Entre los números abundantes, es decir, aquellos que son menores que la suma de sus divisores, se considera que son números **casi perfectos** aquellos tales que la suma de sus divisores es uno menos que el número. Así, el 1616 es un número casi perfecto ya que 1+2+4+8=151+2+4+8=15, teniendo en cuenta que 1,2,41,2,4 y 88 son los divisores de 1616 menores a él, y 1515 es uno menos que 1616. Su tarea consiste en escribir un programa que lee un grupo de números enteros e imprime un mensjae indicando si estos números son casi perfectos o no.

**Input**

La entrada consiste de muchos casos a evaluar. La primera línes contiene el número de casos de prueba a realizar. Llámemos a este número NN con 1≤N≤1001≤N≤100. Cada una de las siguientes líneas contiene un entero positivo XX con 1≤X≤1081≤X≤108, que pueden ser números casi perfectos o no.

**Output**

Para cada uno de los NN números de los casos de prueba se debe imprimir el mensaje X es casi perfecto o X no es casi perfecto de acuerdo a las especificaciones anteriormente presentadas.

**Sample**

| **Input** | **Output** |
| --- | --- |
| 4  1  1024  68  4096 | 1 no es casi perfecto  1024 es casi perfecto  68 no es casi perfecto  4096 es casi perfecto |

Ramsay ha capturado a Theon Greyjoy y le ha prometido torturarlo de forma ejemplar para que no vuelva a escaparse. La tortura consistirá en una serie de latigazos dados por un verdugo experto, pero Ramsay, en su infinita benevolencia le permitirá a Theon escoger el verdugo que lo castigará.

Los verdugos están numerados desde el 11 hasta NN. Theon debe escoger su verdugo indicando uno de esos números. Cada verdugo le ha indicado a Ramsay la cantidad de latigazos que le dará a Theon y nuestra misión es ayudar a Theon a escoger cual es el verdugo que le proporcionará la menor cantidad de latigazos.

**Input**

La entrada consiste de dos líneas únicamente. La primera línea contiene el número NN (1≤N≤1001≤N≤100) que indica la cantidad de verdugos que están dispuestos a torturar al pobre Theon. La siguiente línea tiene NN números enteros indicando el número de latigazos que le proporcionará el verdugo a Theon.

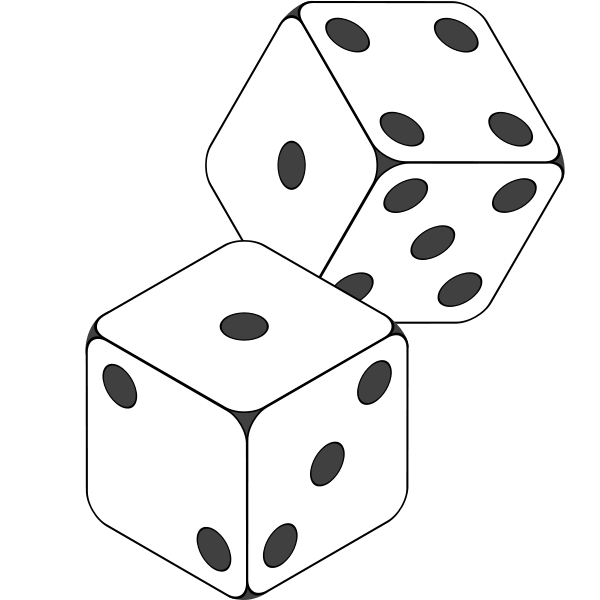
**Output**

Imprima una sola línea que contiene el número del verdugo que debe escoger Theon. Si hay más de una posible respuesta, hay que imprimir el que tenga el número más pequeño.

**Sample**



El juego de **Dados Raros** juegan dos personas con dos dados. Cada persona lanza los dos dados, pero las reglas para determinar quien gana son diferentes en este juego.



