CATEGORÍA 1

RETO CALENTAMIENTO: NÚMEROS PRIMOS

Construir un programa que imprima los N primeros números primos.

Prueba #	**Datos de Entrada (Archivo: NumerosPrimos_#.IN)**	**Datos de Salida (Archivo: NumerosPrimos _#.OUT)**
1	10	1 2 3 5 7 11 13 17 19 23

2	50	1 2 3 5 7 11 13
		17 19 23 29 31 37
		41 43 47 53 59 61
		67 71 73 79 83 89
		97 101 103 107 109 113
		127 131 137 139 149 151
		157 163 167 173 179 181
		191 193 197 199 211 223
		227

3	7	1 2 3 5 7 11 13

Nota: entre número y número va un tabulador.

RETO # 1: EQUIPOS

Pedrito trabaja en un programa de deportes. Ha pasado mucho tiempo viendo los partidos de fútbol de su país. Después de un rato empezó a notar patrones diferentes. Por ejemplo, cada equipo tiene dos juegos de uniformes: uniformes de local y uniforme de visitante. Cuando un equipo juega un partido en casa, los jugadores se ponen el uniforme de local. Cuando un equipo juega en el estadio de otro, los jugadores se ponen el uniforme de visitante. La única excepción a esta regla es la siguiente: cuando el color uniforme del equipo local coincide con el uniforme de los visitantes, el equipo anfitrión se pone su uniforme de visitante. Para cada equipo el color de los uniformes de local y de visitantes es diferente.

Hay n equipos participantes en el campeonato nacional. El campeonato consta de n * (n - 1) juegos: cada equipo recibe a otro equipo en su estadio. En este punto Pedrito se preguntó: ¿cuántas veces durante el campeonato un equipo anfitrión va a ponerse el uniforme de visitante? Tenga en cuenta que el orden de los juegos no afecta a este número.
Usted sabe los colores del uniforme de local y de visitante para cada equipo. Por simplicidad, los colores se numeran por números enteros de tal manera que no hay dos colores distintos con el mismo número. Ayuda Pedrito encontrar la respuesta a su pregunta.

Entrada
La primera línea contiene un entero **n** ($2 \leq n \leq 30$) – Cantidad de equipos. Cada una de los siguientes n líneas contienen un par de distintos números enteros separados por un espacio hi ai ($1 \leq hi, ai \leq 100$) - los colores de los uniformes de local y de visitantes del i-ésimo equipo, respectivamente.

Salida
En una sola línea imprimir el número de juegos en los que el equipo local va a jugar con el uniforme de visitante.

Ejemplos

Entrada
```
3
1 2
2 4
3 4
```
Salida
```
1
```
Entrada
```
4
100 42
42 100
5 42
100 5
```
Salida

```
5
```

Nota
En el primer ejemplo el campeonato consta de 6 juegos. El único juego con el evento en cuestión es el partido entre los equipos de 2 y 1 en el estadio del equipo 2.

En el segundo ejemplo el equipo anfitrión tendrá que usar el uniforme de los visitantes en los partidos disputados entre los equipos: 1 y 2, 2 y 1, 2 y 3, 3 y 4, 4 y 2.

RETO # 2: JOHNNY B. GOODE

Johnny is a really nice and respectful person, in sharp contrast to his little brother, who is a very nasty and disrespectful person. Johnny always sends messages to his friends in all small letters, whereas the little brother sends messages in all capital letters.

You just received a message given by a string **s**. You don't know whether this message is sent by Johnny or his brother. Also, the communication channel through which you received the message is erroneous and hence can flip a letter from uppercase to lowercase or vice versa. However, you know that this channel can make at most **K** such flips.

Determine whether the message could have been sent only by Johnny, only by the little brother, by both or by none.

Input

- The first line of the input contains a single integer **T** denoting the number of test cases. The description of **T** test cases follows.
- The second line contains two space-separated integers **N** and **K** denoting the length of the string **s** and the maximum number of flips that the erroneous channel can make.
- The third line contains a single string **s** denoting the message you received.

Output

For each test case, output a single line containing one string — "Johnny", "brother", "both" or "none".

