Problem S1: Baby Hop, Giant Hop

Problem Description

Samantha the Frog is hopping between lily pads arranged in a straight line, evenly spaced. There are an infinite number of lily pads. The lily pads are numbered, in order, using integers. For every integer, there is also a lily pad.

Samantha starts on a lily pad numbered A and would like to hop onto a lily pad numbered B. She can take a giant hop of length K, or a baby hop of length 1. Each hop can be either forwards or backwards.

Samantha would like to know the fewest number of hops needed to get from A to B. But sometimes, she would like to know the second fewest number of hops needed.

Input Specification

The first line of input contains a single integer, A, the starting lily pad ($-10^{18} \leq A \leq 10^{18}$).

The second line of input contains a single integer, B, the ending lily pad ($-10^{18} \leq B \leq 10^{18}$).

The third line of input contains a single integer, K, the distance of a giant hop ($2 \leq K \leq 10^{18}$).

The fourth line of input contains the integer, T, which is either 1 or 2, indicating if the fewest (when $T = 1$) or second fewest (when $T = 2$) number of steps should be found.

The following table shows how the 15 available marks are distributed:

Marks Awarded	**Bounds on A, B**	**Bounds on K**	**Bounds on T**	**Additional Restrictions**
5 marks	$0 \leq A, B \leq 10$	$K = 2$	$T = 1$	Only hops in the positive direction needed
6 marks	$-10^{18} \leq A, B \leq 10^{18}$	$2 \leq K \leq 10^{18}$	$T = 1$	None
2 marks	$0 \leq A, B \leq 100$	$2 \leq K \leq 4$	$T = 1$ or $T = 2$	None
2 marks	$-10^{18} \leq A, B \leq 10^{18}$	$2 \leq K \leq 10^{18}$	$T = 1$ or $T = 2$	None

Note that for full marks, solutions will need to handle 64-bit integers. For example:

- in C/C++, the type `long long` should be used;
- in Java, the type `long` should be used.

Output Specification

On a single line, output:

- the fewest number of hops, if $T = 1$
- the second fewest number of hops, if $T = 2$

required to move from lily pad A to lily pad B.

La version française figure à la suite de la version anglaise.

Sample Input 1

```
0
10
3
1
```

Output for Sample Input 1

```
4
```

Explanation of Output for Sample Input 1

Samantha hops to lily pads labeled 3, 6, and 9, with three giant hops, and then hops to the lily pad labeled 10 with one baby hop.

Sample Input 2

```
0
11
4
1
```

Output for Sample Input 2

```
4
```

Explanation of Output for Sample Input 2

Samantha hops to lily pads labeled 4, 8, and 12, with three giant hops, and then hops to the lily pad labeled 11 with one (backwards) baby hop.

Sample Input 3

```
0
11
4
2
```

Output for Sample Input 3

```
5
```

Explanation of Output for Sample Input 3

The fewest number of hops needed (4) was found in Sample 2. In this input, the second fewest number of steps is to be found, since $T = 2$. Samantha hops to lily pads labeled 4 and 8 with two giant hops, and then hops to the lily pad labeled 11 with three baby hops.

La version française figure à la suite de la version anglaise.

Sample Input 4

```
0
0
3
1
```

Output for Sample Input 4

```
0
```

La version française figure à la suite de la version anglaise.

Problème S1: Mini saut / Méga saut

Énoncé du problème

Samantha la grenouille saute parmi des nénufars alignés et régulièrement espacés. Il y a une infinité de nénufars, numérotés par des entiers consécutifs, de sorte qu'à chaque entier correspond un nénufar.

Samantha commence sur le nénufar numéroté A et elle veut sauter jusqu'au nénufar numéroté B. Elle peut effectuer un méga saut de longueur K ou un mini saut de longueur 1. Chaque saut peut être en avant ou en arrière.

Samantha veut connaître le nombre minimal de sauts pour atteindre le nénufar B depuis A. Parfois, elle veut aussi connaître le deuxième plus petit nombre de sauts nécessaires.

Précisions par rapport aux données d'entrée

La première ligne des données d'entrée contient un seul entier, A, le nénufar de départ ($-10^{18} \leq A \leq 10^{18}$).

La deuxième ligne des données d'entrée contient un seul entier, B, le nénufar final ($-10^{18} \leq B \leq 10^{18}$).

La troisième ligne des données d'entrée contient un seul entier, K, la longueur d'un méga saut ($2 \leq K \leq 10^{18}$).

La quatrième ligne des données d'entrée contient un seul entier, T, soit 1, soit 2, indiquant s'il faut trouver le nombre minimal de sauts (pour $T = 1$) ou le deuxième plus petit nombre de sauts (pour $T = 2$).

Le tableau suivant indique la manière dont les 15 points disponibles sont répartis :

Points attribués	**Bornes sur A et B**	**Bornes sur K**	**Bornes sur T**	**Restrictions supplémentaires**
5 points	$0 \leq A, B \leq 10$	$K = 2$	$T = 1$	Sauts uniquement dans la direction positive
6 points	$-10^{18} \leq A, B \leq 10^{18}$	$2 \leq K \leq 10^{18}$	$T = 1$	Aucune
2 points	$0 \leq A, B \leq 100$	$2 \leq K \leq 4$	$T = 1$ or $T = 2$	Aucune
2 points	$-10^{18} \leq A, B \leq 10^{18}$	$2 \leq K \leq 10^{18}$	$T = 1$ or $T = 2$	Aucune

Pour obtenir le maximum de points, une solution devra pouvoir traiter des entiers de 64 bits. Par exemple,

- en C/C++, il faudrait utiliser une chaine de type `long long`;
- en Java, il faudrait utiliser une chaine de type `long`.

The English version appears before the French version.

Précisions par rapport aux données de sortie
Sur une seule ligne, afficher :

- le nombre minimal de sauts, si $T = 1$
- le deuxième plus petit nombre de sauts, si $T = 2$

nécessaires pour se déplacer du nénufar A jusqu'au nénufar B.

Données d'entrée du 1er exemple
```
0
10
3
1
```

Données de sortie du 1er exemple
```
4
```

Justification des données de sortie du 1er exemple
Samantha saute sur les nénufars numérotés 3, 6 et 9 avec trois méga sauts, puis elle saute sur le nénufar numéroté 10 avec un mini saut.

Données d'entrée du 2e exemple
```
0
11
4
1
```

Données de sortie du 2e exemple
```
4
```

Justification des données de sortie du 2e exemple
Samantha saute sur les nénufars numérotés 4, 8 et 12 avec trois méga sauts, puis elle saute sur le nénufar numéroté 11 avec un mini saut (en arrière).

Données d'entrée du 3e exemple
```
0
11
4
2
```

Données de sortie du 3e exemple
```
5
```

The English version appears before the French version.

Justification des données de sortie du 3e exemple

Le nombre minimal de sauts (4) a été trouvé dans l'exemple 2. Avec ces données d'entrée, il faut trouver le deuxième plus petit nombre de sauts, puisque $T = 2$. Samantha saute sur les nénufars numérotés 4 et 8 avec deux méga sauts, puis elle saute jusqu'au nénufar numéroté 11 avec trois mini sauts.