Aunque gana el jugador que obtenga el puntaje más alto, para saber quien tiene este puntaje más alto, se debe tener en cuenta lo siguiente:

* Un tresoro (combinación de dados 1-2 o 2-1) es el puntaje más alto de todos.
* Luego, los pares (combinación 1-1 o 2-2 o 3-3 o 4-4 o 5-5 o 6-6) serían el segundo puntaje más alto. Si hay empates (ambos jugadores obtuvieron pares), ganaría el jugador que obtuvo el número más grande.
* Para el resto de casos, gana el jugador que haya obtenido el puntaje numérico más alto. Pero para obtener este puntaje hay que unir los números que se obtuvo en cada dado, colocando el número mayor como primer dígito y el número menor como el segundo dígito (por ejemplo, si al lanzar los dados obtuvo un 3 y un 4, el puntaje numérico sería 43).

Por ejemplo, el jugador 1 obtiene un 2-2 y el jugador 2 obtiene un 6-5, gana el jugador 1, ya que obtuvo un par. Pero si el jugador 1 obtiene un 6-6 y el jugador 2 obtiene un 1-2, gana el jugador 2, ya que obtubo un tresoro (1-2 o 2-1 le gana a todo).

Su misión es escribir un programa que lea los valores de los dados lanzados por los dos jugadores y que indique quien ganó o si hubo empate.

#### Input

La entrada contiene muchos casos. Cada caso contiene una línea con cuatro números enteros separados por un solo espacio, de la forma A B C D, donde A y B representa los valores de los dados cuando lo lanza el jugador 1 (tenga en cuenta que 1≤A,B≤61≤A,B≤6), y C y D son los valores de los dados cuando los lanza el jugador 2.

Para finalizar el programa, se ingresa una línea que contiene 4 ceros separados por un espacio (0 0 0 0).

#### Output

Por cada línea de los casos de prueba, debe aparecer en la salida la frase "Player 1 wins." si el jugador 1 ganó, o "Player 2 wins." si ganó el jugador 2 o "Tie." si hubo un empate.

#### Sample

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Vamos a construir un programa que reciba un conjunto de números enteros no negativos y que ordene estos números de acuerdo con los siguientes criterios: Primero los impares en orden ascendente seguido por los pare en orden descendente.

#### Input

La primera línea de entrada contiene un número entero positivo NN (1≤N≤1001≤N≤100). Este corresponde al número de líneas de entrada que siguen. Las siguientes NN líneas contienen, cada una, un número entero no negativo entre 1 y 1,0001,000.

#### Output

Imprima todos los números de acuerdo con la explicación presentada anteriormente. Cada número debe imprimirse en una línea como se muestra a continuación.

#### Sample

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

En el nuevo Edificio Légacy de la Universidad tenemos unas series de NN salas de estudio para los estudiantes de la institución (2≤N≤3,000,0002≤N≤3,000,000), numeradas de 0…N−10…N−1, con la sala N−1N−1 adyacente a la sala 00.

Cuando un estudiante desea usar una sala, utiliza la sala que más le gusta de todas las que tiene la universida (su *sala favorita*). Sin embargo, si la sala favorita ya está ocupada por otra persona, el estudiante revisa secuencialmente a partir de esta sala hasta que encuentra la primera sala no ocupada, la cual reclama para su suso. En el proceso de revisión, si pasa por la sala N−1N−1, el estudiante continua revisando a partir de la sala 00.

Entonces, dada la sala favorita de cada estudiante, determine la menor cantidad de salas que permanecen desocupadas después que todos los estudiantes han solicitado una sala de la universidad. Note que la respuesta a esta pregunta no depende del orden en que los estudiantes usan las salas.

**Input**

Para evitar problemas leyendo una entrada muy grande, la entrada a este proble se da en un formato conciso: usamos KK líneas (1≤K≤10,0001≤K≤10,000), cada una de estas líneas tiene la forma:

X Y A B

Donde tenemos las preferencias de X×YX×Y estudiantes en total, de la siguiente forma: XX estudiantes prefieren cada uno de las salas f(1),f(2)⋯f(Y)f(1),f(2)⋯f(Y), donde f(i)=(A×i+B)modNf(i)=(A×i+B)modN. Los valores de AA y BB estan en el rango de 0…1,000,000,0000…1,000,000,000.

En la entrada tendremos:

* Línea 1: Dos enteros separados por espacio: N y K.
* Línea de la 2 a la K+1K+1: cada línea contiene los enteros X Y A B, interpretados como se explicó anteriormente. El número total de estudiantes especificados por todas esas líneas será máximo N−1N−1. Se pueden añadir estudiantes a la misma sala por varias de esas líneas.

