Olimpiada Boliviana de Informática

Fase Final - Nivel 2

2015

Olimpiada Boliviana de Informatica

7 de octubre de 2015

Las páginas están numeradas desde el 1 al 11 .

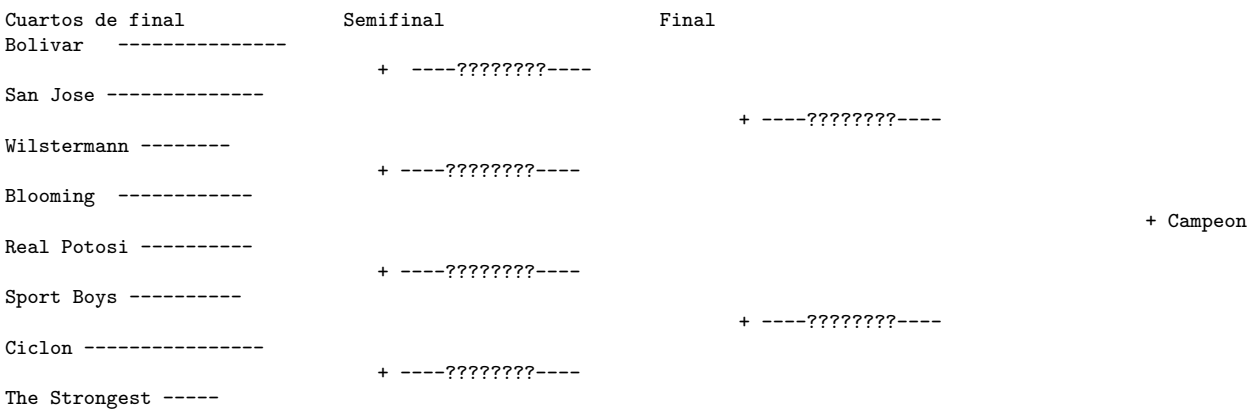
Índice

Copa Bolivia	**1**
Himno	**3**
Perlas	**5**
Plantas	**8**
Virus	**10**

Problema A
Copa Bolivia

Tiempo límite: *1 sec*
Memoria límite: *512 MB*

El futbol es uno de los deportes más jugados en Bolivia, existen varios torneos a nivel amateur pero a nivel profesional se tiene la liga profesional del futbol Boliviano y recientemente se ha creado la Copa Bolivia, un evento que incluye a equipos de primera división y algunos equipos invitados de la segunda división. La copa Bolivia se juega en un formato de eliminación directa, es decir, cada equipo enfrenta a otro equipo a un solo partido para avanzar de ronda, el ganador avanza a la siguiente fase y el perdedor se va a su casa. Por ejemplo, este sería un formato de torneo para 8 equipos con tres rondas (Cuartos de final, semifinal y final):



La Organización Boliviana Institucional de Estadística (OBIE) junto con el gremio de Apostadores Unidos (AU) se ha propuesto construir un sistema para predecir quien será el próximo campeón de la copa Bolivia, para tal efecto, se conocen las probabilidades de victoria de cada equipo contra los demás, por ejemplo, si Bolívar tiene un 80 % de probabilidad de ganarle a San José, Wilstermann un 60 % de ganarle a Blooming, Bolivar tiene un 70 % de probabilidad de ganarle a Wilstermann y un 90 % contra Blooming, entonces la probabilidad de que Bolivar llegue a la final (pase dos rondas) es: 80 % * (70 % * 60 % + 90 % * 40 %) = 62.4 %, considerando las probabilidades en el rango 0.0 - 1.0

Tarea

Su tarea es escribir un programa que calcule las probabilidades de cada equipo participante en la copa Bolivia de ser campeón.

Entrada

La entrada consiste de un número entero n que representa la cantidad de equipos participantes (n será 4, 8 o 16), luego le siguen n líneas, cada línea es una cadena de caracteres sin espacios y representa el nombre de cada equipo participante, a continuación sigue una tabla P de tamaño

nxn donde cada elemento $P[i][j]$ es la probabilidad en porcentajes de que el i-esimo equipo derrote al j-esimo equipo, por ejemplo en el grafico anterior, Bolivar es el equipo 1 y Wilstermann el equipo 3, si $P[1][3] = 70$ significa que Bolivar tiene un 70 % de probabilidad de ganarle a Wilstermann y por ende, Wilstermann tendrá una probabilidad del 30 % de ganarle a Bolivar.

