

Universidad Católica Boliviana “San Pablo”



Facultad de Ciencias Exactas e Ingeniería
Ingeniería de Sistemas

Primer concurso anual
Programador del año UCB 2009

SESIÓN DEL CONCURSO

9 de mayo de 2009
La Paz - Bolivia

Comité Organizador

La organización y producción de los problemas de este concurso fue realizada por docentes de Ingeniería de Sistemas de la Universidad Católica Boliviana “San Pablo”.

Los organizadores del concurso fueron:

- MSc. Irma Prado
Directora de Carrera
- MSc. Orlando Rivera
Docente de Introducción a la Programación, Programación I, Programación II y Sistemas de Información I
- MSc. Oswaldo Figueroa
Coordinador de los programas de las Maestrías de Sistemas de Información
- Ing. Edgar Villegas
Docente de Introducción a la Programación
- Ing. Mauricio Del Carpio
Docente de Introducción a la Programación e Ingeniería de Software

La producción de los problemas fue realizada por:

- MSc. Orlando Rivera
- Ing. Edgar Villegas
- Ing. Mauricio Del Carpio

Reglas

1. Habrán **seis** preguntas a ser completadas en **tres horas**.
2. Todas las preguntas requieren que leas los datos desde la entrada estándar y escribas los resultados por la salida estándar. No puedes usar archivos de entrada o salida.
3. Los lenguajes de programación permitidos son C, C++ y Java.
4. Todos los programas serán recompilados antes de ser probados con los datos del juez.
5. No se pueden utilizar librerías no estándares. La librería estándar STL y la librería `string` de C++ son permitidas. La API estándar de Java está disponible, exceptuando los paquetes considerados dañinos por los oficiales del concurso (e.g., aquellos que puedan generar violaciones de seguridad).
6. No se considerará el estilo de programación en este concurso. Puedes programar con el estilo que prefieras. La documentación no es requerida.
7. Toda la comunicación con los jueces se realizará a través del entorno *PC²*.
8. Las decisiones del juez son finales. No se tolerará ningún tipo de copia.
9. Se pueden utilizar recursos impresos (libros, fotocopias, etc) pero no electrónicos (memorias flash, celulares, etc).

Problema A

Contando

Código fuente: `contando.c`, `contando.cpp` o `contando.java`

Carlitos sabe contar, pero apenas está aprendiendo como escribir números. Él ya aprendió los dígitos 1, 2, 3 y 4. Pero aún no se da cuenta de que el 4 es diferente al 1, de modo que piensa que el 4 es solamente otra forma de escribir 1. Él se está divirtiendo con un jueguito que el creó: el construye números con los cuatro dígitos que conoce y suma sus valores. Por ejemplo $132 = 1 + 3 + 2 = 6$ y $112314 = 1 + 1 + 2 + 3 + 1 + 1 = 9$ (recuerda que Carlitos cree que $4 = 1$). Ahora Carlitos quiere saber cuántos de estos números puede crear cuya suma sea un número N . Para $N = 2$, él puede construir 5 números: 11, 14, 41, 44 y 2. (Él sabe contar más allá del cinco, lo que no sabe es cómo escribirlo.) Sin embargo, el no puede encontrar esta cantidad para N mayor que 2, y por eso pide tu ayuda.

Entrada

La entrada consistirá de una cantidad arbitraria de enteros N tales que $1 \leq N \leq 45$. Debes leerlos hasta el fin de archivo.

Salida

Para cada entero leído, escribe en cada renglón la cantidad de números que puede construir Carlitos tales que la suma de sus dígitos sea N .

Entrada de ejemplo	Salida para la entrada de ejemplo
2	5
3	13
10	9003
23	1707499695
24	4348691431

Problema B

Bloqueos

Código fuente: `bloqueo.c`, `bloqueo.cpp` o `bloqueo.java`

El señor Trujillo es un camionero que transporta diferentes tipos de bienes a través de su país. No obstante, ese país tuvo una serie de problemas sociales, que derivaron en el bloqueo de carreteras. De acuerdo a los contratos que firmó, utilizará el camino más rápido en sus viajes y será multado por cada 24 horas de retraso. En consecuencia, no puede esperar hasta que se resuelvan los conflictos.

