Problem A. Warrencio y el dota

Input file:	`Standard input`
Output file:	`Standard output`
Time limit:	1 second
Memory limit:	64 megabytes

Cierto día Warrencio y sus N amigos fueron al BZ a jugar dota. Cada uno tiene preferencia de usar algunas máquinas y no puede usar otras aunque estén disponibles — al final un jugador sólo puede usar a lo más una máquina pero puede preferir varias. Al llegar a recinto de la sabiduría y las buenas palabras notaron que sólo disponen de m máquinas. Ayuda a Warrencio a calcular cuantos de sus amigos jugarán con él. Warren siempre va a usar una máquina ya que el vicio le hizo tener un trato especial con el dueño del internet.

Input

La entrada consta de T casos de entrada, cada caso sigue el siguiente formato de entrada: La primera línea contiene 3 enteros N, M y P. N es el número de amigos de warren, M es el número de máquinas y P es el número de preferencias de todos los jugadores. Las siguiente P líneas contiene 2 enteros u, v. u es el índice de los jugadores, y v es el índice de las máquinas, $0 \leq u \leq n$ y $0 \leq v < m$ (warren tiene el índice 0) ademas $0 \leq N \leq 99$, $0 \leq M \leq 100$ y $0 \leq P \leq N \times M$.

Output

Para cada caso se debe imprirmir el número máximo de amigos de warren que tienen una maquina para jugar.

Example

Standard input	Standard output
1 3 4 5 0 2 1 0 2 3 3 1 3 2	3

Problem B. Invertidos de Cadenas

Input file:	`Standard input`
Output file:	`Standard output`
Time limit:	TIEMPO second
Memory limit:	64 megabytes

Reciéntemente, un grupo estudiantil decidió digitalizar todos los documentos de texto en la universidad. Xavier, estudiante de Informática quiso agilizar todo el procedimiento mediante algoritmos de reconocimiento de texto en imágenes. Sin embargo, su algoritmo tenía un error que invertía las oraciones. Por ejemplo, uno de los documentos que necesitan ser digitalizado contenía el siguiente texto:

```
La derrota no es el peor de los fracasos.  No
intentarlo es el verdadero fracaso. La  unica
decision  valida  es  no  rendirse  y  seguir
continuando la pelea.
```

Pero el algoritmo de Xavier retornaba:

```
Fracasos  los  de  peor  el es no derrota la.
Fracaso  verdadero  el  es  intentarlo  no.
Pelea la continuando seguir y rendirse no  es
valida decision unica la.
```

Ayuda a Xavier diseñando un algoritmo que reciba como entrada una oración y genere la oración invertida correspondientemente.

Por simplicidad, dentro de una oración sólo el primer caracter de la primera palabra estará en mayúscula, y esto debe de cumplirse siempre. No habrán signos de puntuación a excepción del punto final que indica el fin de una oración. Considerar que entre dos palabras sólo habrá un espacio en blanco de separación.

Input

El problema contiene varios casos de prueba. La primera línea es un entero T ($1 \leq T \leq 10^2$) que denota el número de casos de prueba. Cada caso está compuesto por una oración que contiene una o más palabras. La longitud total de una oración no será mayor a 10^3 caracteres contando espacios en blano y el punto final.

Output

Para cada caso de prueba, el programa deberá imprimir en una línea la oración en el órden correcto.

Example

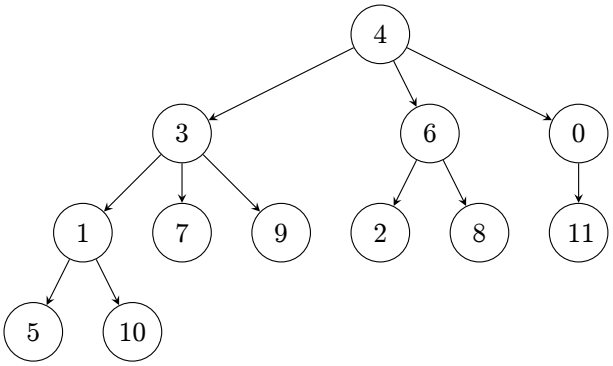
Standard input	Standard output
3 Facil ser de antes dificil es todo. Quieres concurso el ganar si. Debes pregunta esta resolver.	Todo es dificil antes de ser facil. Si ganar el concurso quieres. Resolver esta pregunta debes.