Constraints

- 1 ≤ **T** ≤ 1000
- 1 ≤ **N** ≤ 100
- 0 ≤ **K** ≤ **N**
- **s** consists only of (lowercase and uppercase) English letters

Example

Input

```
4

5 1

frauD

FRAUD
```

Life

LoVer

Output

Johnny

brother

Johnny

None

Two first cases explanation

Example case 1: Only one flip is possible. So it is possible that Johnny sent "fraud" and the channel flipped the last character to get "frauD". However, it is not possible for the brother to have sent "FRAUD", because then it would need 4 flips. Hence the answer is "Johnny".

Example case 2: Only one flip is possible. So it is possible that the brother sent "FRAUD" and the channel didn't flip anything. However, it is not possible for Johnny to have sent "fraud", because then it would need 5 flips. Hence the answer is "brother".

RETO # 3: EL PEQUEÑO OSO

El pequeño OSO ama mucho su tierra. Su mayor amor está en el pueblo de “La Miel”.

Sin embargo,   La Miel es pequeño para establecerse, por lo que el Pequeño OSO quiere ir a alguna otra ciudad. El pequeño OSO no le gusta pasar mucho tiempo de viaje, por lo que para su viaje va a elegir una ciudad a la que necesite un tiempo mínimo para llegar. Si hay múltiples ciudades con la misma condición, el pequeño OSO no se va a ninguna parte.

Para cada pueblo, excepto La Miel, se sabe el tiempo necesario para viajar a esta ciudad. Buscar la ciudad a la que el pequeño OSO irá a o imprimir "Me quedo en La Miel", si se queda en La Miel.

Entrada

La primera línea contiene un entero **n** (1 ≤ **n** ≤ 105) - el número de ciudades. La siguiente línea contiene **n** enteros, separados por espacios: el i-ésimo entero representa el tiempo necesario para ir del pueblo La Miel a la i-ésima ciudad. Los valores de tiempo son números enteros positivos, que no exceden de 109.

Se puede considerar a las ciudades numerados del 1 al n, ambos inclusive. La Miel no se encuentra entre las ciudades numeradas.

Salida

Imprimir la respuesta en una sola línea - el número de la población del pequeño OSO irá. Si hay varias ciudades con mínimo tiempo de viaje, imprime "Me quedo en La Miel " (sin las comillas).

Ejemplos

Entrada

```
2
7 4
```

Salida

```
2
```

Entrada

```
7
7 4 47 100 4 9 12
```

Salida

```
 Me quedo en La Miel
```

Nota

En el primer ejemplo sólo hay dos ciudades donde el pequeño OSO puede ir. El tiempo de viaje hasta la primera ciudad es igual a 7, a la segunda es 4. La ciudad más cercana a La Miel es la segunda, por lo que la respuesta es 2.

En el segundo ejemplo las ciudades más cercanas son las ciudades de dos y cinco, el tiempo de viaje de las dos es igual a 4, por lo que la respuesta es "me quedo es La Miel".

RETO # 4: THE SPORT OF KINGS

Tennis is a popular game. Consider a simplified view of a tennis game from directly above. The game will appear to be played on a 2 dimensional rectangle, where each player has his own court, a half of the rectangle. Consider the players and the ball to be points moving on this 2D plane. The ball can be assumed to always move with fixed velocity (speed and direction) when it is hit by a player. The ball changes its velocity when hit by the other player. And so on, the game continues.

Geek also enjoys playing tennis, but in $n+1$ dimensions instead of just 3. From the perspective of the previously discussed overhead view, Geek's court is an n-dimensional hyperrectangle which is axis-aligned with one corner at $(0,0,0,\ldots,0)$ and the opposite corner at $(l1,l2,l3,\ldots,ln)$. The court of his opponent is the reflection of Geek's court across the $n-1$ dimensional surface with equation $x1=0$.
At time $t=0$, Geek notices that the ball is at position $(0,b2,\ldots,bn)$ after being hit by his opponent. The velocity components of the ball in each of the n dimensions are also immediately known to Geek, the component in the ith dimension being vi. The ball will continue to move with fixed velocity until it leaves Geek's court. The ball is said to leave Geek's court when it reaches a position strictly outside the bounds of Geek's court. Geek is currently at position $(c1,c2,\ldots,cn)$. To hit the ball back, Geek must intercept the ball before it leaves his court, which means at a certain time the ball's position and Geek's position must coincide.
To achieve this, Geek is free to change his speed and direction at any time starting from time $t=0$. However, Geek is lazy so he does not want to put in more effort than necessary. Geek wants to minimize the maximum speed that he needs to acquire at any point in time until he hits the ball. Find this minimum value of speed s_{min}.