Tu trabajo es calcular la multa mínima que el señor Trujillo deberá pagar si es capaz de evadir los bloqueos. Puedes asumir, que sin ningún bloqueo, siempre existirá un camino del origen al destino. Cada carretera es de doble tráfico y toma recorrerla el mismo tiempo de ida o de vuelta. No existirá más de una carretera entre dos ciudades.

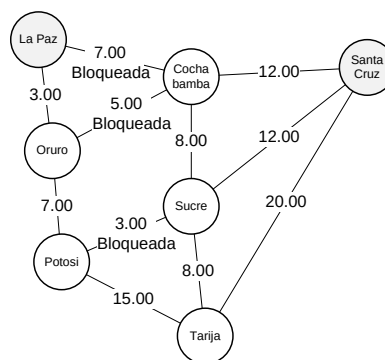


Figura 1: Mapa de un país de ejemplo.

Por ejemplo, la figura 1 representa las ciudades y carreteras de un país. Cada carretera está asociada a un tiempo en horas y una etiqueta que indica si está “Bloqueada”. El origen es “La Paz” y el destino es “Santa Cruz”. Si no existen problemas de bloqueo, toma 19 horas viajar esta distancia. Sin embargo, considerando las carreteras bloqueadas, la ruta más rápida toma 45 horas. Este tiempo es 26 horas mayor que el tiempo ideal lo que implica que el conductor deberá pagar una multa.

Entrada

Tendrás varios casos de prueba. Cada caso comenzará con un par de enteros CR separados por un espacio en una línea. C representa el número de ciudades en el país, y R el número de carreteras que conectan un par de ciudades ($2 \leq C, R \leq 100$). Las siguientes C líneas contendrán los nombres de las ciudades en ningún orden particular, que serán de 12 caracteres como máximo, sin espacios. Las siguientes R líneas representan las carreteras, cada línea tendrá 2 ciudades, las H horas ($0,00 \leq H \leq 100,00$) que toma recorrerla y si está “Bloqueada” o “Expedita”; esta información estará separada por espacios. La última línea tendrá la ciudad de origen, la ciudad de destino y la multa F que debe pagar por cada 24 horas de retraso ($0,00 \leq F \leq 300,00$), separada por espacios. Tanto el tiempo en horas y la multa en pesos, tendrán una precisión de dos dígitos decimales.

La última línea de la entrada contiene “0 0” y no debe ser procesada.

Salida

Para cada caso de prueba, debes imprimir una línea con la multa mínima que el señor Trujillo pagará por su retraso si puede evadir los bloqueos. El formato de la multa comenzará con el símbolo \$ y tendrá 2 dígitos decimales de precisión. Si no hay una ruta alternativa, debes imprimir el mensaje “Bloqueado”.

Entrada de ejemplo	Salida para la entrada de ejemplo
<pre> 7 11 La_Paz Cochabamba Santa_Cruz Tarija Oruro Potosí Sucre La_Paz Cochabamba 7.00 Bloqueada La_Paz Oruro 3.00 Expedita Oruro Cochabamba 5.00 Bloqueada Oruro Potosí 7.00 Expedita Potosí Tarija 15.00 Expedita Potosí Sucre 3.00 Bloqueada Tarija Santa_Cruz 20.00 Expedita Cochabamba Santa_Cruz 12.00 Expedita Sucre Tarija 8.00 Expedita Sucre Santa_Cruz 12.00 Expedita Sucre Cochabamba 8.00 Expedita La_Paz Santa_Cruz 100.00 4 3 New_York Washington Albany Baltimore New_York Albany 3.57 Expedita New_York Washington 9.87 Bloqueada Washington Baltimore 3.25 Expedita Albany Baltimore 225.00 0 0 </pre>	<pre> \$100.00 Bloqueado </pre>

Problema C

Cocina de Altura

Código fuente: `cocina.c`, `cocina.cpp` o `cocina.java`

Doña Sonia solía vivir en una ciudad a nivel del mar y se trasladó a una ciudad con altitud. Durante los años pasados hizo un cuaderno de recetas con los minutos de cocción de cada plato. Cuando intentó cocinar en su nueva ciudad se dio cuenta que la cocción tomaba más tiempo.