Problem C. Mostrando grafos

Input file:	`Standard input`
Output file:	`Standard output`
Time limit:	1 second
Memory limit:	64 megabytes

En la clase de algoritmos y estructuras de datos, el profesor estaba explicando el tema de árboles y Gerardo (uno de sus estudiantes) lo interrumpió para preguntar que tipos de recorridos en árboles existían. Como faltaban 5 minutos para el final, el profesor decidió dejarles de tarea hacer un algoritmo que recorra un árbol por niveles.

Dado que ya no tenía mucho tiempo para explicar todos los detalles dibujó en la pizarra el siguiente árbol:



Tu deber es recorrer el árbol y mostrarlo por niveles, es decir, primero la raiz (nodo 4), luego sus hijos (nodos 3, 6, y 0), después los hijos de estos nodos que se encuentran en el siguiente nivel (nodos 1, 7, 9, 2, 8, y 11), y así sucesivamente. Para el ejemplo de la figura anterior, la respuesta que se busca sería la siguiente:

```
4
3 6 0
1 7 9 2 8 11
5 10
```

Pero dependiendo de como se arme el árbol se pueden tener varias respuestas válidas. Para nuestro ejemplo, otra respuesta válida puede ser:

```
4
6 0 3
2 8 11 7 9 1
10 5
```

Un árbol $T = (V, E)$ está conformado por un conjunto de vértices $V = \{0, 2, \ldots, n-1\}$, y un conjunto de aristas $E = \{(p, q)/p, q \in V\}$ con la condición de que existe un nodo raíz $r \in V$, y que hay $n - 1$ aristas. Cada elemento $(p, q) \in E$, indicará que existe arista desde el nodo p hacia el nodo q. Para el ejemplo de la figura anterior, $V = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11\}$, $E = \{(4, 3), (4, 6), (4, 0), (3, 1), (3, 7), (3, 9), (6, 2), (6, 8), (0, 11), (1, 5), (1, 10)\}$, el nodo 4 es el nodo raíz y el árbol tiene 4 niveles.

Input

El problema contiene varios casos de prueba. La primera línea es un entero T ($1 \leq T \leq 10^2$) que denota el número de casos de prueba. Cada caso está compuesto por dos líneas, en la primera de ellas se encuentra

el número entero N indicando la cantidad de nodos que tendŕa el grafo ($1 \leq N \leq 10^3$). Luego, en la segunda línea del caso de prueba se encuentran $N-1$ pares de números p, q ($0 \leq p, q < n$, $p \neq q$) que representan las aristas del árbol en cuestión. El nodo raíz es determinado como aquel nodo que no tiene aristas entrantes, màs sí aristas salientes.

Output

Para cada caso de prueba, el programa deberá imprimir "`Caso #i:`" (sin comillas) y a partir de la siguiente linea el árbol por niveles. Como pueden haber varios recorridos por niveles válidos para un mismo árbol, imprimir cualquier recorrido válido.

Example

Standard input	Standard output
3 2 0 1 5 0 1 1 2 2 3 3 4 12 4 3 3 1 6 2 0 11 3 9 4 0 1 5 6 8 1 10 3 7 4 6	Caso #1: 0 1 Caso #2: 0 1 2 3 4 Caso #3: 4 3 6 0 1 7 9 2 8 11 5 10

Problem D. El viaje de Adam

Input file:	Standard input
Output file:	Standard output
Time limit:	1 second
Memory limit:	64 megabytes

Adam es una persona a quien le gusta conocer nuevos lugares, actualmente él se encuentra en la ciudad 1 y quiere llegar a la ciudad de $n+1$ pasando por n sitios durante su recorrido. Adam sabe que para ir de la ciudad i a la ciudad $i+1$ existen a_i formas de llegar. Adam desea saber el número de caminos distintos de llegar a la ciudad $n+1$ desde la ciudad 1. Sin embargo Adam solo desea saber el primer y último dígito del número de caminos posibles, ya que el número de caminos puede ser un número muy grande.

