# Séptimo Concurso Local de Programación ACM ICPC

## *Universidad Autónoma Metropolitana Unidad Azcapotzalco*

## Examen final, 8 a 11 de octubre de 2010

***Inicio: 15:00 del 8 de octubre. Fin: 09:00 del 11 de octubre.***

**Instrucciones**: A continuación encontrarás los enunciados de tres problemas. Cada problema tiene un valor de 100 puntos, los cuales se obtendrán de las salidas que tu programa entregue para cada una de 10 entradas distintas. Todos los programas se evaluarán en Linux. El que tu programa funcione con el ejemplo no quiere decir que funcionará siempre. Ningún programa se ejecutará por más de 5 segundos.

Para cada problema que resuelvas deberás enviar por correo el código fuente de un programa a la dirección `uam10acm@gmail.com` indicando claramente tu nombre completo. De preferencia envía todos tus códigos en un solo correo. El nombre de los archivos que envíes debe ser de la forma `xxx.zzz`, donde `xxx` es el nombre del problema y `zzz` es la extensión, según el lenguaje que hayas usado. Si así lo deseas, puedes resolver cada problema en un lenguaje distinto.

Tu programa deberá leer y escribir exactamente los datos que se indican (ni más ni menos) usando los **archivos** de entrada y salida indicados. En particular, tu programa no deberá: borrar la pantalla, escribir ningún tipo de letrero adicional, esperar ninguna tecla, usar la biblioteca conio, etc.

Quienes programen en C deberán usar el `gcc`, quienes programen en C++ deberán usar el `g++` y quienes programen en Java deberán usar el `gcj` de la siguiente manera:

```
gcc xxx.c -lm -o xxx
g++ xxx.cpp -lm -o xxx
gcj xxx.java –main=xxx -Wall -o xxx
```

**Nota**: Todos los puntos calculados que resulten fraccionarios se redondearán siempre hacia abajo.

## *Problema 1: No te salgas (100 puntos)*

## Código fuente: nts.c, nts.cpp, nts.java

### *Archivo de entrada: nts.ent Archivo de salida: nts.sal*

Una pulga muy especial se encuentra en la esquina de una mesa y quieres que brinque hasta la esquina contraria de la misma mesa. Lo especial de la pulga es que se ha aprendido una sucesión numérica y la usará para determinar cómo brinca. Cada vez que aplaudas, la pulga decidirá si se queda donde está o si da un brinco cuya longitud está determinada por el término correspondiente de la sucesión. Los brincos siempre serán paralelos a los lados de la mesa y siempre deberán alejar a la pulga de su posición inicial.

Lo ideal es que la pulga logre llegar a la esquina contraria de la mesa. Sin embargo, basta con que llegue a una posición tan cercana como sea posible. Lo que no está permitido es que la pulga salte hacia afuera de la mesa (le ha costado mucho subir a la mesa como para que ahora la dejes caer).

Como ejemplo, si la sucesión de **n** = 7 saltos de la pulga fuera **s** = (3, 1, 4, 1, 5, 9, 2) y la mesa midiera **a** = 5 por **b** = 6 entonces la pulga podría lograr su objetivo si su sucesión de brincos fuera arriba, arriba, derecha, arriba, nada, nada y derecha. Por otro lado, si su sucesión fuera arriba, arriba, derecha, nada, nada, nada, nada entonces no llegaría a la esquina contraria pero tampoco se saldría de la mesa.

**Entrada**: El archivo de texto `nts.ent` contendrá un renglón con tres enteros **n**, **a** y **b** separados por espacios seguido de un renglón con **n** enteros positivos $\mathbf{s_1}$, $\mathbf{s_2}$, ..., $\mathbf{s_n}$ y separados por espacios. Puedes suponer que todos estos enteros están en el rango de 1 a 1000, incluyéndolos.

**Salida**: El archivo de texto `nts.sal` deberá contener un renglón con tres enteros **n**, **c** y **d** separados por espacios seguido de un renglón con **n** enteros positivos $\mathbf{t_1}$, $\mathbf{t_2}$, ..., $\mathbf{t_n}$ y separados por espacios. Los valores de **c** y **d** representan la coordenada a la que llega la pulga y para $1 \le \mathbf{i} \le \mathbf{n}$, el valor de $\mathbf{t_i}$ es 0 si la pulga no dio el salto de longitud $\mathbf{s_i}$, 1 si lo dio hacia arriba y 2 si lo dio hacia la derecha.

