Universidad Nacional de San Antonio Abad del Cusco
Facultad de Ingenieria Electrica, Electronica, Informatica y Mecanica
Escuela Profesional de Ingenieria Informatica y de Sistemas

Concurso de Programación
VIII CusContest

PROBLEM SET

Este conjunto de problemas contiene X problemas,
páginas numeradas de 1 a Y.

Organizado por:
ACM Chapter Cusco
EPIIS-UNSAAC

Julio X, 2016
CUSCO - PERÚ

INFORMACIÓN GENERAL

Salvo se indique lo contrario, lo siguiente vale para todos los problemas.

Entrada

- La entrada se debe leer de la entrada estándar (*standard input*).
- Cuando una línea de datos contiene varios valores, estos se separan utilizando exáctamente un espacio entre ellos. Ningún otro espacio aparece en la entrada.
- No hay letras con tildes, acentos, diéresis, ni otros signos ortográficos (ñ, Ã, ç, ú, æ, etcétera).

Salida

- La salida se debe escribir en la salida estandar (*standard output*).
- El resultado del caso de prueba debe de aparecer en la salida utilizando una cantidad de líneas que depende del problema. No debe de haber otros datos en la salida.
- Cuando una línea de resultados contiene varios valores, estos se deben de separar utilizando exáctamente un espacio entre ellos. Ningún otro espacio debe de aparecer en la salida.
- No debe de haber letras con tildes, acentos, diéresis, ni otros signos ortográficos (ñ, Ã, ç, ú, æ, etcétera).

Tiempo Límite

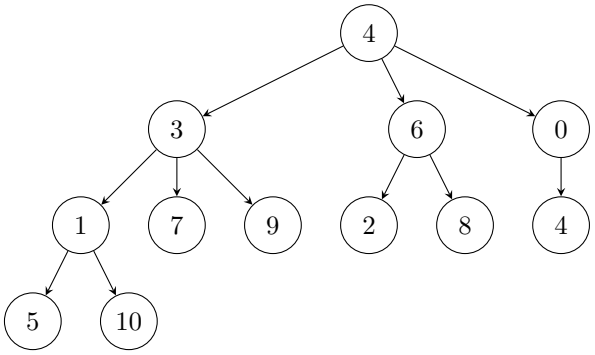
- El tiempo límite informado corresponde al tiempo total permitido para la ejecución de una solución sobre todo el conjunot de casos de prueba de un determinado problema.

Equipo de desarrollo:
Alexander Pinares, Federal University of Rio Grande do Sul
Berthin Torres, University of Campinas
Grover Castro, University of São Paulo
Javier Vargas, University of Campinas

Problem A. Mostrando grafos

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

Dado un árbol $T = (V, E)$ como un conjunto de vértices $V = \{1, 2, \ldots, n\}$, 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. Por conveniencia, se considerará un grafo dirigido, y para cada elemento $(p, q) \in E$, indicará que existe arista desde p hacia q. Por ejemplo, la siguiente figura muestra un árbol de con 12 nodos.



Tu deber es recorrer el grafo y mostrarlo por niveles, es decir, primero la raiz (nodo 4), luego sus hijos (nodos 3, 6, y 0), luego los hijos de estos nodos q se encuentran en el siguiente nivel (nodos 1, 7, 9, 2, 8, y 4), 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 4
5 10
```

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 grafo por niveles.

Example

Standard input	Standard output
2 2 0 1 5 0 1 1 2 2 3 3 4	Caso #1: 0 1 Caso #2: 0 1 2 3 4

Problem B. 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.*.*.1*`	6
`1*2.*.*.*`	8
`*.*.*.*`	12

Input

El problema contiene varios casos de prueba. Las primera línea es un entero T ($1 \leq T \leq T$) 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. 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 C. Nombre del Problema

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

Descripción del problema...

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 encuentran los números enteros N y K, siendo N el número de piedras medidas ($1 \leq N \leq 10^5$, $1 \leq K \leq 10^9$). Luego, en la segunda línea del caso de prueba se encuentran N enteros distintos $a_1, a_2, \ldots, a_N$ que representan las alturas de las piedras ($1 \leq a_i \leq 10^9$).

Output

Para cada caso de prueba, el programa deberá imprimir la longitud del mayor conjunto libre de K-múltiplos que se pueda obtener de la lista de alturas.

Example

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