Tu tarea es determinar cuántos minutos le tomará a doña Sonia cocinar sus platos. Asume que el tiempo de cocción se incrementa según la siguiente fórmula:

$$T_{altura} = (1 + \frac{H_{altura}}{6000}) \times T_{mar}.$$

Entrada

Tendrás varios casos de prueba. Cada caso de prueba tendrá un par de enteros TH separados por un espacio en una línea. T es el tiempo en minutos que toma cocinar el plato a nivel del mar ($10 \leq T \leq 400$) y H la altura en metros a la cual doña Sonia se trasladará ($0 \leq H \leq 5000$).

La última línea de la entrada contiene “0 0” y no debe ser procesada.

Salida

Para cada caso de prueba, debes imprimir el tiempo en minutos que le tomará cocinar el mismo plato a la altitud H . Si el resultado es decimal, debes truncarlo al entero inmediato inferior.

Entrada de ejemplo	Salida para la entrada de ejemplo
40 3000	60
20 1000	23
20 1100	23
0 0	

Problema D

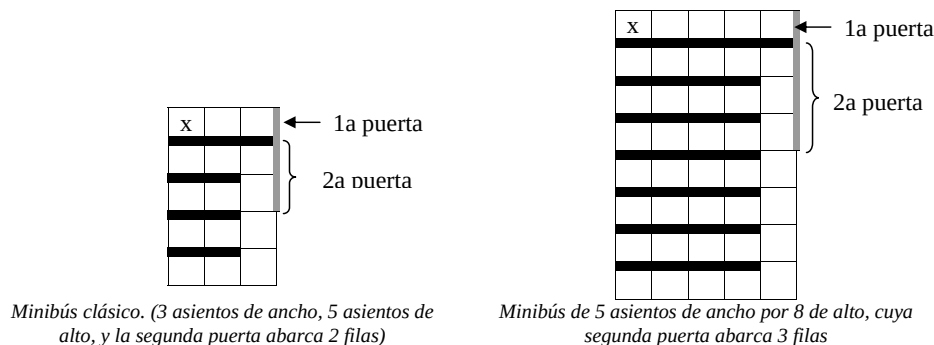
Minibus 1024

Código fuente: `minibus.c`, `minibus.cpp` o `minibus.java`

Los *minibuses* son un medio de transporte muy popular en Bolivia. Un minibús es un tipo especial de bus que tiene la peculiaridad de no tener pasillo, todo el espacio dentro del minibús está ocupado por asientos, por lo que no se desperdicia nada de espacio en el interior. Los extranjeros, muy asombrados por este tipo de transporte tan compacto, a veces lo llaman “enlatado”.

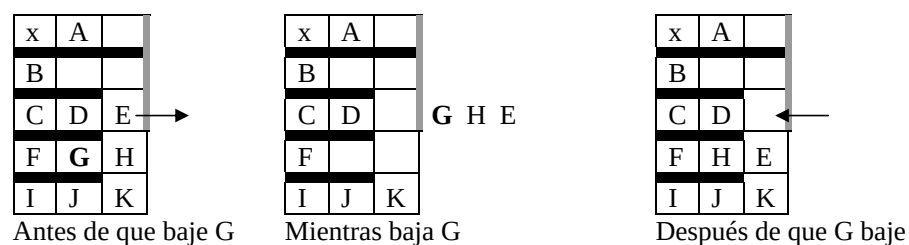
Los minibuses tienen sólo dos puertas para los pasajeros, ambas en el lado derecho del minibús. La primera puerta se encuentra en la primera fila de asientos (la fila donde se encuentra el chofer), y la segunda puerta empieza en la segunda fila y ocupa usualmente un par de ellas. Los asientos del extremo derecho, desde la segunda fila hasta la última son plegables, lo que permite a los pasajeros subir y bajar por la segunda puerta. Los demás asientos son fijos.

La siguiente figura ilustra dos minibuses diferentes. Cada casilla representa un asiento, y la “x” representa el chofer.



