Instancia de Selección de Equipos FICH-UNL

International Collegiate Programming Contest

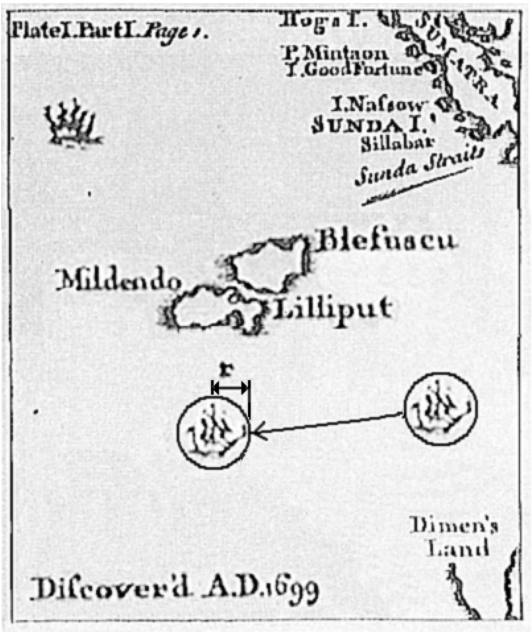
Problem Set

1 de Octubre de 2009

1. Lilliput y Blefuscu 1
2. Huellas genéticas 3
3. Sopa de Letras 5
4. Brainf*** 7

Problema 1

Lilliput y Blefuscu



Lilliput y Blefuscu son dos islas vecinas al sur del Océano Índico, separadas por un canal de 730 metros de ancho. Antiguamente, los habitantes de Lilliput, acostumbraban romper los huevos por su extremo mas grande. Sin embargo, hace algunas generaciones, el rey dictó un decreto ordenando que todos sus súbditos debían hacerlo por el extremo más pequeño, bajo pena de muerte.

Este decreto generó gran descontento entre los habitantes de Lilliput, al punto que se estima que once mil personas han preferido morir antes que someterse a la humillación de romper los huevos por el extremo más pequeño. El Emperador de Blefuscu (donde la práctica de romper los huevos por el extremo pequeño es habitual), aprovechándose de la situación, incitó varias rebeliones entre los lilliputenses, y finalmente, los imperios se declararon la guerra.

Muchas lunas pasaron desde el inicio de esta obstinada y sangrienta guerra, y tanto Lilliput como Blefuscu sufrieron importantes bajas. Lo preocupante de la situación es que Blefuscu ha equipado recientemente una numerosa flota, y se teme que en cualquier momento inicie un ataque devastador contra Mildendo, la capital del reino. Afortunadamente los lilliputenses han logrado hacerse de los planes de ataque del enemigo, por lo que Su Real Majestad Calin Deffar Plune, Emperador de la Gran Isla de Lilliput, te ha convocado para que lo ayudes a evaluar una estrategia de contraataque para el combate.

Te informan que el área sobre la superficie del mar ocupada por un barco puede aproximarse por un círculo de radio r y cada barco posee cañones que alcanzan una distancia de fuego máxima de 2000m. Al recibir un disparo, el barco se hunde *inmediatamente* y se va al fondo del mar. Lógicamente, *una vez que el barco se hundió, no puede realizar más disparos*.

Los planes de ataque del enemigo son una lista de las posiciones desde donde cada barco estará situado, la dirección y alcance de los cañonazos que planea realizar, en el orden que los disparos serán ejecutados. Una vez iniciado el combate, los barcos permanecen en la posición indicada. Su Real Majestad ha intercalado ordenes de contraataque para la flota de Lilliput, pero no sabe cuando se hundirán los barcos!

ENTRADA

La entrada consiste en varios casos de prueba. Cada caso de prueba comienza con una linea que contiene dos enteros N, F ($0 < N \leq 200$, $0 \leq F \leq 1000$) indicando la cantidad total de barcos y disparos, respectivamente.

A continuación vienen N líneas que contienen, para cada uno de los N barcos: la inicial del reino al que pertenece ("`L`" o "`B`"), las coordenadas X, Y de su ubicación y el radio r con el que puede aproximarse el área ocupada ($0 \leq X \leq 730$, $0 \leq Y \leq 1000$, $0 < r \leq 50$). Los barcos nunca están superpuestos.

Luego vienen $F \geq 0$ líneas que contienen información de los disparos que se esperan realizar, en el orden que estos disparon van ocurriendo: las coordenadas X, Y del cañon, y la dirección del disparo θ (medida en grados sexagesimales enteros, en sentido antihorario respecto al semieje X positivo, $0 \leq \theta < 360$). Los cañones siempre están situados en un barco y en ningún caso un barco se dispara a sí mismo.

