'), this is invalid. Output: **NO**.

Test Case 3:

- `s = "paper"`, `t = "title"`
- 'p' -> 't', 'a' -> 'i', 'e' -> 'l', 'r' -> 'e'.
- This is a valid one-to-one mapping. Output: **YES**.

Krish's Situationships on Stack overflow

HackerRank

`Time Limit: 2 seconds`

`Memory Limit: 256MB`

The *tea* is boiling. Krish *bhaiya*'s love life is officially a stack overflow error. He is currently juggling n **different situationships** at the same time, and the drama is about to crash his entire system.

To prevent a total breakup apocalypse, he has decided to systematize his "reply game." He has to survive m **rounds** of texting. Each of the n situationships has sent him m **specific messages** that he needs to reply to. Each message has a "Drama Score" (an integer).

Krish has n situationships (numbered 1 to n). Each situationship i has a list of m distinct message "Drama Scores." Across all situationships, there are exactly $n \times m$ messages with distinct scores ranging from 0 to $n \cdot m - 1$.

Krish needs to establish a **fixed "Reply Order"** p (a permutation of situationships $1 \dots n$). He will cycle through this exact order m times (once for each round).

The Game Rules:

1. The "Current Drama Level" starts at -1.
2. In each step of the cycle, the current situationship p_i must pick one message from their list to "reply" with.
3. **Constraint:** The message they pick must have a Drama Score **strictly greater** than the "Current Drama Level."
4. Once played, that message's score becomes the *new* "Current Drama Level."
5. This continues until all n situationships have replied m times (emptying their lists).

If at any point a situationship is up next but **cannot** play a message higher than the current level, Krish gets blocked, and the game is lost.

Your Task: Does there exist a permutation p that allows Krish to survive all m rounds? If yes, output the permutation (space-separated). If no, output `-1` (Krish is cooked).

Input Format

- The first line contains an integer t — the number of test cases.
- The first line of each test case contains two integers n and m — the number of situationships and the number of messages each one has.
- The following n lines each contain m integers — the Drama Scores for each situationship.
- **Note:** All $n \cdot m$ scores are guaranteed to be distinct and in the range $[0, n \cdot m - 1]$.

Constraints

- $1 \le t \le 1000$
- $1 \le n, m \le 1000$
- $1 \le n \cdot m \le 10^6$
- The sum of $n \cdot m$ over all test cases won't exceed 10^6.

Output Format

For each test case:

- If a valid order exists, output n integers representing the permutation p.
- Otherwise, output `-1`.

Sample Input 0

```
4
2 3
0 4 2
1 5 3
1 1
0
2 2
1 2
0 3
4 1
1
2
0
3
```

Sample Output 0

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

Explanation 0

Test Case 1:

- $n = 2, m = 3$.
- Sorted cards: $0, 1, 2, 3, 4, 5$.
- Owners:
  - 0 -> Row 1
  - 1 -> Row 2
  - 2 -> Row 1
  - 3 -> Row 2

    - 4 -> Row 1
    - 5 -> Row 2
- The sequence of owners is `1, 2, 1, 2, 1, 2`. This repeats the pattern `1 2` perfectly.
- Output:**1 2**

Test Case 2:

- $n = 1, m = 1$. One card, one owner. Trivial YES.
- Output: **1**

Test Case 3:

- $n = 2, m = 2$.
- Sorted cards: $0, 1, 2, 3$.
- Owners:
    - 0 -> Row 2
    - 1 -> Row 1
    - 2 -> Row 1
    - 3 -> Row 2
- Sequence of owners: `2, 1, 1, 2`.
- For a valid cycle of length $n = 2$, the second half (`1, 2`) should match the first half (`2, 1`). It does not.
- Output: **-1**

Test Case 4:

- $n = 4, m = 1$.
- Sorted cards: $0, 1, 2, 3$.
- Owners:
    - 0 -> Row 3
    - 1 -> Row 1
    - 2 -> Row 2
    - 3 -> Row 4
- Sequence: `3, 1, 2, 4`. Since there is only `1` round, this valid permutation is the answer.
- Output: **3 1 2 4**

Lavanya's Party Cap

`Time Limit: 2 seconds`

`Memory Limit: 512MB`

The tea on Lavanya *di* is legendary. She isn't just "mad for parties"—she's a party *vortex*. Her social calendar is a blur, and she's left a trail of IOUs and unpaid bills across the college. She has successfully partied with seniors, juniors, professors, and (allegedly) a homeless guy she met at 3 AM.

Now, the bill is due. She's staring at a massive, terrifying list of N party expenses (decor, damages, "emotional support" snacks...).

But Lavanya doesn't *pay* bills. She *delegates* them.

She has successfully cornered D "sponsors"—unwitting victims she charmed or blackmailed into helping. She now needs to "distribute the opportunity" (i.e., dump her debt) onto them.

