Problema A
Programando el online de GTA

Has llegado a los estudios de Rockstar, y te han incluido en el equipo de desarrollo del nuevo Grand Theft Auto. No solo eso, si no que te han metido en el grupo que se está encargando de desarrollar el online, ¡Qué chulo!

En el juego online del nuevo GTA se pueden comprar objetos chulos, que además se pueden intercambiar con los jugadores. Por empezar con una tarea sencilla, tu primera tarea es programar un pequeño método que lea el archivo que contiene todos los objetos del juego.

Para combatir los tramposos en el juego, la idea es que con cada transacción (Compra, o intercambio), el juego compruebe que el objeto que se está cambiando o comprando realmente existe. Para ello, a cada objeto se le ha asignado un identificador secreto y único.

Dada la lista de objetos del juego, tu trabajo es identificar, en cada transacción, si el objeto que se está cambiando existe o no.

Entrada

La entrada consistirá en un único caso de prueba, comenzando por un número O de objetos en el juego. Seguirán O líneas, cada una formada por un número I, identificador secreto y único del objeto, el nombre del objeto (Que solo contendrá letras minúsculas del alfabeto inglés, como mucho 20) y un número P, su precio.

A continuación acompañará un número T de transacciones realizadas en el juego, seguido de T líneas, cada una con un número N, el identificador del objeto que se está comprando o intercambiando.

La entrada debe ser leída de forma estándar.

Salida

Por cada transacción T, se debe imprimir "CORRECTO" si el objeto existe en la base de datos del juego, o "TRAMPOSO" si el objeto no existe.

La salida debe ser escrita de forma estándar

Entrada ejemplo	**Salida ejemplo**
3 29173 sable 300 3412 porra 450 26154 lanza 900 5 29173 8263 97312 3412 6545312	CORRECTO TRAMPOSO TRAMPOSO CORRECTO TRAMPOSO

Límites

- $1 \leq O \leq 10^6$
- $1 \leq I, P, N \leq 10^8$

Problema B
Ayudame Atica!

Despúes de un duro cuatrimestre de asignaturas variadas, llega el temido periodo de exámenes para los alumnos de la URJC.

Cada alumno es un mundo, (algunos trabajan, otros se han empezado a viciar a muerte al Apex, otros se encierran en rectorado), por lo que cada alumno tiene un número diferente de horas disponibles para estudiar antes de que comienzen los exámenes. Por cada hora que el alumno decida estudiar una asignatura, aumentará tanto la nota que saca en esa asignatura en un punto. Sabiendo la nota mínima que sacaría cada alumno en cada asignatura si no estudiara nada, ¿sabrías decir quienes aprobarían todas las asignaturas y quienes no?

Input

El primer número representa el número de alumnos (N) o casos de prueba que tenemos que analizar. Para cada alumno, se proveen los siguientes datos:

- En la primera línea, se encuentran, en este orden, el número de asignaturas (M) y las horas disponibles que tiene el alumno.
- En las siguientes M líneas se encuentra la nota (X) que sacaría el alumno si no le dedicara ni un minuto de su vida más. La nota es un valor entre 0 y 100 sin decimales.

La entrada debe ser leída de forma estándar.

Output

Para cada alumno, se deberá escribir "APRUEBA TODO" si distribuyendo de forma inteligente las horas tiene garantizado aprobar todas las asignaturas, o "NOS VEMOS EN JUNIO..." si al menos suspende una asignatura. Una asignatura se considera aprobada si su nota es ≥ 50.

La salida debe ser escrita de forma estándar

Entrada ejemplo	**Salida ejemplo**
3 4 10 40 70 50 40 2 10 48 46 3 5 71 83 59	NOS VEMOS EN JUNIO... APRUEBA TODO APRUEBA TODO

Constraints

- $1 \leq N \leq 1.000$
- $1 \leq M \leq 10.000$
- $0 \leq X \leq 100$

Problema C
Pito Pito Gorgorito

"Pito pito gorgorito donde vas tu tan bonito a la era verdadera pin pan pun fuera tú te vas y tú te quedas"
"En la casa de pinocho todos cuentan hasta 8. 1 2 3 4 5 6 7 8."
"Calcetín suerte para mi si no fuera para mi será para ti"
Todos de pequeño alguna vez hemos utilizado esta forma para determinar quien "se la hinca" a jugar a cualquier juego (escondite, pilla pilla, etc). Dada la cantidad de silabas, calcular dado una posición si nos salvamos de hincarla o no.
Por ejemplo si somos 10 personas y la frase tiene 5 silabas, el 5 la hinca. Si somos 4 personas y la frase tiene 5 silabas la hinca el número 1.

Entrada

Un número T con el número de casos, luego, la descripción de T casos de prueba.

