Problema A

Invertir

Definiremos una función **F**(n) que devuelve la inversa de n por ejemplo **F**(120)=21

Dado dos números a y b devolver **F**(**F**(a)+**F**(b))

Input

La primera línea del INPUT comienza con un número M indicando el número de casos (1<=M<=100), M líneas le siguen.

- Cada línea tendrá 2 números enteros indicando a y b.
- a y b se encontrara entre 1 y 5000000.

Output

Deberá formatear su salida de la forma "Caso #N: R:"

(Comillas para claridad) donde N es el número de caso y R es el resultado.

Input	**Output**
4 10 11 5 6 32 60 1 1	Caso #1: 21 Caso #2: 11 Caso #3: 92 Caso #4: 2

- Para el primer Caso F(10)=1 , F(11)=11 por tanto F(12)=21

Problema B

CSS-JavaScript

En los últimos años, CSS y JavaScript se han convertido en las herramientas más utilizadas para la construcción de páginas webs. Dado su gran importancia se han elaborado muchos frameworks que hacen mucho uso de estas tecnologías, tales como JQuery, ExtJS, HTML5, por mencionar algunas.

En esta ocasión nuestra tarea es simple; consta de tratar de pasar nuestras variables de CSS a JavaScript. Tenemos en cuenta que CSS utiliza el carácter '-' para separar dos palabras y JavaScript utiliza la notación Camel que consiste en convertir la primera letra de todas las palabras, con excepción de la primera a mayúscula. Por ejemplo, en CSS para margin-left su equivalente en JavaScript sería marginLeft.

Input

La primera línea del INPUT comienza con un número M indicando el número de casos (1<=M<=100), M líneas le siguen.

- Cada línea tendrá una cadena que podría estar con una longitud de 1 a 50.
- La cadena contendrá ['a'-'z'] y '-' de caracteres.
- Una clase en CSS no puede comenzar con '-'

Output

Deberá formatear su salida de la forma "Caso #N: R".

(Comillas para claridad) donde N es el número de caso y R es el resultado en JavaScript.

Input	Output
4 font-style letter-spacing-border body border-radius-top	Caso #1: fontStyle Caso #2: letterSpacingBorder Caso #3: body Caso #4: borderRadiusTop

Problema C

TeamSplit

La gente salió a la cancha para comenzar la pichanga. Como de costumbre, se comienza la selección de la gente para los equipos. Capitán A y B eligen siempre los más fuertes que aún quedan libres.

Durante cada turno, un capitán selecciona un persona para su equipo, así para al final hacer su equipo lo más fuerte posible. Las personas a escoger tienen fuerzas que te son entregados en una lista de números enteros. Donde el mayor significa el más fuerte.

Por ejemplo: si las fuerzas de las personas serian {5,7,8,4,2}

Turno	Capitán	Persona	Fuerza
1	A	3ra	8
2	B	2da	7
3	A	1ra	5
4	B	4ta	4
5	A	5ta	2

Devolver la diferencia absoluta de fuerza entre ambos equipos. Del ejemplo anterior seria 4=15-11

Input

La primera línea del INPUT comienza con un número M indicando el número de casos (1<=M<=100), M líneas le siguen.

- Cada línea tendrá una lista de fuerzas enteras entre 1 y 50 elementos
- Cada elemento de fuerza estará entre 1 y 1000 inclusive.

Output

Deberá formatear su salida de la forma "Caso #N: R:"

(Comillas para claridad) donde N es el número de caso y R es el resultado.

Input	Output
4 5 7 8 4 2 100 1000 1000 1 5 10 1 5 10	Caso #1: 4 Caso #2: 100 Caso #3: 0 Caso #4: 0

- Para el segundo caso, solo se le asigna una persona al primer equipo dejando al otro con 0 de fuerza.

Problema D

Programación Paralela

Tú estas programando un planificador de procesos para un sistema operativo que es controlado por muchos procesadores. El paralelismo alcanzado por este nuevo sistema operativo es total. Lo que significa que cualquier número de procesos puede ser ejecutado al mismo tiempo sin molestar a los demás. Algunos procesos necesitan de otros procesos que hayan finalizado para iniciar este.

Se le brindara **tiempo** de cada proceso como int[], donde **tiempo**[i] es el número de milisegundos que necesitara para ser finalizada el proceso i. La información acerca de precedencia es dado como String[i] **prec**, donde el jth carácter del ith elemento si es 'Y' el proceso i necesita ser ejecutado para iniciar el proceso j y 'N' en caso contrario. Devolver el menor tiempo para ser ejecutados todos los procesos ó -1 si es imposible.

Input

La primera línea del INPUT comienza con un número M indicando el número de casos (1<=M<=100), M casos le siguen.

- La primera línea del caso se le dará N, N+1 líneas le siguen.
- La segunda línea será **tiempo** como una lista de N enteros separados por un espacio.
- La tercera línea del caso en adelante será **prec** como N líneas que describen la precedencia de los procesos.
- **Tiempo** y **prec** tienen el mismo número de elementos.
- En la primera línea de cada caso tendrá una lista de tiempos de ejecución
- **tiempo** tendrá entre 1 y 50 elementos.
- Cada elemento de tiempo estará comprendido entre 1 y 1000
-

Output

Deberá formatear su salida de la forma "Caso #N: R:"

(Comillas para claridad) donde N es el número de caso y R es el resultado.

Input	Output
5 3 150 200 250 NNN NNN NNN 3 150 200 250 NNN YNN NNN 3 150,200,250 NYN NNY YNN 3 150 200 250 NYN NNY NNN 6 345 335 325 350 321 620 NNNNNN NNYYYY YNNNNN NNYNYN NNNNNN NNNNNN	Caso #1: 250 Caso #2: 350 Caso #3: -1 Caso #4: 600 Caso #5: 1355

- Primer caso : No existe dependencia , por tanto se pueden ejecutar al mismo tiempo , tomándose un tiempo del mayor 250
- Segundo caso: El primer proceso debe esperar terminar al segundo proceso para iniciarse. Tomándose un tiempo de 350, el tercer proceso fácilmente puede ser ejecutado de manera paralela en ese tiempo.
- Tercer caso: Ningún proceso puede iniciarse porque todos los procesos esperan que terminen otros para iniciarse.
- Cuarto caso: Aquí no existe paralelismo por sus dependencias , ejecutándose en este orden 1-2-3