Being methodically lazy, she won't reorder her expense list. It's too much work. She'll just go down the line:

- The first sponsor (Victim 1) gets a contiguous block of expenses from the start (e.g., expenses 1-3).
- The second sponsor (Victim 2) gets the *next* contiguous block (e.g., expenses 4-6).
- ...and so on, until all N expenses are covered by her D sponsors.

She has one critical problem: if any single sponsor's bill is too high, they might actually realize they've been scammed and start a revolution. She needs to find the "sweet spot" to keep everyone just *calm* enough.

She needs you to find the **minimum possible value for the *maximum* bill** any single sponsor gets stuck with. If she can make this "maximum bill" (let's call it C) as small as possible, no single person will feel *too* betrayed, and she can live to party another day.

Input Format

- The first line contains a single integer T — the number of test cases.
- Each test case consists of two lines:
  - The first line contains two integers N (the number of expenses) and D (the number of sponsors).
  - The second line contains N space-separated integers, $e_1, e_2, \ldots, e_N$, representing the cost of each party expense.

Constraints

- $1 \leq T \leq 10$
- $1 \leq D \leq N \leq 10^5$

- $1 \leq \text{expense}_i \leq 10^6$
- The sum of N over all test cases will not exceed 10^6.
- **Note:** Her party bills are... astronomical. The total bill for a sponsor might overflow a 32-bit integer.

Output Format

- For each test case, output a single integer: the **minimum possible value for the largest bill** any sponsor has to pay.

Sample Input 0

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

Sample Output 0

```
18
9
```

Explanation 0

Test Case 1

- **Input:** $N = 5$ expenses, $D = 2$ sponsors.
- **Expenses:** `[7, 2, 5, 10, 8]`

We need to split the list into **2 contiguous subarrays** such that the largest sum is minimized. The optimal split is:

1. **Sponsor 1:** `[7, 2, 5]` $\rightarrow$ Sum = $7 + 2 + 5 = 14$
2. **Sponsor 2:** `[10, 8]` $\rightarrow$ Sum = $10 + 8 = 18$

The bills are 14 and 18. The maximum bill is 18. *(Note: If we tried splitting it as* `[7, 2]` *and* `[5, 10, 8]`*, the bills would be* 9 *and* 23*. Since* $23 > 18$*, that arrangement is worse.)*

Test Case 2

- **Input:** $N = 6$ expenses, $D = 3$ sponsors.
- **Expenses:** `[1, 2, 3, 4, 5, 6]`

We need to split the list into **3 contiguous subarrays**. The optimal split is:

1. **Sponsor 1:** `[1, 2, 3]` $\rightarrow$ Sum = 6
2. **Sponsor 2:** `[4, 5]` $\rightarrow$ Sum = 9
3. **Sponsor 3:** `[6]` $\rightarrow$ Sum = 6

The bills are 6, 9, and 6. The maximum bill is 9.

Ranjan's Ex-OR Dilemma

HackerRank

`Time Limit: 2 seconds`

`Memory Limit: 256MB`

The *tea* on Ranjan *bhaiya* is legendary. We all know about his "tragic" school breakup. But the real *hot tea* isn't that he broke up; it's that it took him **forever** to move on.

Why? Because Ranjan is the President of the "Lazy Geniuses" club.

Instead of just going to the gym or blocking her number like a normal person, he convinced himself that he could only "officially" move on if he solved a complex "Closure Equation" involving the memory of his ex (represented by the integer c).

He needs to find two new hobbies, a and b, that satisfy a perfectly balanced, yet needlessly complicated, mathematical condition. But here's the catch: **Ranjan is too lazy to actually do the math.** He would rather stay in bed.

He needs you to find the numbers for him so he can finally close this chapter and go back to sleep.

You are given a positive integer c ($1 \le c \le 10^7$), representing the "Memory Constant."

You need to find **any** two positive integers a and b such that they satisfy the "Closure Equation":

$$(a \oplus c) + (b \oplus c) = \operatorname{lcm}(a, c) + \operatorname{lcm}(b, c)$$

Where:

- $\oplus$ denotes the bitwise XOR operator.
- $\operatorname{lcm}(x, y)$ denotes the lowest common multiple of x and y.
- The values of a and b must satisfy $1 \le a, b \le 10^{17}$.

Note: It can be proven that a solution always exists under the given constraints.

Input Format

- The first line of input contains a single integer T — the number of test cases.
- Each test case consists of a single line containing one integer c.

Constraints

- $1 \le T \le 10^5$
- $1 \le c \le 10^7$

Output Format

- For each test case, output two space-separated integers a and b that satisfy the equation.
- If there are multiple valid pairs, you may output **any** of them.