Note: If an object moves with fixed velocity $\vec{v}$ and is at position $\vec{x}$ at time 0, its position at time t is given by $\vec{x} + \vec{v} * t$.

Input

- The first line contains t, the number of test cases. t cases follow.
- The first line of each test case contains n, the number of dimensions.
- The next line contains n integers $l_1,l_2,\ldots,l_n$ the bounds of Geek's court.
- The next line contains n integers $b_1,b_2,\ldots,b_n$, the position of the ball at $t=0$.
- The next line contains n integers $v_1,v_2,\ldots,v_n$ the velocity components of the ball.
- The next line contains n integers, $c_1,c_2,\ldots,c_n$, Geek's position at t=0.

Output

- For each test case, output a single line containing the value of s_{min}. Your answer will be considered correct if the absolute error does not exceed 10^{-2}.

  10

Constraints

- $1 \le t \le 1500$
- $2 \le n \le 50$
- $1 \le l_i \le 50$
- $0 \le b_i \le l_i$ and $b_1=0$
- $-10 \le v_i \le 10$ and $v_1>0$
- $0 \le c_i \le l_i$
- It is guaranteed that the ball stays in the court for a non-zero amount of time.

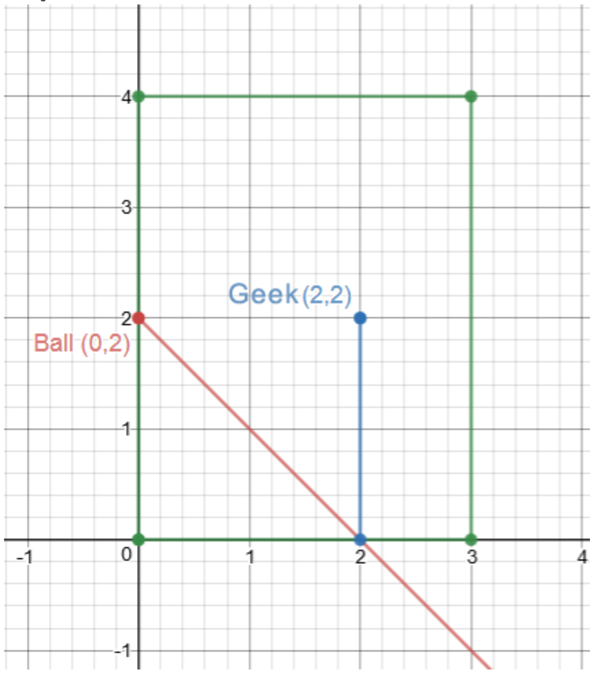
Sample Input

```
2

2

3 4

0 2

2 -2

2 2

3

10 10 10

0 0 0

1 1 1

5 5 5
```

Sample Output

```
2.0000

0.0000
```

ExplanationCase 1: The court is 2-dimentional.



The ball's trajectory is marked in red. For Geek it is optimal to move along the blue line at a constant speed of 2 until he meets the ball at the boundary.

Case 2: The court is 3-dimensional and the ball is coming straight at Geek. So it is best for Geek to not move at all, thus $s_{min}=0$.

RETO # 5: WARCRAFT BATTLES

In the world of **WarCraft** there are fierce warriors called **Thralls**. Also there are even fiercer warriors called **Illidians** – the mortal enemies of **Thralls**.

The power of each warrior is determined by the amount of chakra he possesses which is some positive integer. Warriors with zero level of chakra are dead warriors :) When the fight between **Thrall** with power C_I and **Illidian** with power C_F occurs the warrior with lower power will die and the winner will lose the amount of chakra that his enemy have possessed before the fight. So three cases are possible:

- $C_I > C_F$. Then **Illidian** will die while the new power of **Thrall** will be $C_I - C_F$.
- $C_I < C_F$. Then **Thrall** will die while the new power of **Illidian** will be $C_F - C_I$.
- $C_I = C_F$. In this special case both warriors die.