Considere que para cada probabilidad en la tabla se cumplirá que $P[i][j] + P[j][i] = 100$ para todos los n equipos.

Salida

Para cada caso mostrar n líneas con el formato "NOMBRE_EQUIPO (P.PP %)" sin las comillas, donde NOMBRE_EQUIPO es el nombre de cada equipo en la copa Bolivia y P.PP es su probabilidad en porcentaje de ganar la copa Bolivia. Utilice el mismo orden de los equipos que aparece en la lectura.

Ejemplo de entrada

```
8
Bolivar
SanJose
Wilstermann
Blooming
RealPotosi
SportBoys
Ciclon
TheStrongest
50 80 70 90 40 60 45 70
20 50 50 80 70 40 50 40
30 50 50 60 20 70 40 50
10 20 40 50 60 45 40 80
60 30 80 40 50 30 40 50
40 60 30 55 70 50 50 80
55 50 60 60 60 50 50 30
30 60 50 20 50 20 70 50
```

Ejemplo de salida

```
Bolivar(35.46%)
SanJose(5.68%)
Wilstermann(11.04%)
Blooming(2.56%)
RealPotosi(8.38%)
SportBoys(20.46%)
Ciclon(8.85%)
TheStrongest(7.58%)
```

Problema B
Himno

Tiempo límite: *1.5 sec*
Memoria límite: *128 MB*

Oh no! Wordo, el malvado villano, ha invadido la ciudad! :′ (

Luego de cuantificar los daños, los inspectores se han dado cuenta de que falta algo muy importante: El himno. Si, Wordo se ha llevado el himno de la ciudad (aún se investiga como es que esto puede ser posible). Afortunadamente tu eres el mejor compositor de la ciudad, y te han dado el trabajo de escribir un nuevo himno para la ciudad. Claro, esto no es gratis, y han ofrecido pagarte una gran comisión, proporcional a la belleza del himno que compongas.

Para poder medir la belleza de tu himno, el alcalde de la ciudad te ha dado la siguiente información:

- Se usará una escala de notas especial, basada en la serie de fibonacci, donde cada nota esta representada por el siguiente valor $F(i)\,\%M$ (M es un número que elegiras tú). Si se usan otras notas la belleza del himno es 0.
- Un himno se considera bello cuando las notas que lo componen están ordenadas, es decir, el himno tiene la forma: $F(0)\,\%M, F(1)\,\%M, F(2)\,\%M...F(N)\,\%M$. De otra manera, se considera que la belleza del himno es 0.
- El himno no debe repetir juntas un mismo par de notas. Es decir, si en algun momento el himno contiene el par de notas X, Y en ese orden, no puede existir X, Y en ninguna otra parte del himno. Si esto sucede la belleza del himno es 0.
- Si se cumplen las demas condiciones, la belleza del himno es igual a la cantidad de notas que se tocan. Por ejemplo si M es igual a 2, el himno es: $0, 1, 1$, la belleza total será 3

Solo por si no lo recuerdas, la serie de fibonacci es esta: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, ...

Entrada

Este problema no tiene entrada.

Salida

En la primera línea de salida deberán estar dos enteros N y M, la cantidad de notas que se tocan en tu himno y el número M que elegiste. En la segunda línea deberás mostrar N números, separados por un espacio, que representa el himno que compusiste.

```
4 2
0 1 1 0
```

Tarea

Este problema tiene una subtarea única que vale 100 puntos. Para conseguir el puntaje completo, tu himno deberá tener una belleza de por lo menos 300.000.

Si la belleza del himno que tu programa produce es de X, tu puntaje será dado por la proporción de X con respecto a 300.000. Formalmente:

$$puntaje = (min(X, 300{,}000)/300{,}000) * 100$$

Problema C
Perlas

Tiempo límite: *2 sec*
Memoria límite: *128 MB*

Addis se fue de viaje unos días a algún lugar que nadie conoce. Como siempre piensa en sus amigos, decidió traer unos souvenirs para ti: pequeñas perlas esféricas.

