Octavo Concurso Local de Programación ACM ICPC

Universidad Autónoma Metropolitana

Examen eliminatorio, 23 a 28 de septiembre de 2011

Instrucciones: Abajo encontrarás los enunciados de cinco problemas. Cada problema tiene un valor entre 20 y 60 puntos, los cuales se obtendrán de las salidas que entregue tu programa para cada una de 10 entradas distintas. **Todos los programas se evaluarán en Linux y se ejecutarán un máximo de 5 segundos.** Que tu programa funcione con el ejemplo no quiere decir que funcionará siempre.

Para cada problema que resuelvas deberás enviar por correo electrónico el código fuente de un programa a la dirección `uam11acm@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 e imprimir exactamente los datos que se indican (ni más ni menos) usando la entrada y la salida estándar. En particular, tu programa no deberá borrar la pantalla, escribir ningún tipo de letrero adicional, usar la biblioteca conio, etc. Quienes programen en C deberán usar `gcc`, quienes programen en C++ deberán usar `g++` y quienes programen en Java deberán usar `gcj` como sigue:

```
gcc nombre_del_fuente.c -o nombre_del_ejecutable -lm
g++ nombre_del_fuente.cpp -o nombre_del_ejecutable -lm
gcj nombre_del_fuente.java -o nombre_del_ejecutable
```

Deberás enviar tus códigos fuente a más tardar el 28 de septiembre de 2011 a las 22:00.

Problema 1: Ganador del juego de gato (20 puntos)
Código fuente: gjg.c, gjg.cpp, gjg.java

El gato es un juego donde participan dos personas y se juega en un tablero cuadrado que se encuentra dividido por dos líneas horizontales y dos líneas verticales, formando una cuadrícula de tamaño 3 por 3. A cada jugador se le asignan cinco fichas, que pueden ser A o B. El juego comienza cuando uno de los dos jugadores coloca una de sus fichas en el tablero, luego el oponente coloca una ficha y así sucesivamente se van turnando ambos jugadores. Cualquiera de los dos jugadores puede comenzar el juego. El objetivo del juego es que algún jugador forme una línea recta con tres fichas a lo largo del tablero. Gana el primero que cumpla esta meta. Escribe un programa que, dado un tablero que contiene una partida de gato, determine el estado de dicha partida.

Entrada: Tres líneas cada una con tres caracteres indicando el estado del tablero. Cada carácter será una 'A', 'B' o '.', donde el carácter '.' denota una casilla vacía.

Salida: Una A si el jugador con las fichas A ha ganado el juego, una B si el jugador con las fichas B ha ganado el juego, una E si no se puede decidir un ganador según el tablero dado o una I si el estado del tablero es inválido.

Evaluación: 2 puntos por la salida correcta.

Ejemplo de entrada	*Ejemplo de salida*	*Ejemplo de entrada*	*Ejemplo de salida*

`AA.`
`.A.`
`BBB`
	`B`	`AAB`
		`AAA`
		`BBB`
			`I`

Problema 2: Área máxima de un trapecio (30 puntos)
Código fuente: amt.c, amt.cpp, amt.java

Un carpintero tiene cuatro tablas de madera de longitudes **a**, **b**, **c**, **d** con las cuales desea construir los lados de una caja con forma de *trapecio* (es decir, un cuadrilátero que tiene dos lados paralelos). Aunque sabe cómo construir dicha caja, no sabe cuál será el área del fondo resultante. Escribe un programa que determine el área que tendrá el fondo de la caja considerando que ha sido construida con un área máxima. Puedes suponer que con las longitudes dadas se puede construir al menos un trapecio.

Entrada: Cuatro números enteros **a**, **b**, **c**, **d** con $1 \leq$ **a**, **b**, **c**, **d** ≤ 1000.

Salida: El área máxima **m** que puede tener la caja. Si dicha área tiene una parte decimal, ésta debe ser truncada.

Evaluación: 3 puntos por el valor correcto de **m**.

Ejemplo de entrada	*Ejemplo de salida*	*Ejemplo de entrada*	*Ejemplo de salida*
2 2 2 2	4	2 5 3 2	6

Problema 3: Números en base Fibonacci (40 puntos)
Código fuente: nbf.c, nbf.cpp, nbf.java

La sucesión de Fibonacci comienza con F(1) = 1, F(2) = 2 y continúa con F(**n**) = F(**n**-1) + F(**n**-2) para toda **n** ≥ 3. Todo número entero positivo se puede escribir como suma de elementos de la sucesión de Fibonacci, por ejemplo, el número 10 se puede escribir como F(4)+F(4), F(4)+F(3)+F(2) o F(5)+F(2).