Données d'entrée du 4e exemple

```
0
0
3
1
```

Données de sortie du 4e exemple

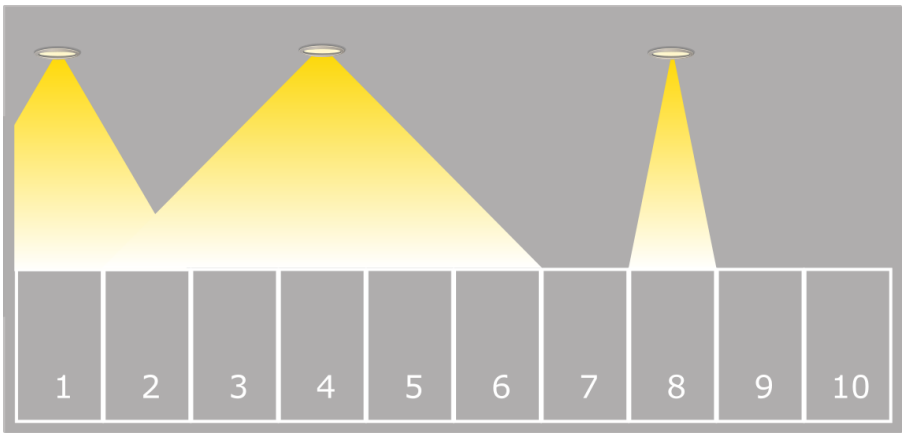
```
0
```

The English version appears before the French version.

Problem J5/S2: Beams of Light

Problem Description

Along one wall of a parking garage there are identical parking spots numbered from 1 to N. A collection of lights illuminates the parking garage. Each light shines on some number of adjacent parking spots.



You will be questioned about the parking spots. For each parking spot you are questioned about, your job is to determine whether or not it is illuminated by at least one light.

Input Specification

The first line of input contains a positive integer, N, representing the number of parking spots. The second line contains a non-negative integer, L, representing the number of lights. The third line contains a positive integer, Q, representing the number of parking spots you will be questioned about.

The next L lines provide information about the L lights. Line i will contain two integers, P_i and S_i, separated by a single space. The first integer, $1 \leq P_i \leq N$, represents the number of the parking spot above which a light is hung. The second integer, $0 \leq S_i \leq N$, represents the spread of the light's beam. Light i shines on the parking spot that is directly below it. It also shines on the S_i parking spots located on either side, unless there are fewer than S_i spots on a side, in which case all the spots on that side will be illuminated. There could be more than one light directly above a parking spot.

The next Q lines of input each contain a positive integer between 1 and N inclusive, representing the number of the parking spot you are questioned about.

La version française figure à la suite de la version anglaise.

The following table shows how the 15 available marks are distributed:

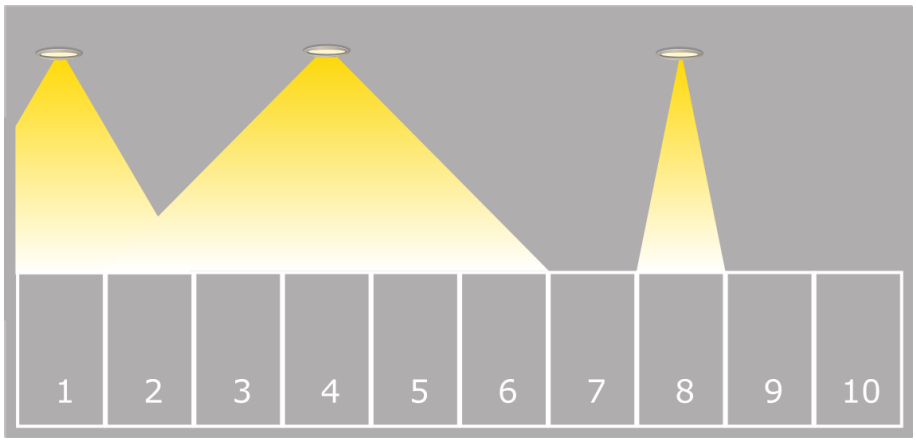
Marks	Number of Spots	Number of Lights	Number of Questions
1	$N \leq 50$	$L \leq 1$	$Q \leq 50$
2	$N \leq 50$	$L \leq 50$	$Q \leq 50$
3	$N \leq 50$	$L \leq 500\,000$	$Q \leq 500\,000$
9	$N \leq 500\,000$	$L \leq 500\,000$	$Q \leq 500\,000$

Output Specification
There will be one line of output for each of the Q parking spots you are questioned about.

On each of these lines, output `Y` if the corresponding parking spot is illuminated by at least one light, or `N` if the corresponding parking spot is not illuminated by any light.

Sample Input

```
10
3
4
8 0
1 1
4 2
4
10
7
1
```



Output for Sample Input

```
Y
N
N
Y
```

Explanation of Output for Sample Input
The input describes the picture of the parking garage shown above.

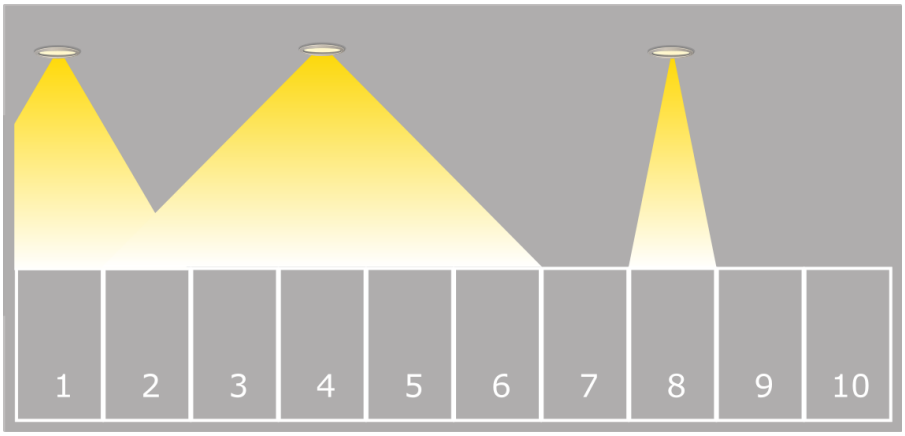
Parking spots 4 and 1 are illuminated by at least one light.

Parking spots 10 and 7 are not illuminated by any light.

La version française figure à la suite de la version anglaise.

Problème J5/S2: Faisceaux lumineux

Énoncé du problème
Le long d'un mur du parking, il y a des places de stationnement identiques, numérotées de 1 à N. Une série de lumières éclaire le parking. Chaque lumière éclaire un certain nombre de places de stationnement adjacentes.



Vous devrez répondre à des questions concernant les places de stationnement. Pour chaque place mentionnée dans une question, votre tâche consiste à déterminer si elle est éclairée ou non par au moins une lumière.