**Evaluación**: 1 punto si los **n** saltos indicados dejan a la pulga en la coordenada (**c**, **d**) de la mesa. En ese caso, $9(\mathbf{c}^2 + \mathbf{d}^2)/(\mathbf{a}^2 + \mathbf{b}^2)$ puntos adicionales. El primer ejemplo de salida mostrado abajo obtendría $1 + 9(5^2+6^2)/(5^2+6^2) = 10$ puntos, el segundo ejemplo $1 + 9(4^2+4^2)/(5^2+6^2) = 4$ puntos y el tercer ejemplo 0 puntos porque la pulga se salió de la mesa.

| Ejemplo de entrada | Ejemplo de salida | Ejemplo de salida | Ejemplo de salida |
|---|---|---|---|
| 7 5 6<br>3 1 4 1 5 9 2 | 7 5 6<br>1 1 2 1 0 0 2 | 7 4 4<br>1 1 2 0 0 0 0 | 7 15 10<br>1 1 2 2 2 1 1 |

## *Problema 2: Dígitos borrados en una suma (100 puntos)*

## Código fuente: `dbs.c`, `dbs.cpp`, `dbs.java`

### *Archivo de entrada: `dbs.ent` Archivo de salida: `dbs.sal`*

Un maestro de primaria no puede creer que tiene un pupilo que es capaz de hacer sumas mentalmente. Entonces se le ocurrió ponerle un reto más sofisticado a su alumno. En lugar de pedirle que haga una suma, le pedirá que descubra cuál es la suma que se tiene que hacer. Así, secretamente el maestro hace la suma por su cuenta (lamentablemente con la ayuda de una calculadora) y se la escribe en el pizarrón a su estudiante pero con algunos dígitos borrados. La labor del escolar es descubrir los dígitos faltantes.

Por ejemplo, si la suma escogida fuera 123 + 456 = 579 entonces el maestro pudo haber escrito en el pizarrón 12X + X5X = 5X9. Por su parte, el colegial pudo haber respondido que 121 + 458 = 579 y no habría manera de que el maestro le dijera que su respuesta estaba mal.

**Entrada**: El archivo de texto `dbs.ent` contendrá tres cadenas **A**, **B** y **C** formadas por los caracteres del 0 al 9 o la letra X. Cada cadena tendrá un mínimo de 1 carácter y un máximo de 10 caracteres. A lo más la mitad de todos los caracteres involucrados son la letra X.

**Salida**: El archivo de texto `dbs.sal` deberá contener un entero **n** seguido de **n** renglones cada uno de ellos con tres cadenas $\mathbf{A_i}$, $\mathbf{B_i}$ y $\mathbf{C_i}$ formadas por los caracteres del 0 al 9 que sean soluciones distintas.

**Evaluación**: 1 punto si todas las soluciones propuestas son distintas y correctas. En ese caso, 9**n**/**m** puntos adicionales, donde **m** es el número total de soluciones distintas. El ejemplo de salida mostrado abajo recibiría 1 + 9*2/10 = 2 puntos debido a que en realidad hay 10 soluciones distintas.

| Ejemplo de archivo de entrada | Ejemplo de archivo de salida |
|---|---|
| `12X X5X 5X9` | `2`<br>`123 456 579`<br>`121 458 579` |

## *Problema 3: Máximos de muchos rangos (100 puntos)*

## Código fuente: `mmr.c`, `mmr.cpp`, `mmr.java`

### *Archivo de entrada: `mmr.ent` Archivo de salida: `mmr.sal`*

Pedro es estadista y necesita obtener cierta información de una encuesta cuyo resultado es una lista de **n** enteros positivos $\mathbf{a_1}$, $\mathbf{a_2}$, …, $\mathbf{a_n}$. Lo que él necesita saber es cuál es el valor máximo de los elementos de la lista en el rango $\mathbf{a_p}$, …, $\mathbf{a_q}$ y a esto le llama una consulta. Hacer esta tarea una vez es algo muy simple, sin embargo el problema es que él necesita realizar **c** consultas y al final requiere calcular la suma de los resultados de las consultas realizadas.

Por ejemplo si Pedro tiene **n** = 10 datos y estos son 3, 1, 4, 1, 6, 2, 7, 1, 8, 2 entonces el resultado de la consulta con **p** = 1 y **q** = 10 es 8, mientras que el resultado de la consulta con **p** = 2 y **q** = 5 es 6. Si además Pedro hizo las consultas con **p** = 3 y **q** = 3, **p** = 4 y **q** = 8 y nuevamente **p** = 1 y **q** = 10 entonces obtendría los resultados 4, 7 y 8. Así, la suma de los cinco resultados obtenidos es 33.

**Entrada**: El archivo de texto `mmr.ent` contendrá un entero **n**, seguido un renglón con **n** enteros $\mathbf{a_1}$, $\mathbf{a_2}$, …, $\mathbf{a_n}$ separados por espacios, seguido de un renglón con un entero **c**, seguido de **c** renglones con dos enteros $\mathbf{p_i}$ y $\mathbf{q_i}$ separados por espacios. Puedes suponer que $1 \leq \mathbf{n} \leq 10{,}000$, que $1 \leq \mathbf{a_i} \leq 10{,}000$ para toda $1 \leq \mathbf{i} \leq \mathbf{n}$, que $1 \leq \mathbf{c} \leq 10{,}000$ y que $1 \leq \mathbf{p_i} \leq \mathbf{q_i} \leq \mathbf{n}$ para toda $1 \leq \mathbf{i} \leq \mathbf{c}$.

**Salida**: El archivo de texto `mmr.sal` deberá contener un entero **s**.

**Evaluación**: 10 puntos si el valor de **s** es correcto.

| Ejemplo de archivo de entrada | Ejemplo de archivo de salida |
|---|---|
| 10<br>3 1 4 1 6 2 7 1 8 2<br>5<br>1 10<br>2 5<br>3 3<br>4 8<br>1 10 | 33 |