Pero el ahorro de espacio en los minibuses tiene un precio. Cuando un pasajero desea bajar, incomoda a todos los pasajeros en su trayecto a la puerta más cercana, haciéndolos bajar también. Una vez que el pasajero ha abandonado el minibús, los pasajeros “incomodados” que bajaron se sientan de nuevo siguiendo el siguiente criterio: el primer pasajero incomodado ocupa el lugar del pasajero que se fue, el segundo pasajero incomodado ocupa el lugar que tenía el primero, el tercero del segundo y así sucesivamente.

Por ejemplo, si el pasajero G de la siguiente figura quiere bajar, incomoda a los pasajeros H y E para bajar.



Como puede ver, después de que bajó G, H ocupa el asiento que tenía G y E ocupa el asiento que tenía H (aunque haya otros asientos libres). Se ha incomodado a 2 pasajeros, dicho de otro modo, se ha causado 2 molestias.

Si después quisiera bajar I, se incomodaría a J, K y E, entonces tendríamos $2 + 3 = 5$ molestias hasta ese momento.

El minibús 1024 es un minibus peculiar, con pasajeros peculiares. El minibús 1024 sólo recoge pasajeros en su parada y no en el camino. Note que el minibús no siempre se llena, así que puede partir con asientos vacíos.

Los pasajeros del minibús 1024 siempre se bajan en lugares diferentes, por lo que no habrá dos pasajeros que se bajen en el mismo lugar.

Además, los pasajeros se han dado cuenta de que si se organizan inteligentemente antes de subir al minibús, podrían incomodarse menos el momento de bajar.

Tu trabajo es escribir un programa que ayude a los pasajeros del minibús 1024 a lograr este objetivo, es decir hacer que cada pasajero se siente en el lugar que cause la menor cantidad posible de molestias, hasta que se bajen todos.

Entrada

Cada caso empieza con una línea que contiene 4 enteros X , Y , P y N . X es un entero que representa cuantos asientos tiene de ancho el minibus ($2 \leq X \leq 10$). Y similarmente representa el número de filas del minibús ($3 \leq Y \leq 20$), P representa la longitud (en asientos) de la segunda puerta ($1 \leq P < Y/2$), y N representa la cantidad de pasajeros que subirá ($1 \leq N < X \times Y$).

La próxima línea contiene N enteros positivos distintos D_i indicando las distancias, en cuadras desde el punto de partida hasta el punto en que bajará cada pasajero ($1 \leq D_i \leq 10000$). Note que una menor distancia significa que el pasajero bajará antes.

Una línea en blanco separa a los casos de prueba. La entrada finaliza cuando $X = Y = L = N = 0$.

Salida

Para cada caso en la entrada, el programa deberá escribir varias líneas de salida. La primera línea contiene un entero indicando el número total de molestias que se causó hasta que se bajaron todos los pasajeros. Las próximas Y líneas deben mostrar el estado inicial del minibús (después de que subieron los pasajeros). Cada línea tiene X valores separados por un espacio, indicando el estado de cada asiento. Escriba “x” si es el asiento del chofer, “-” si el asiento está vacío. Si el asiento está ocupado por un pasajero, escriba la distancia de ese pasajero.

Si hubieran varias formas de obtener el mínimo de molestias, cualquiera se considera válida. Debe escribirse una línea en blanco después de cada caso.

Entrada de ejemplo	Salida para la entrada de ejemplo
3 5 2 2	0
7 9	x - -
	- - -
4 6 3 9	9 7 -
15 10 20 18 17 5 19 12 16	- - -
	- - -
0 0 0 0	0
	x 20 - 18
	- - 16 5
	- - - -
	- - - -
	- - - 10
	19 17 15 12

Problema E

Hiperinflación

Código fuente: `hiperinflacion.c`, `hiperinflacion.cpp` o `hiperinflacion.java`

Es el año 2040 y la inflación en Bolivia está alarmante. El Dr. Conde se pone a hacer un análisis histórico (desde el año 2010) de los precios de diversos productos del mercado. Observa algo muy interesante: a partir del año 2012, el precio de un producto en cierto año es igual al precio que tenía el producto el año anterior más el doble de su precio dos años atrás.