Input

El problema contiene varios casos de prueba. La primera línea es un entero T ($1 \leq T \leq 10^3$) que denota el número de casos de prueba. Cada caso está compuesto por dos líneas, en la primera de ellas se encuentran el entero n, el cual denora el número de ciudades que adam debe recorrer ($1 \leq n \leq 10^3$). Y en la segunda línea del caso de prueba se encuentran n enteros distintos $a_1, a_2, \ldots, a_n$ que representan el número de caminos posibles ($1 \leq a_i \leq 10^9$) para ir de la ciudad i a la ciudad $i+1$.

Output

Para cada caso de prueba, el programa deberá imprimir el primer y último dígito de número de caminos posibles para ir de la ciudad 1 a la ciudad n.

Example

Standard input	Standard output
3 1 10 3 3 5 7 2 123 456	1 0 1 5 5 8

Problem E. Contador de direcciones IP

Input file:	`Standard input`
Output file:	`Standard output`
Time limit:	1 second
Memory limit:	64 megabytes

Xavier, un programador muy hábil fue contratado por una gran compania de software llamada TORO Out of Control Technologies (TORO-OTC). Reciéntemente, había surgido un problema masivo con las direcciones IP de todas las computadoras de la compañía (aprox. 1000 equipos) y necesitaban con urgencia poder saber la cantidad de direcciones IP que seguían un determinado patrón.

Muy brevemente, una direccion IP (la empresa sólo trabaja con la versión 4 por el momento) es una etiqueta asignada a un dispositivo dentro de una red y está representado por un número de 32 bits. Dicho número, para ser fácilmente leible por nosotros, los humanos, es presentado como 4 números decimales (que varían de 0 a 255) separados por un punto. Por ejemplo, algunas direcciones IP son:

```
192.168.1.1   192.168.0.1   172.0.0.1
192.168.1.13  192.168.0.15  172.0.0.15
192.168.1.14  192.168.0.16  172.0.0.16
192.168.1.15  192.168.0.17  172.0.0.17
```

El problema que TORO-OTC tenía yace básicamente en saber cuantas direcciones fueron asginadas siguiendo un determinado patrón. Un patrón era una dirección IP que podía contener asteriscos ('*') los cuales reemplazaban números (entre 0 y 9), pero también un asterisco puede reemplazar números en el rango de 0-255 siempre y cuando no haya ningún otro número que condicione la naturaleza del asterisco — para mayor claridad, consultar los ejemplos. Considerando las direcciones IP proporcionadas, la Tabla 1 muestra algunos ejemplos del tipo de consultas que Xavier debería de responder.

Table 1

Patrón	Número de direcciones
`192.168.1.*`	4
`192.168.*.15`	2
`192.*.*.*`	8
`192.168.0.14`	0
`192.168.1.1`	1
`1*2.*.0.1*`	6
`1*2.**8.*.*`	8
`1*2.*8.*.*`	0
`*.*.*.*`	12

Input

El problema contiene varios casos de prueba. Las primera línea es un entero T ($1 \leq T \leq 100$) que denota el número de casos de prueba. Cada caso inicia con una línea "`Inicio`" y termina con una lína "`Fin`" (sin comillas) y está compuesto por una lista de consultas en medio. Las consultas pueden ser de dos tipos. "`Agregar IP`", donde IP tendrá el formato de una IP válida antes expuesta, y "`Contar PATRON`" que define el patrón sobre el cual se debe de contar las IPs almacenadas hasta el momento. Se garantiza que no se agregarán IPs repetidas para un mismo caso de prueba.

Output

Para cada caso de prueba, el programa deberá imprimir "`Caso #i:`" en una nueva línea (sin comillas) y a continuaciíon las respuestas a las consultas del tipo `Contar`. Cada respuesta debe de ir en una línea.

Example

Standard input	Standard output
2 Inicio Contar 192.168.1.1 Agregar 192.168.1.1 Agregar 192.168.1.10 Agregar 192.158.1.1 Contar 192.168.1.* Fin Inicio Agregar 172.0.0.1 Agregar 192.0.0.1 Contar 1*2.*.*.1 Contar *.*.*.* Fin	Caso #1: 0 2 Caso #2: 2 2

Problem F. Un viaje de ensueño

Input file:	Standard input
Output file:	Standard output
Time limit:	1 second
Memory limit:	64 megabytes

Warren y sus amigos piensan ir de paseo hacia Arequipa, para celebrar sus buenas notas en todos los cursos. Ellos contrataron un omnibus interprovincial para realizar el viaje, hicieron el contrato para viajar a las $5:00$ am y todos debian estar puntuales.