La entrada termina cuando se lee una linea con $L = 0$ y $F = 0$.

SALIDA

Si Blefuscu se queda sin barcos, entonces Lilliput gana la batalla y se deberá mostrar el texto `LILLIPUT WIN`. Por el contrario, si Lilliput se queda sin barcos, Blefuscu gana la batalla y se deberá mostrar el texto `BLEFUSCU WIN`. Si la batalla no ha terminado (queda al menos un barco en cada armada) se deberá mostrar el texto `MORE BLOOD NEEDED`.

EJEMPLO

ENTRADA	SALIDA

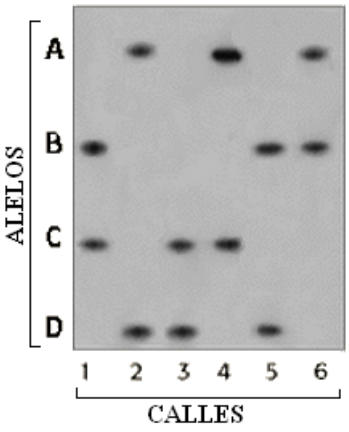
`2 2`
`L 200 500 10`
`B 400 500 10`
`400 500 90`
`200 500 0`
`3 1`
`L 200 500 10`
`L 400 500 10`
`B 600 500 10`
`200 500 0`
`0 0`
	`LILLIPUT WIN`
	`MORE BLOOD NEEDED`

Problema 2

Huellas genéticas

Los exámenes de huellas genéticas utilizan las cadenas de ADN de una persona para propósitos legales. Este tipo de tests pueden identificar a las víctimas de un crimen o de una catástrofe, aclarar o implicar a un sospechoso, o establecer relaciones biológicas entre las personas (por ejemplo, la paternidad).

La técnica se basa en explotar la repetición de secuencias altamente variables de ADN denominadas *Variable Number Tandem Repeat* (VNRT). Para cada locus[1] una persona posee exactamente dos alelos VNRT: uno heredado de su padre biológico, y uno heredado de su madre biológica (nunca un VNTR que ninguno de sus padres posee), por lo que el patrón de VNTRs en el genoma de una persona es prácticamente único.



Una muestra de VNRT del padre (calle 1), la madre (calle 2) y 4 hijos (calles 3 a 6) para un locus determinado.

Es posible que dos personas no relacionadas tengan el mismo VNRT en algunos loci. La probabilidad de que una persona tenga un cierto patrón de 2 alelos VNTR en un locus viene dada por $2pq$, en la cual p y q son las frecuencias individuales de los dos alelos en la población de referencia. Si $p = 0{,}1$ y $q = 0{,}05$, entonces la frecuencia del perfil de ADN es de $2pq = 0{,}01$. La frecuencia de un perfil un loci queda determinada por el producto de las probabilidades individuales para cada locus. Así, si la probabilidad es de $0{,}01$ para el patrón del locus 1 y $0{,}1$ para el patrón del locus 2, el perfil tiene una frecuencia de $0{,}001$ (es decir, se presenta en una de cada 1000 personas). Se asume que los loci son estadísticamente independientes.

Esto significa que la coincidencia en un determinado alelos tiene mayor valor de prueba cuando más raro es el alelo. Así, en el primer ejemplo, es más probable que los sujetos de las calles 3 y 4 sean los padres de 1 (con quien comparten el alelo C que tiene una frecuencia de 1 en 1000) antes que 5 y 6 (que comparten el alelo B, de ocurrencia más frecuente), o 2 (que no comparte ningún alelo).

Evidentemente, cuanto más loci se analicen, mayor fiabilidad se esperará del examen. Así, en el tercer caso de prueba se descubre que el sujeto 4 es el padre probable, pues comparte los alelos C y F con el sujeto 1.

[1] *Locus* (en plural: *loci*), una posición fija sobre un cromosoma, como la posición de un gen o de un marcador genético. Una variante de la secuencia de ADN en un determinado locus se denomina alelo.

ENTRADA

La entrada consiste en varios casos de prueba. La primer linea de cada caso de prueba contiene tres enteros A, C y L que indican la cantidad de alelos ($1 \leq A \leq 26$), la cantidad de calles ($2 \leq C \leq 100$) y la cantidad de loci ($1 \leq L \leq 26$) en estudio. Cada calle pertenece a un individuo distinto. A continuación viene una línea con A enteros, que indican la frecuencia de ocurrencia de cada alelo, en una de cada f_A personas ($2 \leq f_a \leq 1000$). Luego viene una línea con los pares de alelos encontrados en cada una de las C calles. Los alelos se identifican mediante letras mayúscula `A`,`B`,. . .,`Z`.