Sample Input 0

```
3
1
2
7
```

Sample Output 0

```
88 71
80 62
1 35
```

Explanation 0

In the first test case, we are given $c = 1$. The sample output shows $a = 88$ and $b = 71$. Let's verify if this satisfies Ranjan's "Closure Equation."

The equation is:

$$(a \oplus c) + (b \oplus c) = \operatorname{lcm}(a, c) + \operatorname{lcm}(b, c)$$

Step 1: Calculate the Left Hand Side (LHS)

- $88 \oplus 1 = 89$
- $71 \oplus 1 = 70$
- **LHS** $= 89 + 70 = 159$

Step 2: Calculate the Right Hand Side (RHS)

- $\operatorname{lcm}(88, 1) = 88$
- $\operatorname{lcm}(71, 1) = 71$
- **RHS** $= 88 + 71 = 159$

Since $\mathrm{LHS} = \mathrm{RHS}$, the pair $(88, 71)$ is a valid solution.

Note: This problem allows multiple correct answers. For $c = 1$, other pairs like `1 35` or `2 3` are also valid. Your output is correct as long as it satisfies the equation.

Romanch and his "secret girlfriend"

`Time Limit: 1 second`

`Memory Limit: 256MB`

Our beloved president, Romanch bhaiya, is at a crossroads. The *hot tea* is, he (allegedly) has a secret girlfriend, and he's been consulting everyone on whether he should *finally* make the relationship public.

Being an engineer, he refuses to make a decision based on *mere feelings*. Instead, he has devised a complex, totally-not-overanalyzed test.

He has a list of n "topics of conversation" he needs to manage for the "Big Reveal." Each topic has a "chaos score" a_i. A low score is safe (e.g., "nice weather"), but a high score is *spicy* (e.g., "so, about my secret girlfriend...").

To plan the reveal, he must arrange these n topics into a sequence. To keep the conversation balanced and prevent it from exploding into drama, his plan requires a "pseudo-palindrome" structure:

- The **1st** topic he introduces must be compatible with the **n-th** (last) topic.
- The **2nd** topic must be compatible with the **$(n-1)$-th** topic.
- ...and so on, for all i from 1 to n.

Two topics x and y are "compatible" if the absolute difference of their chaos scores is no more than d (i.e., $|x-y| \leq d$).

Is it possible for Romanch to find *any* rearrangement of his n conversation topics such that for all i (from 1 to n), the i-th topic and the $(n+1-i)$-th topic are compatible?

If a low-drama arrangement exists, output 'Yes' (it's time to go public!). Otherwise, output 'No' (the secret stays safe, for now...).

Input Format

- The first line of input contains a single integer T, denoting the number of test cases.
- Each test case consists of two lines of input.
  - The first line contains two space-separated integers n and d — the length of the array (number of topics) and the maximum allowed difference.
  - The second line contains n space-separated integers $a_1, a_2, \ldots, a_n$ — the chaos scores of the topics.

Constraints

- $1 \leq T \leq 10^4$
- $1 \leq n \leq 10^5$

- $1 \leq a_i \leq 10^9$
- $0 \leq d \leq 10^9$
- The sum of n over all test cases won't exceed 10^5.

Output Format

- For each test case, output on a new line the answer: **'YES'** if a valid rearrangement exists, and **'NO'** otherwise.

Note: The output is **not case sensitive**. You can print `YES`, `yes`, `Yes`, `NO`, `no`, `nO`, etc. All will be accepted.

Sample Input 0

```
3
1 1
1
2 0
1 2
2 1000
1 2
```

Sample Output 0

```
YES
NO
YES
```

Explanation 0

Test Case 1:

- $n = 1, d = 1$, array `[1]`.
- With only 1 topic, there are no pairs to check. Since n is odd, the single element is valid on its own.
- Output: **YES**

Test Case 2:

- $n = 2, d = 0$, array `[1, 2]`.
- We must pair `(1, 2)` there are 2 possible ways `[1,2]` or `[2,1]` .
- in both the cases Difference: $|1 - 2| = 1$.
- Since $1 > d$ (which is 0), this pair is invalid.
- Output: **NO**

Test Case 3:

- $n = 2, d = 1000$, array `[1, 2]`.
- We must pair `(1, 2)` there are 2 possible ways `[1,2]` or `[2,1]`.

- in both the cases Difference: $|1 - 2| = 1$.
- Since $1 \leq 1000$, this pair is valid.
- Output: **YES**

Shreya-Paglu

HackerRank

`Time Limit: 1 second`

`Memory Limit: 512MB`

The *tea* is scalding. Our VP, Shreya *di*, is just trying to focus at her new job at Airtel. But "Kosuke," her most *dedicated* admirer from... *somewhere*... has somehow found out where she works and is trying to "match her energy" during her lunch break.