Antes de dártelas, Addis te propone un reto. Él te prestará una cantidad N de perlas, todas idénticas a simple vista y con el mismo peso. Sin embargo, entre ellas habrá una que es diferente de las demás, porque es más liviana. A esta perla la llamaremos *perla falsa*, y el reto consiste en que deberás encontrarla.

Lamentablemente, la diferencia de peso es muy insignificante como para que la puedas detectar con las manos. ¡Qué suerte que Diego está dispuesto a prestarte su balanza!

La balanza tiene dos platos, y para que funcione correctamente, en cada plato debes poner la misma cantidad de perlas. Como Addis encuentra injusto que uses la balanza, te indica que te regalará más perlas mientras menos veces la utilices. Por esto, tienes que encontrar la perla falsa con la menor cantidad de pesajes.

Resumen de la tarea

Escribe un programa que encuentre la perla falsa entre las N que Addis te ha prestado. Se te pedirá que resuelvas múltiples casos de prueba, donde cada uno corresponde a un conjunto diferente de perlas prestadas.

Interludio: Este ejercicio requiere trabajar con un programa evaluador

Para resolver este problema, querido concursante, sólo necesitarás implementar correctamente una función en C++, ya que el ejercicio trae consigo código que se encarga de realizar la lectura de los datos entrada y emitir la salida necesaria para que te evaluemos. Este código que facilita el trabajo se llama programa evaluador (en inglés, **grader**).

Adjunto a este enunciado, o en tu computadora, deberías tener tres archivos:

- grader.cpp, que es el programa evaluador. No necesitas tocar este archivo.
- Perlas.h, que es un archivo de definición de funciones y constantes, utilizadas por el evaluador. No necesitas tocar este archivo.
- Perlas.cpp, que es el único archivo donde deberás escribir código para resolver el problema.

Para realizar pruebas en tu propio equipo, puedes hacer una de dos cosas: 1) crear un proyecto de CodeBlocks y añadirle los tres archivos ya mencionados, ó 2) compilar mediante la línea de comandos usando la siguiente instrucción:

```
g++ -DEVAL -static -O2 -std=c++11 -o Perlas grader.cpp Perlas.cpp
```

De cualquiera manera, al final tendrás un programa que podrás ejecutar. Este programa leerá datos para los diferentes casos de prueba, y se encargará de evaluar el desempeño del código que escribiste en el archivo Perlas.cpp.

Ya que vas a necesitar usar el grader para probar la función que escribas, el último subtítulo de este documento describe el formato en que este programa lee su entrada y lo que imprime como salida.

Descripción de la tarea

Tú programa deberá implementar la función `find_odd_pearl`, de la cual ya existe una definición a medias en el archivo Perlas.cpp. El grader llamará a `find_odd_pearl(N)` exactamente una vez por cada caso de prueba.

Para propósitos del problema, las perlas están numeradas desde 1 hasta N inclusive.

- `find_odd_pearl(N)`
  - Esta función se llama una vez por cada caso de prueba.
  - Su único argumento N te indica la cantidad de perlas que Addis te prestó, entre las cuales obviamente hay una falsa.
  - Deberás determinar cuál es la perla falsa, realizando múltiples llamadas a una función que el grader pone a tu disposición: `weigh`. Más detalles sobre ella abajo.
  - El resultado de esta función debe ser un entero que indique el número de la perla falsa.
- `weigh(A, B)` - corresponde a utilizar la balanza de Diego
  - A y B son vectores de enteros (`vector<int>`). Es decir, la función se utiliza pasándole dos vectores como argumentos. (No puedes usar arreglos estáticos `int[]`).
  - El contenido de A y B indican qué perlas se han de poner en el plato izquierdo de la balanza y qué perlas en el derecho, respectivamente. Por tanto, los enteros que se pongan en estos vectores deben corresponder a números de perlas (entre 1 y N).
  - La función devuelve uno de los siguientes enteros para indicar el resultado del pesaje:
    - -1 (la constante `WR_LESS`). Indica que el plato izquierdo es el más liviano.
    - 0 (la constante `WR_EQUAL`). Indica que ambos platos pesan lo mismo.
    - 1 (la constante `WR_GREATER`). Indica que el plato izquierdo es el más pesado.
    - 2 (la constante `WR_INVALID`). Indica que los vectores A y B no contienen información correcta para realizar un pesaje. Esto ocurre si no tienen la misma cantidad de perlas, si contienen algún entero que no está entre 1 y N, o si alguna perla aparece más de una vez en la comparación.