Por ejemplo, observa que el precio de una manzana varía de la siguiente manera:

Año	2010	2011	2012	2013	2014	2015
Precio Bs.	1.00	3.00	5.00	11.00	21.00	43.00

Analizando varios datos históricos, el Doctor llega a la conclusión de que los precios de todos los productos siguen la misma tendencia. También concluye que el precio el año 2011 siempre es mayor o igual al del año 2010.

Lamentablemente, no tiene un historial completo de precios para todos los productos, y para su investigación necesita saber los precios el año 2010 (una “retro-estimación”) de cualquier producto. Afortunadamente sí tiene los precios del año 2011 y de otro año posterior, para cada producto.

El Dr. Conde te pide que hagas un programa para que averiguar el precio de un producto el año 2010, considerando todo lo mencionado.

Entrada

Cada caso consta de una línea que contiene 3 números X , A y Z separados por espacios. X es un número real que indica el precio exacto del producto el año 2011 ($0,01 \leq X < 10^{16}$). A es un número entero indicando el año para el que se tiene otro precio ($2012 \leq A \leq 2040$). Z es otro número real que representa el precio exacto del producto para el año A ($X < Z < 10^{16}$). X y Z pueden tener hasta 2 decimales. La entrada termina cuando $X = A = Z = 0$.

Salida

Para cada caso de entrada, el programa deberá imprimir una línea de salida. Esta línea contiene un solo número real con dos decimales que representa el precio exacto del producto el año 2010.

Entrada de ejemplo	Salida para la entrada de ejemplo
3.0 2015 43.0	1.00
3.0 2014 21.0	1.00
1000.0 2013 4200.0	600.00
0.60 2012 1.20	0.30
0.0 0 0.0	

Problema F

El Chaski con GPS

Código fuente: `chaski.c`, `chaski.cpp` o `chaski.java`

Un chaski es un mensajero que recorre grandes distancias a pie. Chaski Corp. es una empresa que entrega mensajes con chaskis a lo largo de Bolivia. Chaski Corp. desea controlar los recorridos de sus chaskis. Para eso, la empresa le da a cada chaski un dispositivo que registra periódicamente las coordenadas (x,y) a lo largo de su recorrido.

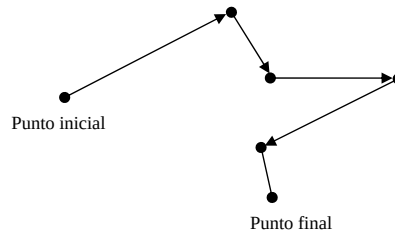


Figura 2: Recorrido de un chaski en 5 trechos.

Chaski Corp. te pide que realices un programa para determinar la distancia total recorrida por el chaski, el trecho más largo que caminó y si volvió al punto de origen al finalizar su recorrido. Asuma que el chaski camina en línea recta entre punto y punto.

Entrada

Tendrás varios casos de prueba. Cada caso empezará con una línea con el número P de puntos en la trayectoria ($2 \leq P \leq 1000$). Las siguientes P líneas contendrán un par de números ($0,00 \leq X, Y \leq 1000,00$) con dos dígitos decimales, cada par corresponde a la coordenada del punto donde se encuentra el chaski. El orden de las líneas refleja la secuencia del recorrido.

La última línea de la entrada contiene “0” y no debe ser procesada.

Salida

Para cada caso de prueba, muestra en una línea, con una precisión de dos dígitos decimales, la distancia total recorrida, la distancia del trecho más largo y si el chaski volvió o no al punto de origen.

Entrada de ejemplo	Salida para la entrada de ejemplo
5	8.23 3.16 Si
1.00 1.00	5.04 1.80 No
3.00 2.00	18.50 6.18 Si
4.50 0.50	
1.50 1.50	
1.00 1.00	
5	
0.50 1.00	
1.50 2.50	
2.50 2.00	
2.00 1.50	
1.00 0.50	
6	
3.00 3.00	
9.00 4.50	
7.00 1.00	
2.00 2.50	
1.00 2.50	
3.00 3.00	
0	