Précisions par rapport aux données d'entrée
La première ligne de l'entrée contient un entier strictement positif N représentant le nombre de places de stationnement. La deuxième ligne contient un entier non-négatif L représentant le nombre de lumières. La troisième ligne contient un entier strictement positif Q représentant le nombre de places de stationnement sur lesquelles vous devez répondre.

Les L lignes suivantes décrivent les L lumières. La ligne i contient deux entiers, P_i et S_i, séparés par un seul espace. Le premier entier, avec $1 \leq P_i \leq N$, représente le numéro de la place de stationnement au-dessus de laquelle la lumière est installée. Le second entier, avec $0 \leq S_i \leq N$, représente la portée du faisceau lumineux. La lumière i éclaire la place de stationnement située directement en dessous. Elle éclaire également les S_i places de stationnement situées de chaque côté. S'il y a moins de S_i places d'un côté, alors toutes les places de ce côté seront éclairées. Il peut y avoir plusieurs lumières directement au-dessus d'une place de stationnement.

Chacune des Q lignes suivantes contient un entier strictement positif, compris entre 1 et N inclus, représentant le numéro de la place de stationnement pour laquelle vous devez répondre la question.

The English version appears before the French version.

Le tableau suivant indique la manière dont les 15 points disponibles sont répartis :

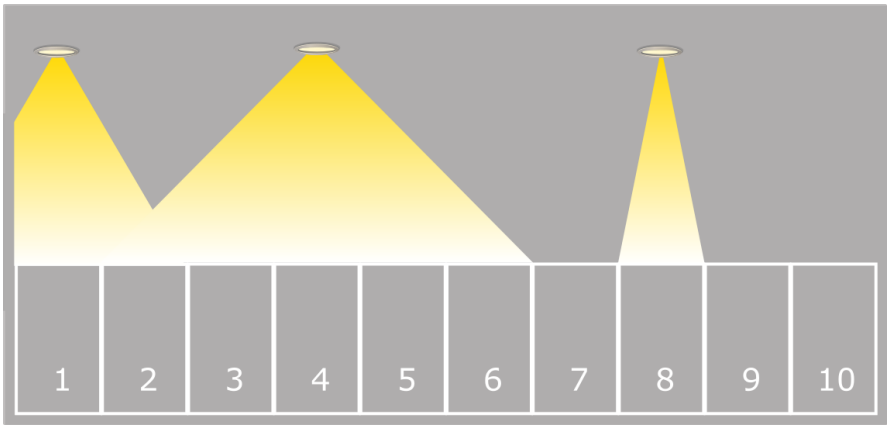
Points	Nombre de places	Nombre de lumières	Nombre de questions
1	$N \leq 50$	$L \leq 1$	$Q \leq 50$
2	$N \leq 50$	$L \leq 50$	$Q \leq 50$
3	$N \leq 50$	$L \leq 500\,000$	$Q \leq 500\,000$
9	$N \leq 500\,000$	$L \leq 500\,000$	$Q \leq 500\,000$

Précisions par rapport aux données de sortie
Les données de sortie devraient contenir une ligne pour chacune des Q places de stationnement indiquées dans les données d'entrée.

Sur chacune de ces lignes, les données de sortie devraient afficher `Y` si la place de stationnement correspondante est éclairée par au moins une lumière. Si la place de stationnement correspondante n'est éclairée par aucune lumière, les données de sortie devraient afficher `N`.

Exemple de données entrée

```
10
3
4
8 0
1 1
4 2
4
10
7
1
```



Exemple de données de sortie

```
Y
N
N
Y
```

Justification des données de sortie
Les données d'entrée décrivent le parking illustré ci-dessus.

Les places de stationnement 4 et 1 sont éclairées par au moins une lumière.

Les places de stationnement 10 et 7 ne sont éclairées par aucune des lumières.

The English version appears before the French version.

Problem S3: Common Card Choice

Problem Description

There are N cards with positive integers written on them. Alice takes some of the cards, then Bob takes some of the remaining cards. Both need to take at least one card; Alice is not allowed to take all of the cards. Then Alice and Bob compute the sum of numbers on the cards that they have taken. If these sums have a common divisor greater than one, Alice and Bob get a bag of sweets; otherwise they don't. Given the cards on the table, determine if it is possible for Alice and Bob to get the sweets, and if so, how.

To make it even more challenging, in some cases, neither Alice nor Bob knows any of the values on the cards.

Note that an integer d is a *divisor* of an integer q if there is no remainder when q is divided by d. An integer z is a *common divisor* of integers x and y if z is a divisor of both x and y.

Input Specification

The first line of input contains N ($2 \leq N \leq 10^5$).

The second line contains N integers, c_i ($1 \leq c_i \leq 10^{12}$).

The following table shows how the 15 available marks are distributed:

Marks Awarded	**Bounds on N**	**Bounds on c_i**
2 marks	$2 \leq N \leq 3$	$1 \leq c_i \leq 100$
1 marks	$2 \leq N \leq 10$	$1 \leq c_i \leq 100$
1 marks	$2 \leq N \leq 10^5$	$1 \leq c_i \leq 2$
2 marks	$2 \leq N \leq 10^5$	$1 \leq c_i \leq 10^5$
2 marks	$2 \leq N \leq 10^5$	$1 \leq c_i \leq 10^{12}$
7 marks	$N = 10^5$	$c_i = -1$ and see Note below

Note: For the last subtask, you will be given N, but all N card values will be labelled with `-1`, indicating that those card values are unknown to you. In this last subtask, there will always be a selection of cards that Alice and Bob can pick to obtain a bag of sweets.

Output Specification

On the first line of output, print `YES` if it is possible to get the sweets, otherwise, print `NO`.

If the answer is `YES`, on the second line of output, produce A, the number of cards that Alice should take, followed by one space, followed by B, the number of cards that Bob should take. On the third line of output, print out A space-separated numbers corresponding to the indices of the cards that Alice should take. On the fourth line of output, print out B space-separated numbers corresponding to the indices of the cards that Bob should take. If there are several ways to select cards, print any of them.

If the input is for the last subtask, output an integer $K \leq 100$ on the first line, followed by K possible combinations of output as specified for all other subtasks. That is, each combination

La version française figure à la suite de la version anglaise.

will be the same sequence of three lines as stated above: one line containing A and B; one line containing A indices of cards; and one line containing B indices of cards. All guesses of indices of cards must be disjoint: that is, no card index can appear in more than one guess. Your output will be judged correct if any of the combinations would result in Alice and Bob obtaining a bag of sweets. Note that you will not receive any feedback between guesses.

Sample Input 1

```
7
19 7 11 31 99 13 17
```

Output for Sample Input 1

```
YES
3 3
2 3 7
4 6 1
```

Explanation of Output for Sample Input 1

The sum of Alice's cards is $7 + 11 + 17 = 35$ and the sum of Bob's cards is $31 + 13 + 19 = 63$, and both 35 and 63 are divisible by 7.

Sample Input 2

```
3
3 11 17
```

Output for Sample Input 2

```
NO
```

Explanation of Output for Sample Input 2

Notice that 3, 11, and 17 have no common divisors, and all possible sums of two of these numbers (14, 20, 28) do not have a common factor with the remaining third number.