Cada caso contiene tres números: N (número de silabas de la canción) y M (número del grupo de amigos) K (número de posiciones que nos preguntamos si nos la hincaríamos o no). Posteriormente K lineas definiendo cada posición por la que se pregunta.

La entrada debe ser leída de forma estándar.

Salida

Para cada consulta K_i, "SI" si nos la hincamos en dicha posición y "NO" si nos salvamos.

La salida debe ser escrita de forma estándar

Entrada ejemplo	**Salida ejemplo**
2 5 10 2 5 6 5 4 1 1	SI NO SI

Límites

- $1 \leq T \leq 100$

- $1 \leq N, M, K \leq 100.000$
- $1 \leq K_i \leq M$

Problema D
Tramitando papeleo

La tramitación de papeles es ciertamente una lotería. Unas veces, entregas un documento y te dan respuesta en el mismo día. Otras, pueden pasar semanas ¡Incluso meses!

Una mañana decides pasarte por la oficina de tramitación, y te has dado cuenta en cuál es su funcionamiento: Cada vez que un nuevo papel entra en la oficina, se deja al final de todo, el último, y van cogiendo papeles de la mesa para tramitarlos, poniendo los nuevos debajo. Sin embargo, el orden de entrada no corresponde siempre con el de tramitación, parece que hay algo más...

Finalmente, un día pasando de refilón te das cuenta al oir "No, ¡este papel me va a quitar mucho tiempo, lo dejo para luego!" ¡Y ves que lo pone de nuevo al final del todo! Eso cuadra más. Y no solo eso, parece que también hacen lo mismo con algunos papeles de forma aleatoria.

Con toda la información, quieres hacer un programa que calculará, cuanto tardará la oficina de tramitación según los papeles que entren.

Según has observado, sabes que para cualquier determinado tiempo T, si se encuentran con un papel que tarda más de $T/5$ en procesarse, lo dejarán para luego (Excepto cuando $T < 10$, que están empezando y tienen más ganas de trabajar), y que también harán lo mismo con algunos papeles específicos, tarden lo que tarden, pero solo si no lo han visto antes. El proceso de coger un papel es instantáneo, pero el de dejarlo para luego tardará un tiempo fijo de $T = 10$.

Entrada

La entrada corresponderá a un único caso de prueba, comenzando por una línea con un número N de papeles, ordenados según tiempo de entrada a la oficina de tramitación. Corresponderá pues una línea con N papeles, donde cada papel vendrá identificado por un número S, la cantidad de tiempo que se tarda en tramitar. Para mayor facilidad, ningún papel tardará lo mismo en tramitarse que otro. Finalmente acompañará una línea con un número P de papeles que, tarden lo que tarden, dejarán para después. Seguirán P líneas con el tiempo S que se tarda en tramitar cada papel de este tipo.

La entrada debe ser leída de forma estándar.

Salida

Se debe imprimir un único número con el tiempo que se tarda en tramitar todos los papeles, siguiendo el sistema de la oficina de tramitación.

La salida debe ser escrita de forma estándar

Entrada ejemplo	**Salida ejemplo**
6 3 2 4 1 6 10 1 2	66

Ejemplo

- La oficina comienza con T=0.
- Se encuentran un papel de 3, como estamos en T ¡ 10, se tramita. T = 3.
- Se encuentran un papel de 2, como está incluido en la lista de papeles para después, se continúa con el siguiente y no se tramita (dejandolo para más tarde). T = 13.
- Se encuentra un papel de 4, que tarda más en tramitarse que 13/5, así que se deja para después. T = 23.
- Se encuentran un papel de 1, que se tramita. T = 24.
- Papel de 6, que tarda mas en tramitarse que 24/5, se deja para despues. Sucede lo mismo con 10. T = 44.
- A continuación se encuentran de nuevo los papeles de 2, 4, 6 y 10 (En ese orden), que son todos tramitados, ya que 2 se ha dejado para después previamente y todos cumplen la condición de tramitarse en menos de T/5.
- T final = 66.