Each warrior (**Thrall** or **Illidian**) has his level of skills which is denoted by some positive integer. The fight between two warriors can occur only when these warriors are **Thrall** and **Illidian** of the same level. In particular, friendly fights are not allowed, i.e., a **Thrall** cannot fight with another **Thrall** and the same holds for **Illidians**.

Lets follow the following convention to denote the warriors. A **Thrall** of level **L** and power **C** will be denoted as **(I, C, L)**, while **Illidian** of level **L** and power **C** will be denoted as **(F, C, L)**. Consider some examples. If **A = (I, 50, 1)** fights with **B = (F, 20, 1)**, **B** dies and **A** becomes **(I, 30, 1)**. On the other hand, **(I, 50, 1)** cannot fight with **(F, 20, 2)** as they have different levels.

There is a battle between **Thralls** and **Illidians**. There are **N Thralls** and **M Illidians** in all. The battle will consist of series of fights. As was mentioned above in each fight one **Thrall** and one **Illidian** *of the same level* take part and after the fight the warrior with lower power will die (or both will die if they have the same power). The battle proceeds as long as there exists at least one pair of warriors who can fight. The distribution of warriors by levels satisfies the following condition: for every **Thrall** of level **L** there exists at least one **Illidian** of the same level **L** and vice-versa. So if for some level **L** we have at least one warrior of this level then there is at least one **Thrall** of level **L** and at least one **Illidian** of level **L**.

There is a powerful wizard, whose name is **Archmage**, on the side of **Thralls**. He can increase the amount of chakra of each **Thrall** by any number. **Archmage** wants the army of **Thralls** to win this battle. But increasing amount of chakra of any **Thrall** by one costs him a lot of his magic power. Hence he wants to minimize the total amount of additional chakra he should give to **Thralls** in order for them to win. Note, however, that the win here means that all **Illidians** should be dead regardless of whether any **Thrall** is alive. Also note that the battle can proceed by different scenarios and the **Archmage** need to distribute additional chakra among the **Thralls** in such a way that they will win for any possible battle scenario. Help **Archmage** and find the minimal amount of additional chakra he should give to **Thralls** in order for them to win.

Input

The first line of the input contains an integer **T**, the number of test cases. **T** test cases follow. The first line of each test case contains two space separated integers **N** and **M**. Here **N** is the number of **Thralls** participating in the battle and **M** is the number of **Illidians** for the same. Each of the next **N** lines contains two space separated integers **C_i** and **L_i**, the amount of chakra and level of **i**-th **Thrall** correspondingly. The next **M** lines describe power and level of **Illidians** participating in the battle in the same format.

Output

For each test case output a single integer on a single line, the minimum amount of chakra **Archmage** should give to **Thralls** in order for them to win the battle.

Constraints

Each integer in the input file is positive and does not exceed **100**. That is

1 ≤ T ≤ 100
1 ≤ N ≤ 100
1 ≤ M ≤ 100
1 ≤ Ci ≤ 100
1 ≤ Li ≤ 100

For every **Thrall** of level **L** there exists at least one **Illidian** of the same level **L** and vice-versa.

It is guaranteed that each official test file will satisfy all these constraints. You DON'T need to verify them in your program.

Example

Input:

2

2 3

10 1

20 2

5 2

5 2

18 1

5 5

73 87

69 13

36 36

77 46

43 93

49 46

74 93

78 87

99 13

59 36

Output:

8

89

Explanation

Case 1. The warriors are **I1 = (I, 10, 1), I2 = (I, 20, 2), F1 = (F, 5, 2), F2 = (F, 5, 2), F3 = (F, 18, 1)**. Without the **Archmage** help the battle can proceed as follows.

- **I2** fights with **F1**, **F1** dies, **I2** becomes **(I, 15, 2)**.
- **I2** fights with **F2**, **F2** dies, **I2** becomes **(I, 10, 2)**.
- **I1** fights with **F3**, **I1** dies, **F3** becomes **(F, 8, 1)**.

So if **Archmage** will give **8** additional units of chakra to **I1** the **Thralls** will win the battle and even one **Thrall** (**I2**) will left alive. Hence the answer is **8**.