La version française figure à la suite de la version anglaise.

Sample Input 3

```
100000
-1 -1 -1 ...
```

(The input has been truncated for space. There are a total of 10^5 `-1`s on the second line.)

Sample Output 3

```
2
1 2
1
3 5
3 2
100 101 102
201 200
```

Explanation of Output for Sample Input 3

There are a total of two guesses. The first guess consists of indices $\{1\}$ and $\{3, 5\}$. The second guess consists of indices $\{100, 101, 102\}$ and $\{201, 200\}$.

Note that all sets of indices are disjoint.

Although the output is formatted correctly, the output will receive **Wrong answer** because the two guesses are not enough for Alice and Bob to obtain a bag of sweets.

La version française figure à la suite de la version anglaise.

Problème S3: Choix de carte commune

Énoncé du problème

Il y a N cartes avec des entiers strictement positifs écrits dessus. Alice prend quelques cartes, puis Bob en prend quelques-unes parmi celles qui restent. Chacun doit prendre au moins une carte ; Alice ne peut pas prendre toutes les cartes. Ensuite, Alice et Bob calculent la somme des nombres sur les cartes qu'ils ont prises. Si ces sommes ont un diviseur commun supérieur à 1, Alice et Bob reçoivent un sac de bonbons ; sinon, ils n'en reçoivent pas. Si les cartes sur la table sont données, déterminez s'il est possible pour Alice et Bob de recevoir les bonbons et, le cas échéant, indiquez comment.

Pour augmenter le défi, dans certains cas, ni Alice ni Bob ne connaissent les valeurs inscrites sur les cartes.

Note : Un entier d est un *diviseur* de l'entier q s'il n'y a pas de reste lorsque q est divisé par d. Un entier z est un *diviseur commun* des entiers x et y si z est à la fois un diviseur de x et y.

Précisions par rapport aux données d'entrée

La première ligne des données d'entrée contient N ($2 \leq N \leq 10^5$).

La deuxième ligne des données d'entrée contient N entiers, c_i ($1 \leq c_i \leq 10^{12}$).

Le tableau suivant indique la manière dont les 15 points disponibles sont répartis :

Points attribués	**Bornes sur N**	**Bornes sur c_i**
2 points	$2 \leq N \leq 3$	$1 \leq c_i \leq 100$
1 point	$2 \leq N \leq 10$	$1 \leq c_i \leq 100$
1 point	$2 \leq N \leq 10^5$	$1 \leq c_i \leq 2$
2 points	$2 \leq N \leq 10^5$	$1 \leq c_i \leq 10^5$
2 points	$2 \leq N \leq 10^5$	$1 \leq c_i \leq 10^{12}$
7 points	$N = 10^5$	$c_i = -1$ et voir la note ci-dessous

Note : Pour la dernière sous-tâche, la valeur de N vous est donnée, mais les valeurs des N cartes seront toutes marquées `-1`, ce qui indique qu'elles sont inconnues. Dans cette sous-tâche, il existe toujours une sélection de cartes permettant à Alice et Bob d'obtenir un sac de bonbons.

Précisions par rapport aux données de sortie

Sur la première ligne de sortie, afficher `YES` s'il est possible d'obtenir les bonbons, sinon, afficher `NO`.

Si la réponse est `YES`, sur la deuxième ligne de sortie, afficher A, le nombre de cartes qu'Alice doit prendre, suivi d'un espace, puis B, le nombre de cartes que Bob doit prendre. Sur la troisième ligne de sortie, afficher les indices des A cartes qu'Alice doit prendre, séparés par un espace. Sur la quatrième ligne de sortie, afficher les indices des B cartes que Bob doit

The English version appears before the French version.

prendre, séparés par un espace. S'il existe plusieurs manières de choisir les cartes, en afficher une quelconque.

Si les données d'entrée correspondent à la dernière sous-tâche, afficher sur la première ligne un entier $K \leq 100$. Ensuite, afficher K combinaisons de données de sortie, telles que spécifiées pour toutes les autres sous-tâches. Autrement dit, chaque combinaison est constituée de la même séquence de trois lignes que celle décrite ci-dessus, à savoir : une ligne contenant A et B, une ligne contenant A indices de cartes, et une ligne contenant B indices de cartes. Toutes les sélections d'indices de cartes doivent être disjointes, c'est-à-dire qu'aucun indice de carte n'apparaît dans plus d'une sélection. Les données de sortie sont jugées correctes si au moins une des combinaisons permet à Alice et Bob d'obtenir un sac de bonbons. Notez qu'aucune rétroaction ne sera fournie entre les sélections.

Données d'entrée du 1er exemple

```
7
19 7 11 31 99 13 17
```

Données de sortie du 1er exemple

```
YES
3 3
2 3 7
4 6 1
```

Justification des données de sortie du 1er exemple

La somme des cartes d'Alice est $7+11+17 = 35$ et celle des cartes de Bob est $31+13+19 = 63$, les deux sommes sont divisibles par 7.

Données d'entrée du 2e exemple

```
3
3 11 17
```

Données de sortie du 2e exemple

```
NO
```

Justification des données de sortie du 2e exemple

Notez que 3, 11 et 17 n'ont aucun diviseur commun, et qu'aucune somme de deux de ces nombres (14, 20, 28) n'a de facteur commun avec le nombre restant.

The English version appears before the French version.

Données d'entrée du 3e exemple

```
100000
-1 -1 -1 ...
```

(Les données d'entrée étaient tronquées pour des raisons d'espace. Il y a un total de 10^5 `-1` sur la deuxième ligne.)

Données de sortie du 3e exemple

```
2
1 2
1
3 5
3 2
100 101 102
201 200
```

Justification des données de sortie du 3e exemple

Il y a un total de deux sélections. La première sélection est constituée des indices $\{1\}$ et $\{3, 5\}$. La deuxième sélection est constituée des indices $\{100, 101, 102\}$ et $\{201, 200\}$.

Notez que tous les ensembles d'indices sont disjoints.

Bien que le formatage soit correct, la réponse sera **Mauvaise réponse** (**Wrong Answer**) car les deux tentatives ne suffisent pas à Alice et Bob pour obtenir un sac de bonbons.