Llegado el día del viaje Warren se quedó dormido así que se apresuro en alistarse, y como es una persona ahorradora decidió irse en combi pues confiaba que llegaría a tiempo. Cuando llegó terminal terrestre, el omnibus al que debía embarcar ya había salido y se encontraba a una distancia d del terminal (considerar que todo el trayecto es una línea recta). Sin embargo, aún tiene la opción de realizar el viaje con sus amigos tomando un taxi e intentar dar alcance al omnibus. El taxi va a una velocidad constante de v_t y Waren quiere saber si es posible alcanzar al omnibus que tiene una velocidad constante de v_o, y si fuera así en cuanto tiempo podría alcanzar al bus.

Ayuda a Warren con su dilema.

Input

El problema contiene varios casos de prueba. La primera línea es un entero T ($1 \leq T \leq 10^4$) que denota el número de casos de prueba. Cada caso está compuesto una línea con los números v_t, v_o y d ($0 \leq v_t, v_o, d$); siendo v_t la velocidad del taxi, v_o la velocidad del omnibus y d la distancia del omnibus al terminal terrestre.

Output

Para cada caso de prueba, el programa deberá imprimir el tiempo en el cual el taxi puede alcanzar al omnibus con 4 dígitos de precisión, si fuese imposible entonces deberá imprimir -1.

Example

Standard input	Standard output
2 2 2 2 5 3 11	-1 5.5

Problem G. Warrencio y los Bicliques

Input file:	Standard input
Output file:	Standard output
Time limit:	1 second
Memory limit:	64 megabytes

El profesor de algoritmos está a punto de desaprobar a warrencio pero warrencio en la desesperación le propone hacer lo que sea para que pueda aprobar es así que el profesor le pide una tarea titánica el cual consiste en dado un grafo no dirigido hallar el biclique con más aristas dentro del grafo; al ver la cara entristecida de warren el profesor se apiada y le da una ayuda, el profesor le da el primer conjunto del biclique, ahora la tarea se resumen en dado un conjunto S_1 encontrar el conjunto S_2, se asegura que siempre habrá un S_2 de tamaño mayor o igual a 1.

nota:

un biclique es un grafo bipartido completo, es decir cada elemento de conjunto S1 tiene una aristas a cada elemento de S_2, (un grafo bipartito es un grafo cuyos vértices se pueden separar en dos conjuntos disjuntos).

Input

la entrada consta de T casos de entrada, cada caso sigue el siguiente formato de entrada: la primera línea consta de dos enteros n,m. n es el número de vértices del grafo no dirigido y m el número de aristas las siguientes m líneas contienen dos enteros u y v que representan una arista entre el nodo u y el nodo v seguido de una un entero K que representa el tamaño de S_1 finalmente la última línea de cada caso consta de K enteros.

limitaciones

$1 \leq n, m, k \leq 500; 0 \leq u, v < n$

Output

Para cada caso de prueba, el programa deberá imprimir el conjunto S_2.

Example

Standard input	Standard output
1 4 1 4 2 1 3 2 3 6 5 4 10	1 3

Problem H. Horarios

Input file:	`Standard input`
Output file:	`Standard output`
Time limit:	1 second
Memory limit:	64 megabytes

En la carrera profesional de Ingeniería Informática y de Sistemas muchos alumnos llevan cursos en diferentes horarios. Algunos cursos pueden ser en la mañana, tarde o noche y no siempre se tiene un horario consecutivo.

Warnoldo, es un estudiante de la carrera que estaba preocupado en determinar cuanto es el máximo intérvalo libre y el máximo intervalo consecutivo de cursos que tiene. No es difícil averiguar para una sóla persona dicha información, pero muchos estudiantes al enterarse de la idea de Warnoldo también sienten curiosidad para determinar dichos valores. Por tanto, ahora se necesita diseñar un programa que realice la tarea mencionada.

Dados N cursos y el intérvalo de tiempo (en minutos) de cada curso, tu tarea es determinar el máximo intérvalo libre entre cursos(t_{libre}) y el máximo intérvalo donde estarás ocupado en los cursos($t_{ocupado}$). Por conveniencia, todos los cursos comienzan a las 7am, los minutos se contarán a partir de las 7am y por tanto, la información de cada curso tendrá un `tiempo_inicio` (t_o) y `tiempo_fin` (t_f) ambos en minutos. Si un curso comienza a las 7am y termina a las 9am, t_o será igual a 0 y t_f igual a 120.