Ejemplo

Imaginemos el caso en que Addis nos da $N = 1$ perla.

Claramente, como sólo hay una única perla, esta tiene que ser la perla falsa. Por ello, podemos retornar 1 (uno) desde nuestra función.

Imaginemos el caso en que Addis nos da $N = 2$ perlas.

Sabemos que una perla es falsa, y para determinar cuál vamos a necesitar pesarlas entre sí. Para esto llamamos a `weigh({1}, {2})`, de modo que en el plato izquierdo de la balanza esté la perla 1 y en el plato derecho la perla 2. Si el resultado de `weigh` es `WR_LESS`, esto quiere decir que la perla 1 es la más liviana y por tanto es la perla falsa; caso contrario, debe ser la perla 2.

Subtareas

Para todas las subtareas se cumple que $1 \leq N \leq 1000$.

Subtarea	**Puntos**	**Restricciones**
1	15	Sin límite de llamadas a `weigh`.
2	25	Hasta 500 llamadas a `weigh`.
3	30	Hasta 10 llamadas a `weigh`.
4	30	Hasta 7 llamadas a `weigh`.

Una solución que no encuentre la perla falsa en todas las situaciones posibles recibirá cero puntos, sin importar la cantidad de llamadas a `weigh` que haga.

Formatos de entrada y salida del grader

El grader trabaja leyendo casos de prueba hasta que alcanza el final de archivo (EOF).

Un caso de prueba se describe mediante dos números enteros N y M separados por varios espacios o saltos de línea, donde N es la cantidad de perlas que Addis te está prestando y M es el número de la perla falsa.

Por ejemplo, una entrada que consista de los casos de prueba `5 3` (5 perlas, la tercer perla es la falsa) y `8 1` (8 perlas, la primera perla es falsa) puede escribirse así:

```
5 3
8 1
```

El grader imprimirá una línea con dos enteros por cada caso. El primer entero es el resultado de la llamada a `find_odd_pearl`, el segundo es la cantidad de llamadas a `weigh` realizadas.

Problema D
Plantas

Tiempo límite: *2 sec*
Memoria límite: *512 MB*

Hay N plantas en un jardín. A cada una de esas plantas le fue añadida un monto de pesticida. Después de cada día, si una planta tiene más pesticida que la planta que está a su izquierda, al ser más débil que la de la izquierda, muere. En otras palabras, si la planta de la izquierda tiene menos pesticida que la de la derecha, la de la derecha muere. Te es dado el monto inicial de pesticida en cada planta. Imprime el número de días después del cual ninguna planta muere. En otras palabras, imprime el tiempo después del cual no hay plantas que tengan más pesticida que la planta que está a su izquierda.

Entrada

La entrada consiste de un entero N. La siguiente línea consiste de N enteros describiendo el arreglo P, donde $P[i]$ denota el monto de pesticida en la planta i.

Salida

Imprime un solo valor que represente el número de días después del cual ninguna planta muere.

Tarea

Subtarea 1: (40 ptos.)

$1 \leq N \leq 1000$ y $1 \leq P[i] \leq 50$

Subtarea 2: (60 ptos.)

$1 \leq N \leq 35000$ y $1 \leq P[i] \leq 10^9$

Ejemplo de entrada

```
7
6 5 8 4 7 10 9
```

Ejemplo de salida

```
2
```

Explicación

Inicialmente todas las plantas están vivas.