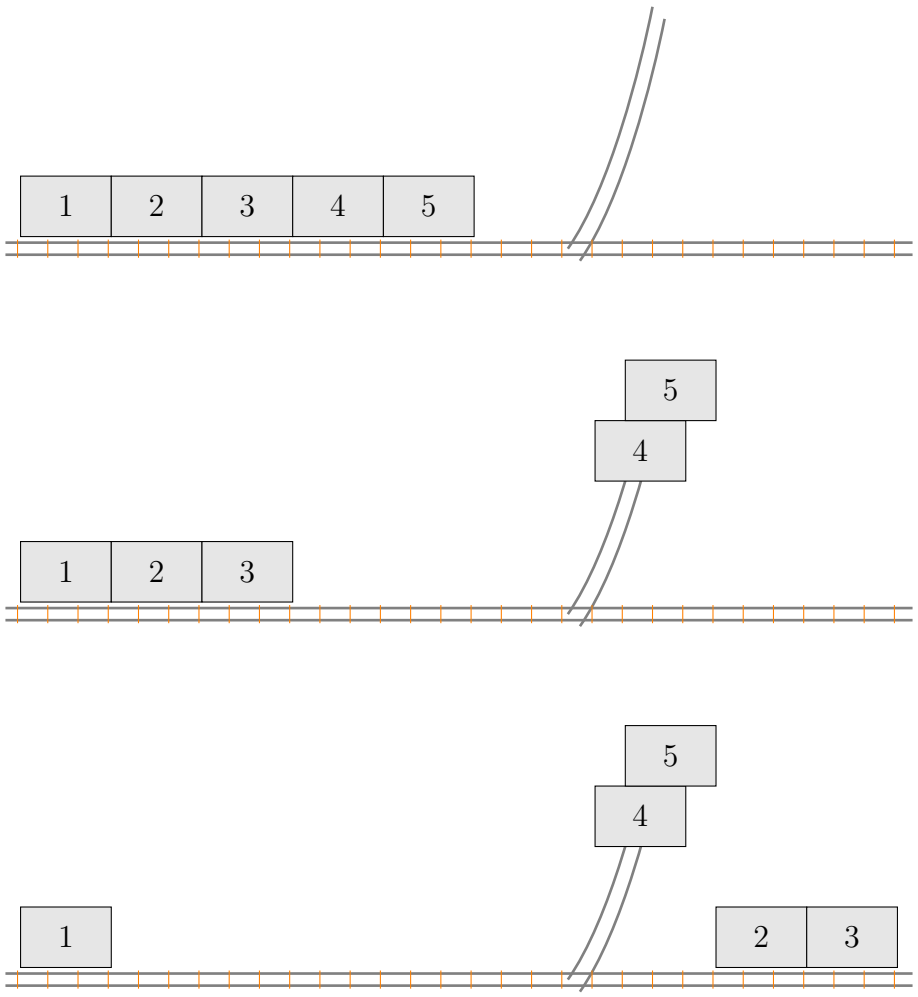
The English version appears before the French version.

Problem S4: Minecarts

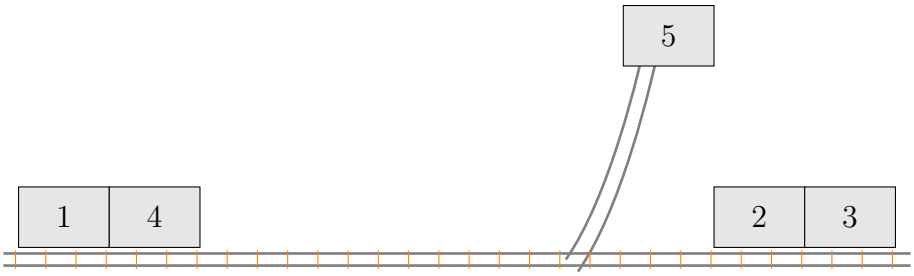
Problem Description

Steve has N minecarts labelled from 1 to N lined up on a track in a row from left to right. Initially, there are a_i gems in the i-th minecart.

Steve wants the minecarts to be arranged in a row such that the number of gems in each minecart is non-decreasing from left to right. To do this, he plans to build a side track to the right of all of the minecarts that splits off from the main track. Steve can move minecarts that are on the left of the side track into the side track, allowing other minecarts on its left to move past it on the main track. Minecarts moved into the side track can be moved back into the main track one at a time: a minecart on the side track will move back to the left of the side track and to the right of all other minecarts which are on the left of the side track. The last minecart moved into the side track must be the first minecart to be moved back out: that is, the side track follows a last-in, first-out method. The side track can be used any number of times. Finally, once a minecart is moved to the right of the side track, it can no longer be moved to the left. Below is an example sequence of moves that can be made with 5 minecarts beginning from the initial position:



La version française figure à la suite de la version anglaise.



Steve has K spare gems, and he can distribute any number of them into the minecarts that are **empty**. Steve has yet to build the side track, and he wants to limit the side track capacity to save work. A side track with a specific capacity can only store up to that many minecarts. For example, if the side track had a capacity of 1, we would not be able to make the moves in the diagrams above, which would require a capacity of at least 2. Given that Steve places his spare gems into the minecarts optimally, what is the minimum capacity of the side track that needs to be built?

Input Specification

The first line of input will consist of two space-separated integers N and K ($1 \leq N \leq 3 \times 10^5$, $0 \leq K \leq 10^{12}$).

The next line contains N integers $a_1, \ldots, a_n$ ($0 \leq a_i \leq 10^6$), where the i-th integer indicates the number of gems in the i-th minecart.

The following table shows how the available 15 marks are distributed:

Marks Awarded	**Bounds on N**	**Bounds on K**
2 marks	$1 \leq N \leq 5\,000$	$K = 0$
2 marks	$1 \leq N \leq 5\,000$	$K = 10^{12}$
2 marks	$1 \leq N \leq 5\,000$	$0 \leq K \leq 10^{12}$
3 marks	$1 \leq N \leq 3 \times 10^5$	$K = 0$
3 marks	$1 \leq N \leq 3 \times 10^5$	$K = 10^{12}$
3 marks	$1 \leq N \leq 3 \times 10^5$	$0 \leq K \leq 10^{12}$

Output Specification

Output a single integer, the minimum capacity of the side track that needs to be built given that Steve distributes up to K gems among the empty minecarts optimally.

Sample Input 1

```
4 14
5 0 4 0
```

La version française figure à la suite de la version anglaise.

Output for Sample Input 1

```
1
```

Explanation of Output for Sample Input 1

One optimal distribution is to put 6 of the spare gems into minecart 2 and 7 of the spare gems into minecart 4. Note that this distribution does not use all of the spare gems.

We can then build a side track of capacity 1. We can then arrange the minecarts in non-decreasing order as follows:

1. Move minecart 4 past the side track
2. Move minecart 3 into the side track
3. Move minecart 2 past the side track
4. Move minecart 1 past the side track
5. Move minecart 3 out of the side track
6. Move minecart 3 past the side track

Sample Input 2

```
4 8
5 0 4 0
```

Output for Sample Input 2

```
2
```

Explanation of Output for Sample Input 2

One optimal distribution is to put all 8 spare gems into minecart 4.

Then we can build a side track of capacity 2. We can then arrange the minecarts in non-decreasing order as follows:

1. Move minecart 4 past the side track
2. Move minecart 3 into the side track
3. Move minecart 2 into the side track
4. Move minecart 1 past the side track
5. Move minecart 2 out of the side track
6. Move minecart 3 out of the side track
7. Move minecart 3 past the side track
8. Move minecart 2 past the side track

La version française figure à la suite de la version anglaise.

Sample Input 3

```
4 123456789
40 30 20 10
```

Output for Sample Input 3

```
3
```

Explanation of Output for Sample Input 3

Since there are no empty minecarts, there is only one possible distribution of spare gems: no spare gems are used at all.