Existe la posibilidad de que los horarios para algunos cursos se sobrepongan.

Input

El problema contiene varios casos de prueba. La primera línea es un entero T ($1 \leq T \leq 10^2$) que denota el número de casos de prueba. Cada caso está compuesto por una línea que contiene un número entero N ($0 < N \leq 10^2$) indicando la cantidad de cursos. Seguidamente, N lineas serán presentadas conteniendo dos enteros t_o y t_f ($0 \leq t_o < t_f$).

Output

Para cada caso de prueba, el programa deberá imprimir una línea mostrando los valores de t_{libre} y $t_{ocupado}$ por un espacio en blanco.

Example

Standard input	Standard output
2 2 0 120 240 360 3 300 1000 700 1200 1500 2100	120 120 300 900

Problem I. El Baile de Invierno

Input file:	`Standard input`
Output file:	`Standard output`
Time limit:	1 second
Memory limit:	64 megabytes

Hogwarts fue escogida como sede de la mayor competencia entre magos del mundo, el Torneo de los Tres Magos, que envolverá las tres escuelas de magia más famosas de Europa: Hogwarts, Beauxbatons y Durmstrang. Este torneo funciona de la siguiente forma. Inicialmente, un representante (llamado campeón) de cada escuela es seleccionado a través del Cáliz de Fuego. Después, los campeones realizan, a lo largo del año, tres tareas diferentes. Se consagra de vencedor del Torneo de los Tres Magos el campeón que acumula más puntos en las tres tareas.

Parte tradicional del Torneo de los Tres Magos, el Baile de Invierno se llevará a cabo en el Gran Salón, en el día de Navidad. La profesora Minerva McGonagall fue designada como responsable por las aulas de baile de los alumnos de Hogwarts. A fin de prepararlos para el Baile de Invierno, después del término de aulas muchas parejas fueron formadas voluntariamente. Sin embargo algunos alumnos permanecieron sin pareja. Por esta razón, la profesora Minerva decidió formar parejas por cuenta propia.

Ella cree que la mejor forma de emparejar es unir un hombre y una mujer que tengan, como máximo, una pequeña diferencia entre sus habilidades de baile. Dado que la profesora Minerva conoce la habilidad de baile (representado por número entero entre 1 y 100, tal que el número 1 se refiere a un pésimo bailarín y el número 100 se refiere a un bailarín perfecto) de cada uno de sus alumnos, ella intentará formar el mayor número de parejas tales que, para cada pareja, la diferencia de habilidades de baile entre el hombre y la mujer sea menor o igual a un número entero D. Las parejas formadas deste modo son llamadas "parejas equilibradas".

Tu tarea es escribir un programa que, dadas las habilidades de baile de los alumnos sin pareja y la diferencia máxima de habilidades permitida entre los integrantes de una pareja, determinar el mayor número de parejas equilibradas que pueden ser formadas.

Input

El problema contiene varios casos de prueba. La primera línea es un entero T ($1 \leq T \leq 100$) que denota el número de casos de prueba. Cada caso está compuesto por tres lineas, donde la primera contiene tres números enteros M, N, y D ($1 \leq M, N \leq 10^5, 0 \leq D \leq 99$) tales que M representa el número de hombres, N el número de mujeres, y D la diferencia máxima de habilidades permitida entre cada integrante de una pareja. La segunda línea contiene M números enteros $X_1, X_2, \ldots, X_M$ ($1 \leq X_i \leq 100$) que representan las habilidades de los hombres. La tercera línea contiene N números enteros $Y_1, Y_2, \ldots, Y_N$ ($1 \leq Y_i \leq 100$) que representan las habilidades de las mujeres.

Output

Para cada caso de prueba, el programa deberá imprimir una línea con el formato "`# maximo de parejas equilibradas =` X", donde X indica el mayor número de parejas equilibradas que pueden ser formadas.

Example

Standard input	Standard output
2 2 3 1 3 6 2 2 3 6 5 4 10	# maximo de parejas equilibradas = 2 # maximo de parejas equilibradas = 0