Límites

- $1 \leq N \leq 10.000$
- $1 \leq S \leq 100.000$
- $1 \leq P \leq 100$

Problema E
Elecciones en la URJC

Este año en la URJC ha habido un gran número de elecciones, por lo que el equipo informático de la universidad se ha puesto a trabajar para recopilar los resultados. Han elaborado un sistema informático al que pueden acceder los responsables de cada campus para introducir los resultados.
Sin embargo, por la rapidez con la que se ha tenido que elaborar este sistema, no se ha terminado de pulir muy bien, y en cada campus lo están utilizando de una forma.
Por un lado, algunas personas están metiendo el número exacto de votos que ha conseguido cada candidato, mientras que otras han decidido introducir el porcentaje de voto que ha conseguido cada uno. Y por si eso fuera poco, en algunos campus el trabajo se ha quedado a la mitad, lo han hecho mezclando los dos métodos al haberlo hecho dos trabajadores distintos, e incluso hay casos donde ni siquiera han terminado de introducir los resultados.
Al menos tenemos la certeza de que los datos ya introducidos al programa son exactos, ya que cada candidato solo puede aparecer una vez por campus. Además, los trabajadores se han asegurado al meter los resultados de cada candidato, que no falta ningún voto por contar para ese candidato en concreto.
Con los resultados actuales, y teniendo en cuenta que, en cada campus, los votos sin contar todavía se le podrían asignar a cualquier candidato cuyos votos no se hayan introducido todavía en ese campus. ¿Podría determinarse un ganador exacto? En caso contrario, ¿Quiénes podrían haber ganado?

Input

La entrada se compone de un único caso de prueba. Cada caso de prueba comienza por un número C de candidatos en las elecciones, seguido de C líneas con los nombres de cada candidato.
A continuación aparecerá un número N de campus que han participado en las elecciones.
Para cada campus, la entrada contendrá en primer lugar, una línea con el nombre del campus, un número P con la cantidad de resultados introducidos en ese campus, y otro número V con la cantidad de votos totales correctos contados en ese campus. Posteriormente aparecerán P líneas con el nombre del candidato, seguido del número K_i o porcentaje L_i de votos recibido. En caso de aparecer un porcentaje, este será un número entero sin decimales.
Los nombres de candidato y de campus son strings que solo contienen letras mínusculas del alfabeto inglés. Se garantiza que el porcentaje de voto será siempre exacto en relación con el número de votos, y también se garantiza que ningún campus introducirá más votos de los totales contados para dicho campus.

La entrada debe ser leída de forma estándar.

Output

La salida tendrá en primer lugar el número de candidatos posibles, acompañado de los candidatos con opciones a ganar, en orden alfabético. En caso de empate entre dos o más de los candidatos, todos ellos serán candidatos con opciones a ganar.

La salida debe ser escrita de forma estándar

Entrada ejemplo	**Salida ejemplo**
4 alvaro maria hector pablo 3 arguelles 3 50 alvaro 10 pablo 5 maria 20% vicalvaro 4 20 alvaro 25% maria 10% hector 45% pablo 20% mostoles 2 100 hector 30 maria 20	2 alvaro hector

Constraints

- $1 \leq C \leq 10.000$
- $1 \leq N \leq 300$
- $0 \leq P \leq C$
- $1 \leq V \leq 3.000.000$
- $0 \leq K_i \leq V$
- $0 \leq L_i \leq 100$

Problema F
El Temible Monstruo Come Números

Gritos despavoridos en Matelandia! Los números corren por sus vidas y se esconden de "El temible monstruo come números". Las profecías de Fermatdamus se han hecho realidad y todos deben huir de él.

Los números más valientes han decidido dar su divisibilidad para detener a este temible monstruo, al parecer, Riemann, un sabio de Matelandia, ha descubierto que el monstruo se va a empachar si se come N números "fortachivisibles", un número N tiene un número K de divisores, mientras más tenga, se considera que el número es más "fortachivisible". Tambien sabe que hay una diversidad tremenda entre los números del pintoresco pueblo y que no todos los números son fortachìvisibles.

Se sabe que los números primos, por ejemplo, solo tienen 2 divisores; 1 y el número mismo, en este caso, el temible monstruo se comerá 2 ndivisores. ¿Puedes descubrir cuántos números valientes de los que se han apuntado a luchar contra el monstruo son necesarios para empacharlo? Adicionalmente, estamos interesados en que el mínimo número de números valientes se pierda.

Entrada

La primera línea contiene un entero T denotando el número de casos de prueba.

Por cada caso de prueba se recibe un número N, el número de números valientes que están dispuestos a sacrificarse para detener al monstruo, luego, N números separados por un espacio denotando el número valiente.

Después de eso, un número K, la cantidad de divisores que el monstruo se tiene que comer para que se empache y deje a los aldeanos de Matelandia en paz.

La entrada debe ser leída de forma estándar.

Salida

Para cada caso debes imprimir cuantos números serán comidos por el temible monstruo come números antes de que se empache, si se come a todos los valientes, se debe escribir "Corred insensatos!"

La salida debe ser escrita de forma estándar

Entrada ejemplo	**Salida ejemplo**
2 4 2 3 5 7 2 4 2 50 123 101010 373	1 Corred insensatos!

Límites

- $1 \leq T \leq 100$
- $1 \leq N \leq 100.000$
- $2 \leq N_i \leq 250.000$
- $1 \leq K \leq 1.000.000.000$