Then we can build a side track of capacity 3. We can then arrange the minecarts in non-decreasing order as follows:

1. Move minecart 4 into the side track
2. Move minecart 3 into the side track
3. Move minecart 2 into the side track
4. Move minecart 1 past the side track
5. Move minecart 2 out of the side track and then past the side track
6. Move minecart 3 out of the side track and then past the side track
7. Move minecart 4 out of the side track and then past the side track

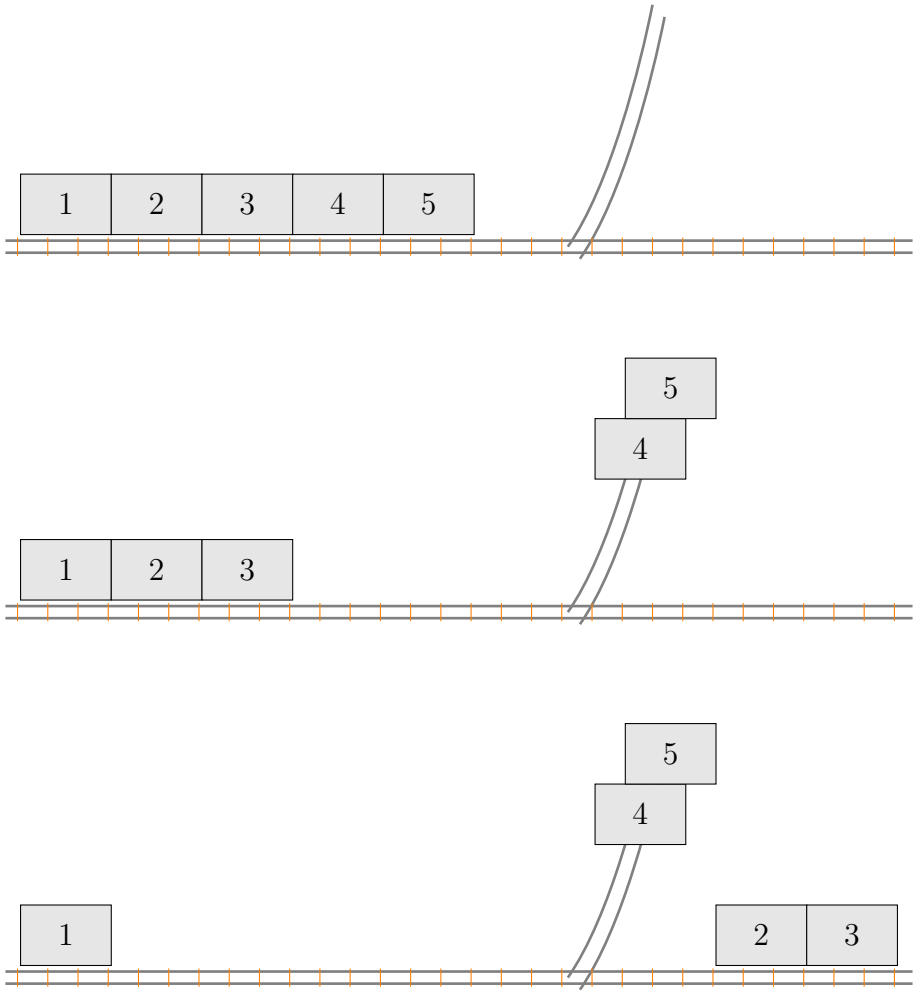
La version française figure à la suite de la version anglaise.

Problème S4: Wagonnets

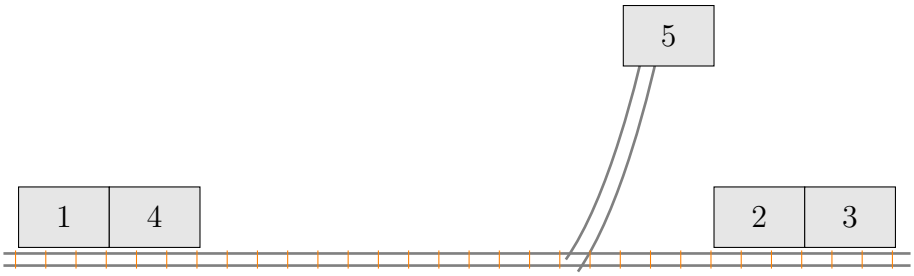
Énoncé du problème

Steve a N wagonnets, numérotés de 1 à N, alignés de gauche à droite sur un rail. Initialement, il y a a_i pierres précieuses dans le i-ième wagonnet.

Steve veut que les wagonnets soient disposés sur le rail de sorte que le nombre de pierres précieuses dans chaque wagonnet soit non décroissant de gauche à droite. Pour ce faire, il prévoit de construire une voie d'évitement à droite de tous les wagonnets. Steve peut déplacer les wagonnets situés à gauche de la voie d'évitement dans celle-ci. Cela permet aux autres wagonnets encore à gauche de la voie de passer sur le rail principal. Les wagonnets sur la voie d'évitement peuvent être remis un par un sur le rail principal : chacun sera replacé à gauche de la voie d'évitement et à droite de tous les autres wagonnets déjà présents à gauche de cette voie. Le dernier wagonnet déplacé dans la voie d'évitement doit être le premier à s'en retirer : la voie d'évitement suit donc le principe du dernier entré, premier sorti. Elle peut être utilisée autant de fois que nécessaire. Enfin, un wagonnet déplacé à droite de la voie d'évitement ne peut plus revenir à gauche de cette voie. Les figures ci-dessous illustrent des séquences possibles de mouvements avec 5 wagonnets à partir de la position initiale :



The English version appears before the French version.



Steve dispose de K pierres précieuses supplémentaires et peut en distribuer autant qu'il le souhaite parmi les wagonnets **vides**. Steve n'a pas encore construit la voie d'évitement et souhaite limiter sa capacité afin de réduire le travail. Une voie d'évitement peut accueillir au maximum un nombre de wagonnets égal à sa capacité. Par exemple, si la voie d'évitement avait une capacité de 1, il ne serait pas possible d'effectuer les déplacements illustrés ci-dessus, car ils nécessiteraient une capacité d'au moins 2. En supposant que Steve place ses pierres précieuses de manière optimale dans les wagonnets, quelle sera la capacité minimale de la voie d'évitement à construire ?

Précisions par rapport aux données d'entrée

La première ligne des données d'entrée contient deux entiers N et K séparés par un espace ($1 \leq N \leq 3 \times 10^5$, $0 \leq K \leq 10^{12}$).

La ligne suivante contient N entiers $a_1, \ldots, a_n$ ($0 \leq a_i \leq 10^6$), où le i-ième entier indique le nombre de pierres précieuses dans le i-ième wagonnet.