Sin embargo, existe una única forma de escribir cada entero positivo como sumas de elementos distintos de la sucesión de Fibonacci que no sean consecutivos, en nuestro caso 10 = F(5)+F(2) sería la única forma válida. Escribe un programa que dado un entero positivo **n** encuentre esta suma única.

Entrada: Un número entero **n**. Puedes suponer que 1 ≤ **n** ≤ 1,000,000,000.

Salida: Un número entero **k**, la cantidad de sumandos, seguido de **k** números enteros a_1, a_2, …, a_k tales que **n** = $F(a_1)+F(a_2)+\ldots+F(a_k)$, $a_1 > a_2 > \ldots > a_k > 0$.

Evaluación: 2 puntos si la salida es correcta y 2 puntos si además es la única forma válida.

Ejemplos de entrada	*Ejemplos de salida*
10	2 5 2
25	3 7 3 1

Problema 4: Prefijos de palabras de diccionarios (50 puntos)
Código fuente: ppd.c, ppd.cpp, ppd.java

Desde pequeño te han gustado mucho los diccionarios y con frecuencia te preguntas cuántas palabras existen que empiecen con determinadas letras. Ya que finalmente aprendiste programación crees poder contestar estas preguntas rápida y fácilmente.

Entrada: Un entero **N** seguido de un diccionario de **N** palabras y después un entero **M** seguido de **M** prefijos. Puedes suponer que 0 ≤ **N** ≤ 100,000 y 0 ≤ **M** ≤ 100,000. Tanto las palabras del diccionario como los prefijos sólo contendrán letras minúsculas y no superarán los 15 caracteres de longitud.

Salida: Para cada uno de los **M** prefijos, un entero que sea el número de palabras del diccionario que comienzan con dicho prefijo.

Evaluación: (5**c**)/**M** puntos, donde **c** es la cantidad de respuestas correctas.

Ejemplo de entrada	*Ejemplo de salida*
5 gato ganso casa caja comida 3 ga ca c	2 2 3

Problema 5: Documentos de marcado equilibrados (60 puntos)
Código fuente: dme.c, dme.cpp, dme.java

Como sabes, muy pocas páginas de internet se ven igual usando exploradores diferentes. Una herramienta muy importante para prevenir incompatibilidades es la validación de código escrito en *lenguajes de marcado* y aunque para páginas web esto se puede complicar bastante, tu labor será más sencilla.

Para nuestros fines, una *etiqueta de apertura* es un *identificador* compuesto por caracteres alfanuméricos y guiones bajos (aunque por regla el primer carácter del identificador no puede ser un número) y su *etiqueta de cierre* será el mismo identificador precedido por una /. Puede haber un número arbitrario de espacios en blanco entre etiquetas. Las etiquetas tienen una *posición* numerada de forma consecutiva a partir del 1.

El documento tiene una *profundidad* inicial de 0, cada vez que se encuentra una etiqueta de apertura la profundidad aumenta en 1 y cada vez que se encuentra una de cierre la profundidad disminuye en 1.

Un documento de marcado está mal formado si:

1. Tiene etiquetas inválidas (123_gato es inválida pues inicia en número).
2. Alguna etiqueta de apertura no tiene su cierre o viceversa.
3. El cierre de una etiqueta externa ocurre antes del cierre de una etiqueta interna (por ejemplo aa bb /bb /aa es correcto pero aa bb /aa /bb no lo es).

Entrada: Una línea de texto de no más de 10,000 caracteres. Puedes suponer que la longitud de las etiquetas contenidas dentro del texto no superará los 100 caracteres cada una.

Salida: Una letra mayúscula seguida de un número entero. En el caso de que el documento esté bien formado entonces la letra debe ser V y el número entero debe ser la profundidad máxima. En el caso de que el documento esté mal formado entonces la letra debe ser I y el número entero deberá ser la posición de la primera etiqueta que provoque un error.

Evaluación: 1 punto si la letra es correcta y 5 puntos si el entero es correcto.

Ejemplos de entrada	**Ejemplos de salida**
`aa bb /bb /aa aa /aa`	`V 2`
`aa 12 /12 /aa`	`I 2`
`aa bb /aa`	`I 3`
`/bb`	`I 1`

```
/* el comité organizador del concurso te desea suerte en tu examen */
```