This has kicked off a silent, high-stakes "Aura War." The vibe of the break room starts at a neutral $x = 0$. They take turns, and **Shreya, who was just trying to eat her lunch, has to make the first move.**

On the i-th turn, the current player changes the "aura level" by $2 \cdot i - 1$ units.

- **Kosuke's Turn:** He's in his "nice guy" era. He tries to impress her by projecting a massive, cringey, **positive aura** (e.g., "wow, you work so hard," "can I get you a coffee?"). He adds $+(2 \cdot i - 1)$ to the aura.
- **Shreya's Turn:** She counters. Going full *'paglu-mode'*, she projects an overwhelming, "leave me alone, I am debugging in my head" **negative aura**. She adds $-(2 \cdot i - 1)$ to the aura.

The battle plays out like this:

1. **Shreya** (Turn 1): Radiates "do not perceive me" energy by -1. Aura is now $x = -1$.
2. **Kosuke** (Turn 2): Tries to "fix the vibe" with $+3$. Aura is now $x = 2$.
3. **Shreya** (Turn 3): Enters an unstoppable "flow state," projecting -5. Aura is now $x = -3$.
4. ...

Shreya has a final "tolerance limit" n for this nonsense. The game *instantly* stops the moment the aura level goes critical (i.e., the absolute value $|x|$ becomes greater than n).

- If $x > n$, Kosuke's positive aura is too overwhelming, and Shreya has to get up and leave.
- If $x < -n$, Shreya's "paglu" aura is too intense, and *he* gets intimidated and finally leaves.

Your task is to determine who makes the *final move* that pushes the aura level $|x|$ past n.

Input Format

- The only line of input contains a single integer n — Shreya's "tolerance limit."

Constraints

- $1 \leq n \leq 10^{18}$

Output Format

- Output a single line with the name of the person who made the final move.
- If Shreya makes the last move (her focus wins), output `Shreya`.

- If Kosuke makes the last move (he's just too much), output `Kosuke`.

Note: The output is **not case sensitive**. You can print `shreya`, `KOSUKE`, `Shreya`, etc. All will be accepted.

Sample Input 0

```
1
```

Sample Output 0

```
Koshuke
```

Explanation 0

- **Turn 1 (Shreya):** She moves first. The move size is $2(1) - 1 = 1$. She subtracts 1.
  - Aura level $x = 0 - 1 = -1$.
  - Condition check: $|x| = |-1| = 1$. This is not greater than $n = 1$. The game continues.
- **Turn 2 (Kosuke):** He moves second. The move size is $2(2) - 1 = 3$. He adds 3.
  - Aura level $x = -1 + 3 = 2$.
  - Condition check: $|x| = |2| = 2$. This is greater than $n = 1$. The game ends. The last move was made by **Kosuke**.

Sample Input 1

```
2
```

Sample Output 1

```
Shreya
```

Explanation 1

The tolerance limit is $n = 2$.

- **Turn 1 (Shreya):** Subtracts 1. $x = -1$. $|-1| \leq 2$. Continue.
- **Turn 2 (Kosuke):** Adds 3. $x = -1 + 3 = 2$. $|2| \leq 2$. Continue.
- **Turn 3 (Shreya):** Subtracts $2(3) - 1 = 5$. $x = 2 - 5 = -3$.
  - Condition check: $|-3| = 3$. This is greater than $n = 2$. The game ends. The last move was made by **Shreya**.

Vrinda & her cutus

HackerRank

`Time Limit: 1 second`

`Memory Limit: 256 mb`

After months of planing Namespace BPIT is again organising NS-Hacks in college campus, but Vrinda *di* is on a mission. She's completely ignoring the hackathon setup and is instead walking around the hall on a "cutu-squad" formation spree.

Her "tea" is that she believes the *only* valid team structure is a **"cutu squad,"** which, according to her very scientific, totally-not-made-up-on-the-spot definition, must have **exactly five participants**.

She's been heard saying, "You? You're a cutu. You're a cutu. You're *also* a cutu..."

She has successfully formed n complete "cutu squads." Before the organizers can *please* get her to sit down, she needs a final count of just how many participants she has blessed with the "cutu" designation.

Given that she has formed n cutu squads, how many "cutu" participants are there in total?

Input Format

- The only line of input contains a single integer n, denoting the number of "cutu squads" that Vrinda has formed.

Constraints

- $1 \leq n \leq 1000$

Output Format

- Output a single integer: the total number of "cutu" participants.

Sample Input 0

```
5
```

Sample Output 0

```
25
```

Explanation 0

Vrinda formed 5 squads. Each squad = 5 members. Total = 5 × 5 = 25.

Sample Input 1

```
0
```

Sample Output 1

```
0
```

Explanation 1

No squads formed → total cutus = 0 × 5 = 0.