ENTRADA EJEMPLO:

10 3

3 2 2 4

2 1 0 1

1 1 1 7

EXPLICACIÓN DE LA ENTRADA:

Hay 10 salas en la Universidad, numeradas de la 0 a la 9. La segunda línea dice que 3 estudiantes prefieren la sala (2 \* 1 + 4) % 10 = 6 y otros 3 estudiantes prefieren la sala (2 \* 2 + 4) % 10 = 8. Note que en esta línea hay 2 salas preferidas ya que el segundo número (la variable Y) es un 2.

La tercera línea dice que 2 estudiantes prefieren la sala (0 \* 1 + 1) % 10 = 1. En esta línea hay una sola sala preferida, ya que el segundo número en la línea (o sea, la variable Y) es igual a 1.

La cuarta línea especifica que 1 estudiante prefiere la sala (1 \* 1 + 7) % 10 = 8 (entonces un total de 8 estudiantes les gusta esta sala 8, 3 de la primera línea y 1 de esta línea).

**Output**

La salida es un solo número entero, la posición de la sala que quedó desocupada. Si varias salas quedaron desocupadas, debe mostrarse aquella sala que tiene la posición más pequeña.

Para el ejemplo anterior la salida sería:

5

Que significa que todas las salas terminaron ocupadas excepto la sala 5.

En la última cena que organizó la Universidad, se sentaron NN personas en una mesa redonda a compartir los platos de la cena. En la mitad de la mesa hay un gran florero que sirve de adorno para la mesa. Las personas que están sentadas en los lados opuestos de la mesa no pueden verse las unas a las otras, ya que el florero bloquea su vista. Su misión es escribir un programa que determine el número de parejas de personas que SI pueden verse la una a la otra a través de una línea directa de vista.

Vamos a indicar que el florero está centrado en el origen (0,0)(0,0) y tiene un radio RR. Ninguna persona está situada dentro o en el círculo correspondiente al florero y ningún par de personas está sentada en una recta tangente al florero. El valor de RR está en el rango 1…1,000,000,0001…1,000,000,000 y cada persona está sentada en un punto con coordenadas enteras en el rango −1,000,000…+1,000,000−1,000,000…+1,000,000.

#### Input

La entrada consiste de varias líneas. En la primera línea encontrarán dos números enteros N y R separados por un espacio, que indican el número de personas que se sientan en la mesa, así como el radio del florero.

En las N líneas a continuación encontraremos también dos números enteros separados por un espacio. Estos números especifican las coordenadas (X,Y)(X,Y) de una persona sentada en la mesa alrededor del florero.

#### Output

Un solo número, que indica el número de pares de personas que se pueden ver la una a la otra.

#### Sample

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

En su tiempo libre el Doctor Doolitle gerencia un zoológico. Cada mañana, cuando se levanta, visita el zoológico con el objetivo de comprobar que ninguno de sus animales haya escapado en la noche. La lista de sus animales es enorme y aunque puede chequear visualmente la presencia de estos, al final no puede estar seguro que estén todos los animales en sus jaulas. Adicionalmente, al doctor solo le interesa saber la cantidad de animales que hay de cada tipo y no le interesa para nada, los detalles específicos de cada especie. Por ejemplo, si hay un perro labrador y perro chihuahua, al doctor Doolitle solo le interesa saber que tiene dos perros.

Su misión es escribir un programa en que dado la descripción de NN animales, genere una lista en orden alfabético de los diversos animales que tiene el doctor Doolitle en su zoológico, seguido de la cantidad de animales.

#### Input

La entrada contendrá muchos casos de prueba. Cada caso de prueba consiste de varias líneas. La primera línea de cada caso contiene un número entero positivo NN (con 1≤N≤1031≤N≤103). Luego seguirán NN líneas con nombres de animales en cada una de ellas. El nombre de cada animal consiste de una o más palabras, en minúsculas o mayúsculas. Como los nombres están en inglés, la palabra que describe el animal es la última de cada línea. Para finalizar el programa, hay que tener en cuenta que la entrada NN sea igual a cero.

#### Output

Para cada caso de prueba, mostrar el número de la lista, seguido por los animales que el Doctor Doolitle tiene en su zoológico en minúsculas y en orden alfabético, con cada nombre del animal seguido por un especio, el caracter | y luego otro espacio, y la cantidad de ese animal que hay en esa lista.