Le tableau suivant indique la manière dont les 15 points disponibles sont répartis :

Points attribués	**Bornes sur N**	**Bornes sur K**
2 points	$1 \leq N \leq 5\,000$	$K = 0$
2 points	$1 \leq N \leq 5\,000$	$K = 10^{12}$
2 points	$1 \leq N \leq 5\,000$	$0 \leq K \leq 10^{12}$
3 points	$1 \leq N \leq 3 \times 10^5$	$K = 0$
3 points	$1 \leq N \leq 3 \times 10^5$	$K = 10^{12}$
3 points	$1 \leq N \leq 3 \times 10^5$	$0 \leq K \leq 10^{12}$

Précisions par rapport aux données de sortie

Afficher un seul entier, représentant la capacité minimale de la voie d'évitement à construire, étant donné que Steve distribue de manière optimale au plus K pierres précieuses parmi les wagonnets vides.

The English version appears before the French version.

Données d'entrée du 1er exemple

```
4 14
5 0 4 0
```

Données de sortie du 1er exemple

```
1
```

Justification des données de sortie du 1er exemple

Une distribution optimale consiste à placer 6 des pierres précieuses supplémentaires dans le wagonnet 2 et 7 autres dans le wagonnet 4. Notez que cette distribution n'utilise pas toutes les pierres supplémentaires.

Ensuite, nous pouvons construire une voie d'évitement de capacité 1. Les wagonnets peuvent s'aligner dans un ordre non décroissant de la façon suivante :

1. Déplacer le wagonnet 4 de l'autre côté de la voie d'évitement
2. Déplacer le wagonnet 3 dans la voie d'évitement
3. Déplacer le wagonnet 2 de l'autre côté de la voie d'évitement
4. Déplacer le wagonnet 1 de l'autre côté de la voie d'évitement
5. Déplacer le wagonnet 3 hors de la voie d'évitement
6. Déplacer le wagonnet 3 de l'autre côté de la voie d'évitement

Données d'entrée du 2e exemple

```
4 8
5 0 4 0
```

Données de sortie du 2e exemple

```
2
```

Justification des données de sortie du 2e exemple

Une distribution optimale consiste à placer les 8 pierres précieuses supplémentaires dans le wagonnet 4.

Ensuite, nous pouvons construire une voie d'évitement de capacité 2. Les wagonnets peuvent s'aligner dans un ordre non décroissant de la façon suivante :

1. Déplacer le wagonnet 4 de l'autre côté de la voie d'évitement
2. Déplacer le wagonnet 3 dans la voie d'évitement
3. Déplacer le wagonnet 2 dans la voie d'évitement
4. Déplacer le wagonnet 1 de l'autre côté de la voie d'évitement

The English version appears before the French version.

5. Déplacer le wagonnet 2 hors de la voie d'évitement
6. Déplacer le wagonnet 3 hors de la voie d'évitement
7. Déplacer le wagonnet 3 de l'autre côté de la voie d'évitement
8. Déplacer le wagonnet 2 de l'autre côté de la voie d'évitement

Données d'entrée du 3e exemple

```
4 123456789
40 30 20 10
```

Données de sortie du 3e exemple

```
3
```

Justification des données de sortie du 3e exemple

Aucun wagonnet n'est vide ; la seule distribution possible des pierres précieuses supplémentaires consiste à ne pas en utiliser.

Ensuite, nous pouvons construire une voie d'évitement de capacité 3. Les wagonnets peuvent s'aligner dans un ordre non décroissant de la façon suivante :

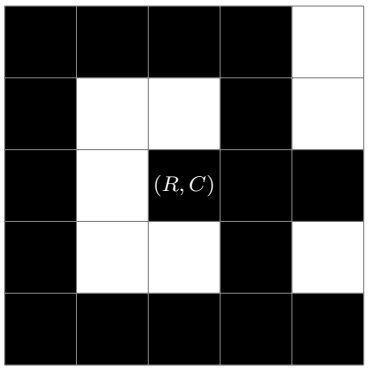
1. Déplacer le wagonnet 4 dans la voie d'évitement
2. Déplacer le wagonnet 3 dans la voie d'évitement
3. Déplacer le wagonnet 2 dans la voie d'évitement
4. Déplacer le wagonnet 1 de l'autre côté de la voie d'évitement
5. Déplacer le wagonnet 2 hors de la voie d'évitement, puis de l'autre côté
6. Déplacer le wagonnet 3 hors de la voie d'évitement, puis de l'autre côté
7. Déplacer le wagonnet 4 hors de la voie d'évitement, puis de l'autre côté

Problem S5: On the Fence

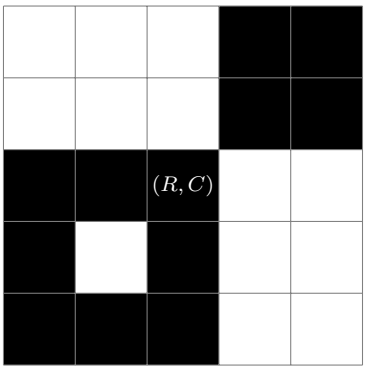
Problem Description

You recently inherited a beautiful plot of land, which can be viewed as a grid with N rows and M columns. Now you want to protect your land from trespassers by building a giant fence on it. You will build this fence by choosing up to K grid cells and filling each of them with a concrete block; these cells are *on the fence*. Due to mysterious construction laws, the cell in row R and column C must be on the fence. Also, the fence cells must form a *single connected component*—you should be able to get between any two cells on the fence by moving vertically or horizontally (*not* diagonally) through fence cells only.

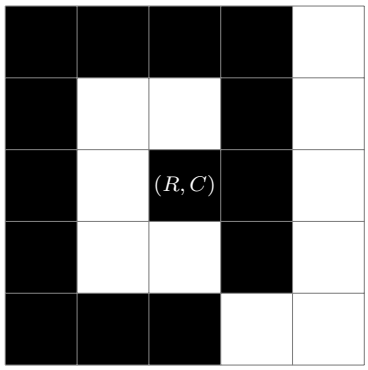
The cells in row 1, column 1, row N, and column M are called *edge cells*. A cell not on the fence is *outside* the fence if you can get from that cell to an edge cell by moving vertically, horizontally, *or* diagonally, without going through any fence cells. Otherwise, the cell is *inside* the fence. The figure below shows some examples of valid and invalid fences.



This is a valid fence made of 17 blocks, with 5 inside cells.



This fence is invalid because it is not connected.



This fence is valid, but it encloses zero inside cells.

You want to protect as much of your land as possible. Find the largest number of inside cells you can enclose within your fence.

Input Specification

The first line of input contains an integer T, the number of test cases in the input.

The next T lines each contain a test case, consisting of five space-separated integers: N, M, K, R, and C ($1 \leq K \leq NM$, $1 \leq R \leq N$, $1 \leq C \leq M$).

The table on the next page shows how the available 15 marks are distributed.

La version française figure à la suite de la version anglaise.

Marks Awarded	Bounds on T	Bounds on N, M	Additional Constraints
1 mark	$T \leq 1000$	$1 \leq N, M \leq 10^9$	$K = NM$
2 marks	$T \leq 10$	$1 \leq N, M \leq 6$	None
2 marks	$T \leq 10$	$1 \leq N, M \leq 40$	None
2 marks	$T \leq 10$	$1 \leq N, M \leq 300$	None
2 marks	$T \leq 10$	$1 \leq N, M \leq 2\,000$	None
3 marks	$T \leq 10$	$1 \leq N, M \leq 10^6$	None
3 marks	$T \leq 1000$	$1 \leq N, M \leq 10^9$	None

Output Specification

Output T lines. The t-th line should contain a single integer, the answer to test case t.