Plantas: (6,1),(5,2),(8,3),(4,4),(7,5),(10,6),(9,7).

Plantas[k]=(i,j) la j-ésima planta tiene “i” de pesticida

Después del primer día, 4 plantas quedan y las plantas 3,5 y 6 mueren.

Plantas= {(6,1),(5,2),(4,4),(9,7)}

Después del segundo día, 3 plantas sobreviven y la planta 7 muere.

Plantas= {(6,1),(5,2),(4,4)}

En el tercer día pasa nada, entonces después del segundo día las plantas dejaron de morir.

Problema E
Virus

Tiempo límite: *2 sec*
Memoria límite: *512 MB*

En biología, un virus (del latín virus, "toxina" o "veneno") es un agente infeccioso microscópico acelular que sólo puede multiplicarse dentro de las células de otros organismos. Los virus infectan todos los tipos de organismos, desde animales y plantas, hasta bacterias. Los virus son los responsables de provocar muchas de las enfermedades en el ser humano, desde la gripe o resfriado hasta el temible ébola. El estudio de los virus es de gran importancia para poder crear vacunas y medicamentos que prevengan enfermedades sin embargo es una tarea muchas veces complicada ya que los virus al ser una de las primeras formas de vida, suelen mutarse y cambiarse dependiendo de la atmósfera donde se desarrollan.

La Organización de Control de Epidemias y Pandemias de Bolivia (OCEPB) es una institución dedicada al estudio de virus y como parte de su investigación ha descubierto un nuevo virus al que ha llamado virus OB-I (compuesto de dos toxinas, la Obotoxina y la Iconitina), este virus tiene un comportamiento peculiar ya que por cada hora de vida del virus, la Obotoxina se multiplica para formar 3 Obotoxinas y 1 Iconitina, y las Iconitinas se multiplican a la siguiente hora, cada Obotoxina se multiplica de la misma forma y las Iconitinas se multiplican formando 4 Iconitinas, a continuación se muestra lo observado en un microscopio durante tres horas, se ha organizado las células infectadas de forma cuadricular para un mejor estudio:

Obotoxina = O Iconitina = I

Hora 1:

	1
1	O

Hora 2:

	1	**2**
1	O	O
2	O	I

Hora 3:

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
1	O	O	O	O	O	O	O	O
2	O	I	O	I	O	I	O	I
3	O	O	I	I	O	O	I	I
4	O	I	I	I	O	I	I	I
5	O	O	O	O	I	I	I	I
6	O	I	O	I	I	I	I	I
7	O	O	I	I	I	I	I	I
8	O	I	I	I	I	I	I	I

La Obotoxina, en la célula (i, j) se multiplica para producir 3 Obotoxinas en las células $(i * 2 - 1, j * 2 - 1), (i * 2 - 1, j * 2), (i * 2, j * 2 - 1)$ y una Iconitina en la célula $(i * 2, j * 2)$. La

Iconitina en la célula (i, j) se multiplicará para producir 4 Iconitinas en las células $(i*2-1, j*2-1), (i*2-1, j*2), (i*2, j*2-1) y (i*2, j*2)$. Se puede apreciar que la cantidad de células infectadas se incremento después de cada hora. Ya que la Obotoxina es peligrosa para los seres vivos y no tanto así la Iconitina, en este problema la institución OCEPB está interesada en el recuento de la cantidad de Obotoxina producida en una determinada cantidad de horas en un determinado rango de células infectadas.

Entrada

Cada caso de entrada está compuesto por tres enteros H, A y B, donde H $(0 \leq H \leq 30)$ es la cantidad de horas donde sucede la observación del virus y $[A, B]$ el rango de células a observar $(1 \leq A \leq B \leq 2^H)$.

Salida

Mostrar el número de Obotoxinas encontradas en el rango de filas de $[A, B]$ después de la hora H.

Ejemplo de entrada 1

```
0 1 1
```

Ejemplo de salida 1

```
1
```

Ejemplo de entrada 2

```
3 1 8
```

Ejemplo de salida 2

```
27
```

Ejemplo de entrada 3

```
3 3 7
```

Ejemplo de salida 3

```
14
```