#### Sample

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

La profe Marce está tratando de diseñar un juego para sus alumnos de tercero de primaria que le ayude a practicar las operaciones básicas de la aritmética, es decir, suma, resta, multiplicación y división. La profe le gustaría que cada estudiante en su clase sea capaz de "pensar matemáticamente" y determinar, dado dos números, estos pueden ser sumados, restados, multiplicados o divididos de alguna forma, para producir un tercer número. Por eso nos ha pedido que le ayudemos construyendo un programa que le facilite la tarea de evaluar el desempeño de sus estudiantes.

Dibujo en blanco y negro

Descripción generada automáticamente con confianza baja

Su misión es escribir un programa que lee tres número enteros positivos y determina si es o no posible, sumar, restar, multiplicar o dividir los dos primeros números en cualquier orden para producir el tercer número. Solo una operación puede ser utilizada para producir el tercer número.

#### Input

La entrada consta de varias líneas. La primera línea contiene el número entero NN (1≤N≤10,0001≤N≤10,000) que indica el número de casos de prueba que vienen a continuación. Las siguientes NN líneas contiene un conjunto de 3 números a,b,ca,b,c (1≤a,b,c≤10,0001≤a,b,c≤10,000) separados por un espacio.

#### Output

Para cada caso de prueba, determine si es posible o no producir el tercer número, cc, usando los primeros dos números aa y bb, ya sea a través de la suma, o la resta o la multiplicación o la división entera de ambos números.

#### Sample

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Logo es uno de los lenguajes de programación más antiguos que existe. Este lenguaje se diseñó para enseñar a los niños los primeros pasos en el arte de programar. Los programas en este lenguaje consisten en mover una tortuga por un campo con el objetivo de encontrar un diamante escondido en ese campo.

Calendario

Descripción generada automáticamente

En este problema vamos a construir un "interpretador" de una versión simple del lenguaje Logo. En este caso, la tortuga se moverá en un campo de 8×88×8. En el campo encontremos la tortuga en una de la casillas que está marcada con la letra T. En el campo también encontraremos casillas vacías (marcadas como .), así como obstáculos que impiden el movimiento de la tortuga. Estos obstáculos pueden ser de roca (marcado como C) o de hielo (casillas marcadas como I). El diamante, que es el objetivo de la tortuga, se encuentra en alguna casilla marcada con la letra D. Además, la tortuga puede moverse únicamente de una casilla vacía a otra casilla vacía o a la casilla donde se encuentra el diamante.

El lenguaje tiene 4 instrucciones que permiten el movimiento de la tortuga:

* F: la tortuga se mueve a la casilla que tiene en frente, dependiendo de la dirección en la que esté mirando. Si al frente de la tortuga hay un obstáculo o está en el borde del campo, el programa termina abruptamente debido a este error.
* R: la tortuga gira 90 grados a la derecha. OJO: la tortuga queda en la misma casilla en la que estaba.
* L: La tortuga gira 90 grados a la izquierda. La tortuga queda en la misma casilla en la que se encontraba.
* X: La tortuga lanza un rayo lazo en la dirección en la que está mirando. Si la casilla al frente de la tortuga es un obstáculo de hielo, éste obstáculo se derrite y esa casilla queda vacía. En otro caso, el programa termina abruptamente debido a un error. La tortuga no se mueve ni cambia de dirección. OJO: Es un error del programa disparar un láser a una casilla vacía o a obstáculos de roca o por fuera del campo.

**Input**

La entrada consiste siempre de 9 líneas. Las primeras 8 líneas contiene el campo donde se moverá la tortuga. Cada letra de esas primeras 8 líneas corresponde al contenido de cada una de las casillas del campo. La tortuga siempre estará en la esquina inferior izquierda del campo. Hay exactamente un diamante en el campo. En la novena línea se encuentra el programa que la tortuga deberá ejecutar.

**Output**

El programa deberá mostrar Diamond! si después de ejecutar el programa no hay error alguno y la tortuga queda en la casilla donde está el diamante. Pero si hay algún error en el programa o si después de finalizar el programa, la tortuga no está en el diamente, deberá mostrarse la palabra Bug!.

**Sample**

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente