

LABORATORIO DE INVESTIGACIÓN DE SOFTWARE

---

10º Competencia de programación  
2018

---

**liS** LABORATORIO  
DE INVESTIGACIÓN  
DE SOFTWARE

**UTN** FACULTAD  
REGIONAL  
CÓRDOBA  
INGENIERÍA EN SISTEMAS DE INFORMACIÓN

## Reglas de la Competencia

- Los integrantes de los equipos no pueden utilizar ningún medio de almacenamiento (pendrives, USB, etc.).
- No está permitido el uso de celulares, notebook, ni ningún otro dispositivo móvil.
- No está permitido el uso de internet.
- El equipo que viole cualquiera de estas condiciones quedará automáticamente descalificado.
- No se responderán consultas sobre los problemas o la programación de las soluciones en forma oral. El único canal para dichas consultas es mediante el envío de una solicitud de clarificación desde el software PC<sup>2</sup>. Sólo se responderán las clarificaciones enviadas hasta las 21:00.



## Problema 1: La mudanza de Juan

Juan es un muchacho que se está por mudar a una ciudad nueva. Esta ciudad tiene la particularidad de que sus calles forman una grilla perfecta, es decir que las calles son paralelas entre sí y que los ángulos de las intersecciones son siempre rectos.

A Juan no le gusta caminar mucho desde su casa para llegar a cada uno de los comercios a los que habitualmente concurre, así que está buscando una casa que esté a menos de 3 manzanas de todos esos comercios.

Para ello relevó todos los comercios del barrio al que se va a mudar, indicando en qué manzana se encuentran. Asignó los números 1 al 5 para cada tipo de comercio con la siguiente codificación: 1- Almacén, 2- Supermercado, 3- Rotisería, 4- Farmacia, 5- Librería. Luego anotó en un plano la lista de comercios que existen en cada manzana.

Por ejemplo, si el barrio tiene 6 manzanas de ancho y 6 de largo, el plano puede haberle quedado de la siguiente manera:

	1	12		135	
		4	3		
		1			12345

Si Juan no quiere caminar más de 3 manzanas por cada comercio, sólo podría mudarse a las manzanas marcadas:

	1	12		135	
		4	3		
		1			12345

### Entrada

El programa primero recibe una línea con un número entero indicando la cantidad de casos de prueba. A continuación se reciben C casos, por cada uno de ellos el programa debe recibir el tamaño del barrio, indicado por dos números enteros (M y N) con el ancho y el largo del mismo ( $0 < M, N < 100$ ). Luego se ingresan N líneas cada una de ellas con M números.

Los números pueden ser 0 si la manzana no tiene comercios o un número con algunos de los dígitos 1 a 5 indicando los comercios disponibles en esa manzana.

Luego se indica la cantidad máxima de manzanas que Juan está dispuesto a caminar.

### Salida

El programa debe imprimir por cada caso únicamente la cantidad de manzanas a las cuales Juan podría mudarse para tener cerca al menos un comercio de cada uno de

los cinco tipos. Es decir, una manzana debe contarse sólo si se puede llegar desde ella a todos los tipos de comercios en la distancia especificada.

### Ejemplo

Entrada

Salida

$[M, U]$   $\begin{bmatrix} m_1 & m_2 \\ m_3 & m_4 \\ m_5 & m_6 \end{bmatrix}$   $\begin{matrix} C \\ J \end{matrix}$

2
6 6
0 0 0 0 0 0
0 1 12 0 135 0
0 0 0 0 0 0
0 0 4 3 0 0
0 0 1 0 0 12345
0 0 0 0 0 0
3
4 4
0 0 0 0
0 0 0 0
0 123 0 0
0 0 0 0
2

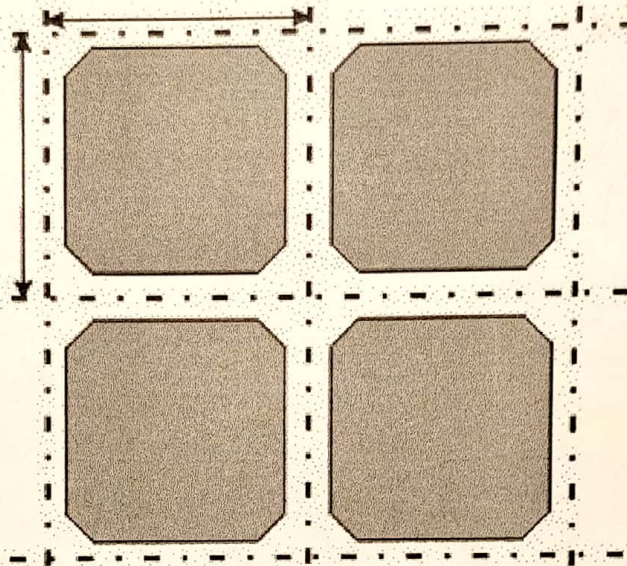
19  
0

11234



## Problema 2: PolilaDrone

La policía de la ciudad de Damerolandia, ha comprado un drone que detecta la distancia de los ladrones y los policías en las persecuciones. La ciudad de Damerolandia tiene ese nombre debido a su característica principal que son sus calles ortogonales y de cuadras de 100 metros cada una.



Para simplificar el problema se ha tomado como referencia el centro de la ciudad y se le entregaron las coordenadas (0,0).

La idea del dron es enviar a la central de la policía cual es la distancia entre el ladrón y los móviles policiales en un momento determinado para poder ayudarlos a ubicar en línea recta quién está más cerca.

Lo que tiene como entrada el dron son los movimientos en cuadras y por direcciones que vienen dados por cámaras que hay en las calles, por lo que es necesario calcular la posición final de cada uno y determinar quién está más cerca en línea recta del ladrón.

### Entrada

La entrada inicia con un número entero  $C$  indicando la cantidad de casos de prueba. Cada uno de los casos incluye las siguientes entradas:

- La primera línea contiene un entero ( $0 < M < 5$ ) sobre cuántos móviles policiales participan de la persecución.
- La siguiente línea contiene, en primer lugar las coordenadas de arranque (x,y) del ladrón y los movimientos del ladrón que fueron rastreados.
- Las siguientes líneas contienen las coordenadas de arranque (x,y) de cada uno de los  $M$  móviles policiales y los movimientos de los mismos.



- Los movimientos vienen determinados por unidades de cuadras expresadas como un número entero y la dirección dada por puntos cardinales E, O, N, S. Ejemplos: 1S, 2N, 23O, etc.

## Salida

La salida es un conjunto de números separados por espacios de las distancias en metros en línea recta desde cada uno de los móviles policiales hasta el ladrón. Las distancias deben estar expresadas en metros redondeadas al entero más cercano. Si hay más de un móvil las distancias deben presentarse en el mismo orden en que fueron ingresados los movimientos de los móviles.

En otra línea se debe mostrar el número del móvil que esté más cerca del ladrón.

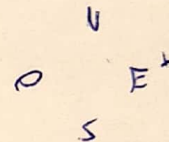
## Ejemplo

### Entrada

C	3
M	2
	0,0 2N 3S 20 1E
	1,1 2E 40 1N
	2,3 4E 2S
M	2
	0,0 6S 8E
	7,6 2N 9S 6E 80
	3,4 9N 8S 3E 100
M	2
	0,10 2N 9S 10E 60
	7,10 7N 9E 30
	3,2 9N 3S 9E 10

### Salida

300 728  
1  
583 1628  
1  
1664 860  
2



X 0,0 0,0 0,200 0,-100 -200,-100 [-100,-100]

~~0,200~~ ~~100,300~~ ~~200,300~~

0<sub>1</sub> 100,100 300,100 -100,100 [-100,200]  
0<sub>2</sub> 200,300 600,300 [600,100]

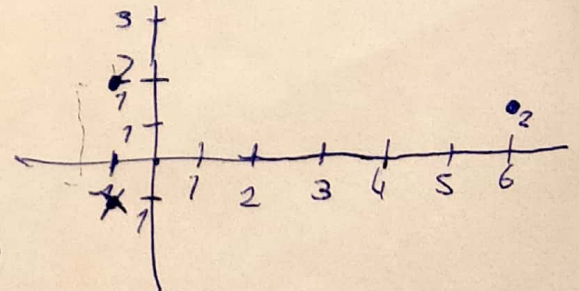
$$h = \sqrt{x^2 + y^2}$$

$$0^2 + 300^2$$

$\sqrt{200^2 + 100^2}$

0 100

$$0_1 [(-100 + 100), (-100 - 200)]$$



$$0_2 [(-700 - 600), (-100 - 100)] \quad h = \sqrt{700^2 + 200^2}$$



### Problema 3: La Calesita

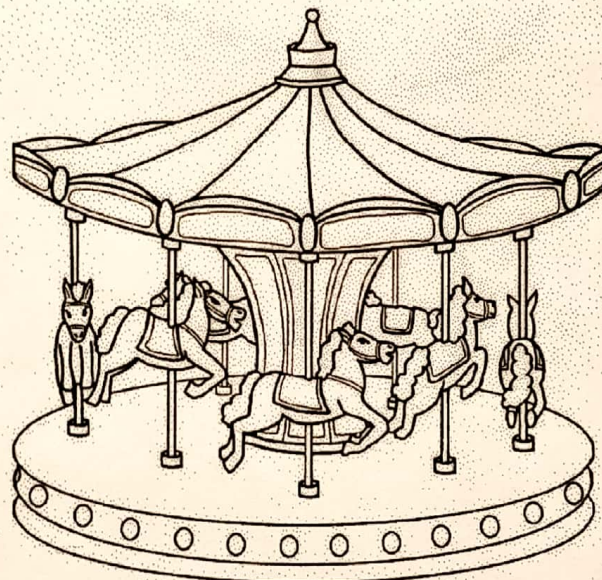
Se sabe que en la plaza de Alta Córdoba hay una calesita que tiene la siguiente particularidad: permite a los niños subirse y bajarse mientras la calesita está funcionando y mientras dure el tema musical, siempre que los padres les ayuden en dicha tarea y bajo su propia responsabilidad.

La calesita funciona mientras dura un tema musical, cuando el tema musical se termina la calesita se para completamente, y algunas veces quedan niños a la espera sin poder subir a la misma.

Entonces se nos solicita un programa que sea capaz de brindarnos información acerca del estado de la calesita al terminar la duración de un cierto tema musical.

Al programa se le informará el tiempo que cada niño se va a estar en la calesita, como así también la cantidad de niños que tienen intenciones de subir y cuanto tiempo desean estar en la misma, lo que llamaremos el “deseo de calesita”.

Vamos a resolver el problema para una calesita genérica.



© Colorpk.be

#### Entrada

Inicialmente se lee un número entero  $T$  indicando la cantidad de casos de prueba. En la línea siguiente viene un número entero  $D$  indicando la duración del tema musical.

En la próxima línea vendrá un número  $N$  indicando la capacidad de la calesita que siempre comienza con su capacidad máxima, es decir que la cantidad de niños con los que empieza a funcionar la calesita siempre coincide con su capacidad. En la siguiente línea  $n$ -números con el tiempo que cada niño se quiere quedar en la misma.

A continuación, vendrá un número  $K \geq 0$  indicando la cantidad de niños en una lista de espera para de subir a la calesita, y en la siguiente línea vendrán  $k$ -números indicando el tiempo que cada uno de esos niños quiere permanecer en la calesita (“deseo de calesita”). Si no hay niños en espera la línea vendrá en blanco.



## Salida

Por cada caso de prueba, al terminar una canción puede ocurrir que no hayan quedado niños en la calesita en cuyo caso debe imprimir "sin niños" (respectando las minúsculas y sin las comillas dobles).

En el caso de que al finalizar el tema hayan quedado niños subidos en la calesita, se debe imprimir un par ordenado de números enteros  $(x,y)$ , donde  $x$  indica la cantidad de niños que se quedaron con  $y$ -minutos de "deseo de calesita".

Cada par debe ir separado por una coma en la misma línea, tal como lo muestra el ejemplo del caso de prueba. Es decir, no se debe dejar ningún espacio (ni entre los elementos del par ordenado, ni entre los pares ordenados), y los pares deben mostrarse ordenados en base a su segunda componente. Esto debe repetirse para cada caso de prueba.

## Ejemplo

### Entrada

T 1  
D 5  
N 7 - Siempre empieza llena  
[n] 4 2 4 7 12 2 5 + tiempo que quieren  
K 4 quieren subir  
[k] 12 3 6 8

### Salida

(1,2), (1,5), (2,7), (1,9)  
cuantos le quedaron de andar  
cantar niños, min



## Problema 4: Evaluador de ecuaciones

Se nos contrató como ingenieros para que creemos un programa que verifique las ecuaciones matemáticas que son generadas de manera aleatoria por el programa secreto MATRIX.

Es decir nuestro programa es un corrector de las ecuaciones matemáticas, por lo que debe detectar inconsistencias en las ecuaciones de entrada en lo que respecta a las reglas de paréntesis, llaves y corchetes.

Esto es lo que determina el orden de evaluación de las operaciones algebraicas. Es por ello que es muy importante que estén bien definidos y que se vea si se está forzando el orden de evaluación de alguna operación.

El gran problema de todo esto es que el generador de ecuaciones secretas MATRIX nos devuelve las ecuaciones matemáticas en donde los paréntesis, corchetes, llaves y otros símbolos están representados por números que comienzan con 0.

El listado de símbolos de las ecuaciones viene dado por:

Símbolo	Valor
(	01
)	02
[	03
]	04
{	05
}	06
+	07
-	08
×	09
/	010
^	011

Se nos pide entender la ecuación de entrada y determinar:

- Si es una ecuación válida (no le faltan componentes)
- La ecuación expresada en símbolos aritméticos

### Entrada

La entrada es un número que determina la cantidad de ecuaciones y un array de números separados por espacios que expresa la ecuación por cada línea que sigue. En el array vienen incluidos los operadores con su código.

### Salida

Por cada ecuación de la entrada debe haber 2 líneas de salida, una que determine si la ecuación es VALIDA o tiene ERROR y la otra debe mostrar la ecuación en expresión aritmética sin espacios entre los símbolos.

**Ejemplo****Entrada**

2

03 01 5 08 16 08 5 09 2 04

05 03 9 07 01 5 010 2 02 04 08 9 06

03 01 5 08 02 01 02 04

**Salida**

ERROR

 $[(5-16-5 \times 2)]$ 

VALIDA

 $\{[9+(5/2)]-9\}$



## Problema 5: Mensajes secretos

Dos amigas necesitan enviarse por correo electrónico mensajes muy importantes y extremadamente reservados. Ante el temor de que alguien pueda leer los correos, deciden codificar los mensajes. La estrategia que utilizan es la siguiente: algunos fragmentos de texto los escriben en orden inverso y los encierran entre paréntesis, de manera tal de no olvidar que esos fragmentos deben ser leídos al revés.

Para facilitarles la tarea se les pide que escriban un programa que permita decodificar los mensajes enviados. El mensaje original a transmitir está formado por caracteres que pueden ser letras, números y/o signos de puntuación pero no paréntesis. De esta manera se evitan problemas en el momento del encriptado.

### Entrada

La entrada comienza con un número entero indicando la cantidad de mensajes  $C$  que hay que descryptar. Luego se ingresan  $C$  líneas, cada una con un mensaje con longitud máxima de 2000 caracteres. Los mensajes constan de letras, números, signos de puntuación y posiblemente paréntesis producto del encriptado.

### Salida

Por cada mensaje ingresado se deben imprimir dos líneas, la primera conteniendo el mensaje descryptado y la segunda con el fragmento más corto que está encerrado entre paréntesis.

### Ejemplo

#### Entrada


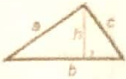
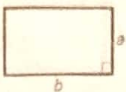
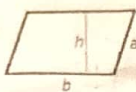
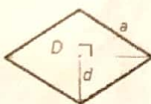
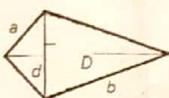
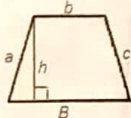
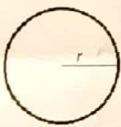
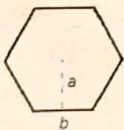
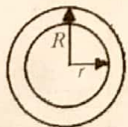
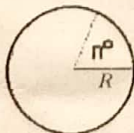
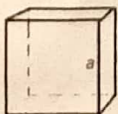

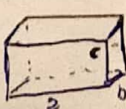
1

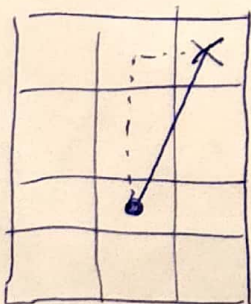
Hoy (.sh 22 sal a) (ed asac ne sominuer son) Marcelo.

#### Salida

Hoy a las 22 hs. nos reunimos en casa de Marcelo.  
a las 22 hs.

str[::-1]

A, área o superficie - P, perímetro - V, volumen		
Cuadrado		$A = a^2$ $P = 4 \cdot a$
Triángulo		$A = \frac{b \cdot h}{2}$ $P = a + b + c$
Rectángulo		$A = b \cdot a$ $P = 2 \cdot (b + a)$
Paralelogramo		$A = b \cdot h$ $P = 2 \cdot (b + a)$
Rombo		$A = \frac{D \cdot d}{2}$ $P = 4 \cdot a$
Cometa		$A = \frac{D \cdot d}{2}$ $P = 2 \cdot (b + a)$
Trapezio		$A = \frac{(B + b) \cdot h}{2}$ $P = B + b + a + c$
Círculo		$A = \pi \cdot r^2$ $P = 2 \cdot \pi \cdot r$
Polígono Regular		$A = \frac{P \cdot a}{2}$ $P = n \cdot b$ n. es el número de lados a. es la apotema
Corona Circular		$A = \pi \cdot (R^2 - r^2)$
Sector Circular		$A = \frac{\pi \cdot R^2 \cdot n}{360}$
Cubo		$A = 6 \cdot a^2$ $V = a^3$
Cilindro		$A = 2\pi \cdot R(h + R)$ $V = \pi \cdot R^2 \cdot h$
		$A = 2 \cdot (ab + ac + bc)$ $V = abc$

[illegible]



[41]

contador Par Izq  
" " Der  
" Corc Izq  
" " Der  
" lere Izq  
" " Der

puede haber más de un paréntesis?

$[(2+5)-(3+4)]$

puede haber corchetes/paréntesis/llaves  
sin contenido?  $[(1)2+4]$

Importa el orden?  $\{[(1)]\}$  válida?  
 $\{[(1)]\}$  válida?

duracion (entero)

n (capacidad - entero)

$[a, b, c, d]$

Calesita

- while que en cada pasada reste 1 a la duración ~~y~~
- ~~se puede~~ restarle ① a cada  $[n]$
- - chequear si algún  $[n]$  es 0 y quitarlo en ese caso
- - de haber lugar, mover un  $[k]$  a  $[n]$
- si todas en  $[n]=0$ , print "sin niños"
- else
- - arma vector con  $(x, y)$

↓      ↓  
cant    cant  
niños   minutos restantes

↓  
busca toda la ocurrencia de y minutos en el  
vector  $[n]$  y quitarla hasta que el mismo  
quede vacío