La entrada termina cuando $A = C = 0$.

SALIDA

Para cada caso de prueba se deberá informar un entero F y una lista de índices $C_1, C_2, \ldots C_n$ indicando el número de calle de los sujetos que con mayor probabilidad pueden ser padres (o madres) del sujeto cuyo material genético se presenta en la calle 1, tal que este material genético se encuentra normalmente en 1 de cada F personas, aunque no estén relacionadas.

Si la probabilidad es nula para todos los individuos, el valor de F será 0 (cero) y se indicarán todos los índices. Se garantiza que el valor de F entra en un entero con signo de 32 bits.

EJEMPLO

ENTRADA	SALIDA

`4 6 1`
`10 100 1000 10`
`BC AD AC DC BD AB`
`4 6 1`
`10 100 1000 10`
`BB AD AC DC CD AC`
`8 6 2`
`10 100 1000 10 10 10 10 10`
`BC AD AC DC BD AB`
`GF GF HH EF EF HF`
`0 0`
	`1000:3 4`
	`0:2 3 4 5 6`
	`10000:4`

Problema 3

Sopa de Letras

La sopa de letras es un pasatiempo inventado por Pedro Ocón de Oro que consiste en una cuadrícula rellena con diferentes letras y sin sentido aparente. El juego consiste en descubrir un número determinado de palabras enlazando estas letras de forma horizontal, vertical o diagonal y en cualquier sentido: son válidas las palabras tanto de derecha a izquierda como de izquierda a derecha, y tanto de arriba a abajo, como de abajo a arriba.

La Asociación de Caracteres Mezclados (ACM) organiza anualmente un Infernal Certamen de Pasatiempos Complicados (ICPC). Este famosísimo certamen se planea con mucha anticipación, y todos los detalles son cuidadosamente verificados: en primer lugar, el comité seleccionador de vocablos confecciona listas de palabras con una temática en común (el "diccionario") y posteriormente un perro, saltando sobre el teclado, genera las cuadrículas de caracteres.

Finalmente, un equipo verificador de soluciones se encarga de resolver todos los acertijos e informar cuales son las palabras del diccionario que aparecen en la cuadrícula. Lamentablemente, el gremio de verificadores está en huelga desde hace varios meses, reclamando un aumento injustificado. Por esta razón, la ACM ha decidido contratar un programador (al que le pagarán una miseria) para que desarrolle un Sistema Solucionador de Sopas.

ENTRADA

La entrada contiene sucesivos casos de prueba La primer línea de cada caso de prueba contiene tres enteros D, N, M ($0 < D \leq 100$ es la longitud del diccionario y $0 < N, M \leq 20$ definen las dimensiones de la cuadrícula). A continuación vienen D líneas con las palabras del diccionario (cada línea contiene una palabra sin espacios de entre 1 y 20 caracteres). Después vienen N líneas con M caracteres cada una, que representan la sopa de letras. El perro salta únicamente sobre letras mayúsculas.

La entrada termina cuando se lee $D = N = M = 0$

SALIDA

Por cada caso de prueba, una línea conteniendo las palabras encontradas (sin repeticiones, y en orden alfabético ascendente). Si una palabra aparece varias veces, solo se deberá listar una vez. Si una palabra del diccionario está contenida en otra (por ejemplo POLLO y REPOLLO), al encontrar la palabra de mayor extensión también se encuentra la palabra más corta.

Si ninguna palabra del diccionario aparece en la sopa de letras, se deberá mostrar un guión '−'.

EJEMPLO

ENTRADA	SALIDA
5 5 6 ICPC SALE SI ICPC NO NNJBMA GOSDEC TABTFM WSALEP GYJXIT 1 4 4 SOPA GDQZ UHVT CVVN CJUA 0 0 0	NO SALE -
3 4 3 ONU UNO NO UNO ZXW ZAQ CMG 3 4 9 POLLO REPOLLO OPERA OAKVGZLMO AREPOLLOP BVRUYWQFK EWWTYFDOO 0 0 0	NO ONU UNO OPERA POLLO REPOLLO

Problema 4

Brainf***

Un sistema computational que puede calcular cualquier función Turing-computable se denomina Turing-completo. Todos los sistemas Turing-completos actualmente conocidos son Turing-equivalentes (todas las funciones que pueden calcular son Turing-computables). *Brainfuck* es un lenguaje de programación esotérico y Turing-completo, creado por Urban Müller en 1993. Como el nombre sugiere, los programas en brainfuck generalmente son difíciles de comprender.