Sample Input

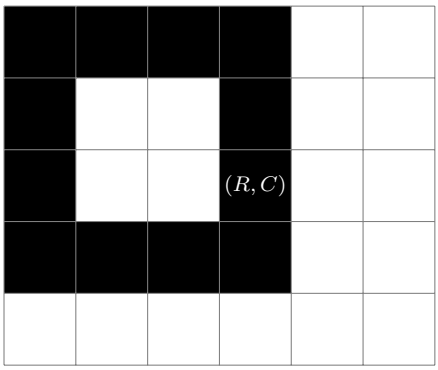
```
2
5 6 12 3 4
3 6 18 2 4
```

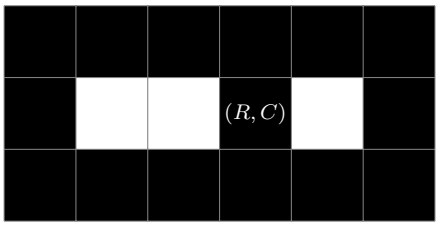
Output for Sample Input

```
4
3
```

Explanation of Output for Sample Input

Below are optimal fences for the two test cases, which enclose 4 and 3 inside cells, respectively:





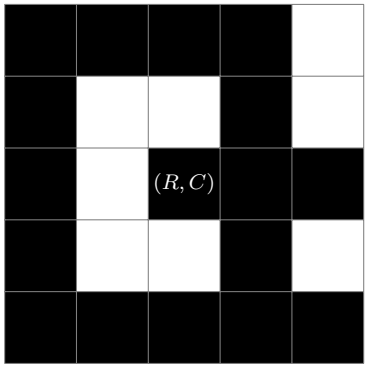
La version française figure à la suite de la version anglaise.

Problème S5: Sur la clôture

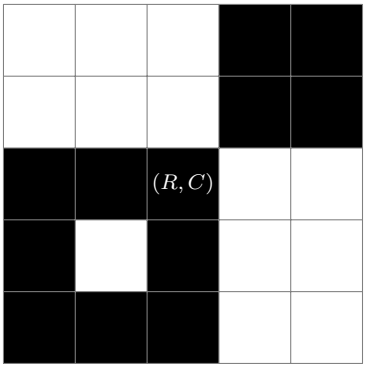
Énoncé du problème

Vous avez récemment hérité d'un magnifique terrain, que l'on peut représenter comme une grille de N lignes et M colonnes. Vous souhaitez protéger votre terrain des intrus en y construisant une clôture géante. Vous allez construire cette clôture en choisissant au maximum K cases de la grille et en remplissant chacune d'elles avec un bloc de béton ; ces cases seront *sur la clôture*. Selon des lois de construction mystérieuses, la case située à la ligne R et à la colonne C doit se trouver sur la clôture. De plus, les cases de la clôture doivent former un *ensemble relié*—il doit être possible de passer d'une case de la clôture à une autre en se déplaçant verticalement ou horizontalement (*et non* en diagonale), en passant uniquement par les cases de la clôture.

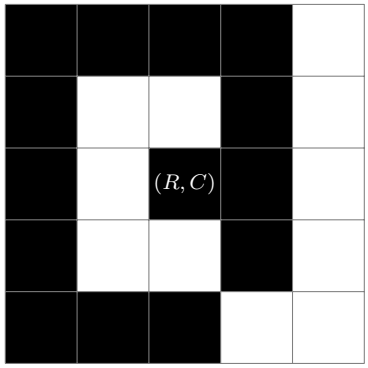
Les cases dans la ligne 1, colonne 1, ligne N et colonne M s'appellent *cases bordure*. Une case qui n'est pas sur la clôture est dite *externe* à la clôture si elle peut être reliée à une case bordure par un trajet se déplaçant verticalement, horizontalement *ou* en diagonale, sans passer par les cases de la clôture. Sinon, la case est dite *intérieure*. La figure ci-dessous illustre quelques exemples de clôtures valides et non valides.



Clôture valide composée de 17 blocs, avec 5 cases intérieures.



Clôture non valide car les cases ne sont pas reliées.



Clôture valide, mais sans case intérieure.

Vous souhaitez protéger votre terrain de manière optimale. Trouvez le plus grand nombre de cases intérieures que votre clôture peut renfermer.

Précisions par rapport aux données d'entrée

La première ligne des données d'entrée contient un entier, T, le nombre de cas de test dans l'entrée.

Les T lignes suivantes contiennent chacune un cas de test, composé de cinq entiers séparés par des espaces : N, M, K, R et C ($1 \leq K \leq NM$, $1 \leq R \leq N$, $1 \leq C \leq M$).

Le tableau suivant indique la manière dont les 15 points disponibles sont répartis :

The English version appears before the French version.

Points attribués	Bornes sur T	Bornes sur N, M	Contraintes supplémentaires
1 mark	$T \leq 1000$	$1 \leq N, M \leq 10^9$	$K = NM$
2 marks	$T \leq 10$	$1 \leq N, M \leq 6$	None
2 marks	$T \leq 10$	$1 \leq N, M \leq 40$	None
2 marks	$T \leq 10$	$1 \leq N, M \leq 300$	None
2 marks	$T \leq 10$	$1 \leq N, M \leq 2\,000$	None
3 marks	$T \leq 10$	$1 \leq N, M \leq 10^6$	None
3 marks	$T \leq 1000$	$1 \leq N, M \leq 10^9$	None

Précisions par rapport aux données de sortie

Afficher T lignes. La t-ème ligne doit contenir un seul entier, la réponse au test t.

Exemple de données d'entrée

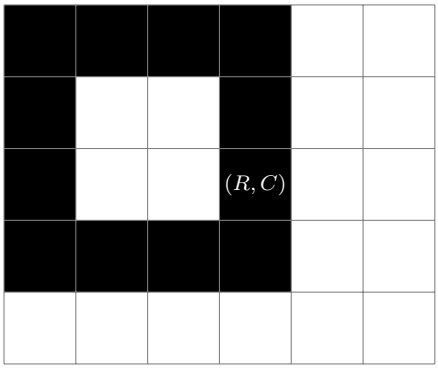
```
2
5 6 12 3 4
3 6 18 2 4
```

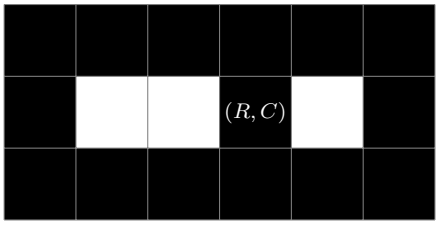
Exemple de données de sortie

```
4
3
```

Justification des données de sortie

Ci-dessous figurent les clôtures optimales pour les deux cas de test, qui entourent respectivement 4 et 3 cellules internes :





The English version appears before the French version.