Brainfuck consiste en ocho instrucciones sin operandos que se codifican como un caracter. Un programa en brainfuck es una secuencia de estas instrucciones, posiblemente mezcladas con otros caracteres (que son ignorados). Las instrucciones (excepto `[]`) se ejecutan secuencialmente, incrementando el puntero de instrucción (que al comienzo de la ejecución apunta al primer comando). El programa termina cuando el puntero de instrucción se mueve a la posición siguiente al último comando.

Además del puntero de instrucción, el modelo de máquina de brainfuck consiste en un arreglo de 30.000 celdas (de un byte cada una) inicializado con ceros, un puntero a dato (inicializado a la primer posición del arreglo), un flujo de caracteres (stream) de entrada, y un flujo de caracteres de salida.

Referencias de comandos en Brainfuck

Instrucción	Significado
`>`	Incrementar el puntero de datos.
`<`	Decrementar el puntero de datos.
`+`	Incrementar en 1 el byte en la dirección apuntada.
`-`	Decrementar en 1 el byte en la dirección apuntada.
`.`	Escribir en el stream de salida el valor ASCII de la celda apuntada.
`,`	Leer un caracter ASCII (byte) del stream de entrada y guardarlo en la celda apuntada.
`[`	Si la celda apuntada contiene el valor 0, saltar al comando `]` correspondiente; en caso contrario, incrementar el puntero de instrucción al siguiente comando.
`]`	Si la celda apuntada contiene un valor distinto de 0, mover el puntero de instrucción al comando que sigue al `[` correspondiente; en caso contrario, incrementar el puntero de instrucción al siguiente comando.

Los organizadores del ICPC (Insane Computer Programming Challenge) planean realizar una competencia internacional de resolución de problemas en este bello lenguaje. Lamentablemente, no disponen de un compilador para el mismo, por lo que deciden pedirte que implementes una máquina virtual para ejecutar código brainfuck.

ENTRADA

Cada caso de prueba comienza por dos enteros $LC > 0$, $LD \geq 0$. Los restantes caracteres de la primera línea (que serán menos de 16) determinan el nombre del programa. A continuación vienen LC líneas de código en brainfuck, seguidas de LD líneas de datos para el stream de entrada de la máquina virtual (esta información será leída por la instrucción `,`). Cada linea tendrá menos de 256 caracteres. En total, por caso de prueba habrá menos de 10.000 bytes de código, y menos de 10.000 bytes de entrada.

Al finalizar *cada* línea de datos, la máquina virtual "lee" un caracter `\n` (ASCII 10 en decimal). Al finalizar la entrada del caso de prueba, la máquina virtual "lee" un caracter nulo (ASCII 0).

La entrada termina cuando se lee $LC = LE = 0$.

Asumir que los programas no contienen errores (por ejemplo, el puntero de datos nunca excede el espacio de 30.000 bytes, jamás se lee después del fin de la entrada, las instrucciones `[]` están correctamente anidadas, y el programa finaliza en un tiempo finito).

SALIDA

Para cada caso de prueba la salida comenzará por una línea con la cadena literal `PROGRAM:` seguida inmediatamente por el nombre del programa. En la siguiente línea se escribirá el stream de salida del programa en brainfuck (es decir, la salida producida por la instrucción `.`). Al finalizar la salida *para cada caso* se agregará un fin de línea adicional.

Considere que cuando la máquina virtual "escribe" un caracter `\n`, corresponde escribir un fin de linea en el stream de salida. Cuando la máquina virtual "escribe" un caracter distinto de `\n` y que no está en el rango 32..126, corresponde escribir un caracter de espacio (ASCII 32).

EJEMPLO

ENTRADA	SALIDA

`1 2 COPY.bf`
`,[.[-],]`
`ABCD`
`EFGH`
`0 0`
	`PROGRAM: COPY.bf`
	`ABCD`
	`EFGH`

`2 1 UPPER.bf`
`[one char to uppercase]`
`,>++++[<-------->-]<.`
`a`
`5 2 UPPERCASE.bf`
`++[>,`
`----- -----[`
` ----- ----- ----- ----- --.,`
` ----- -----`
`]++++++++++.[-]<-]`
`hello world`
`brainfuck instructions are KMNL\^{}`
`0 0`
	`PROGRAM:UPPER.bf`
	`A`
	`PROGRAM:UPPERCASE.bf`
	`HELLO WORLD`
	`BRAINFUCK INSTRUCTIONS ARE +-.,<>[]`