Séptimo Concurso Local de Programación ACM ICPC

Universidad Autónoma Metropolitana

Examen eliminatorio, 23 a 28 de septiembre de 2010

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.** 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 `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 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 2010 a las 22:00.

Problema 1: Recuperación de aminoácidos (20 puntos)
Código fuente: rda.c, rda.cpp, rda.java

En bioinformática, los aminoácidos son formados a partir de tripletas de nucleótidos a los cuales se les llama codones. Cada tripleta que forma un aminoácido para todas las combinaciones posibles, recibe un nombre y a esto es lo que se le conoce como código genético. Dicho código, al ser codificado como cadena puede contener solamente las letras A, C, G y T. En el laboratorio se han mezclado por accidente secuencias de códigos genéticos con otros tipos de códigos que por el momento no nos interesan. Dichos códigos están formados por todas las letras del alfabeto menos la A, C, G y T. A pesar de la mezcla, en ocasiones es posible recuperar la secuencia de código genético siempre que se mantenga la propiedad de que sea posible formar tripletas y sólo estén involucradas las letras A, C, G y T. Las demás letras pueden ser eliminadas. Escribe un programa que, dada una secuencia alterada, encuentre la cantidad **t** de tripletas formadas correctamente y la cantidad **f** de letras que falten para formar un correcto código genético.

Entrada: Una cadena de **n** letras mayúsculas, con $1 \leq \mathbf{n} \leq 1{,}000{,}000$.

Salida: Un número entero **t** y un número entero **f**.

Evaluación: 1 punto por el valor correcto de **t** y 1 punto si **f** es correcto.

Ejemplos de entrada	*Ejemplos de salida*
GATSW**AC**Q**A**ERY**CA**KOP**CACT**	4 0
GATSW**AC**Q**A**ERY**CA**KOP**CAC**N	3 1

Problema 2: Números reflejados en el espejo (30 puntos)
Código fuente: nre.c, nre.cpp, nre.java

Tienes una calculadora en la que escribes un número y un espejo vertical en el que puedes reflejarlo:

Algunas veces el número escrito y el reflejado son el mismo (algunas veces no y otras veces lo que se refleja ni siquiera es un número). Escribe un programa que, dados dos enteros **m** y **n**, encuentre la cantidad **r** de enteros en el intervalo de **m** a **n** (incluyéndolos a ambos) que sean iguales a sus números reflejados en el espejo. Por ejemplo, si **m** = 7 y **n** = 12 entonces **r** = 2 puesto que los números 8 y 11 son iguales a sus números reflejados en el espejo. Observa que el número 12851 también es igual a su número reflejado en el espejo.

Entrada: Dos números enteros **m** y **n** con $0 \leq \mathbf{m} \leq \mathbf{n} \leq 999{,}999{,}999$.

Salida: Un número entero **r**.

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

Ejemplo de entrada	*Ejemplo de salida*
7 12	2

Problema 3: Matrices binarias giradas (40 puntos)
Código fuente: mbg.c, mbg.cpp, mbg.java

Una matriz binaria tiene todas sus entradas iguales a 0 o a 1. Dos matrices **A** y **B** tienen una entrada común si $\mathbf{A}_{ij} = \mathbf{B}_{ij}$. Si además permitimos que esas matrices se giren o reflejen de todas las formas posibles, entonces la cantidad de entradas comunes puede cambiar. Escribe un programa que lea dos matrices binarias cuadradas y que encuentre la cantidad máxima **p** y mínima **q** de entradas comunes cuando se permite girar o reflejar cualquiera de las dos matrices de cualquiera de las formas posibles.

Entrada: Un número entero **n** seguido de dos matrices **A** y **B** de $\mathbf{n} \times \mathbf{n}$. Puedes suponer que $1 \leq \mathbf{n} \leq 100$.

Salida: Un número entero **p** y un número entero **q**.

Evaluación: 2 puntos por el valor correcto de **p** y 2 puntos por el valor correcto de **q**.

Ejemplo de entrada	*Ejemplo de salida*
3 **0 0 1** **1 0 0** **0 0 1** 0 1 0 0 0 1 1 0 0	7 3

Problema 4: Euclides y Euler (50 puntos)
Código fuente: eye.c, eye.cpp, eye.java

Dados dos enteros positivos **p** y **q**, su máximo común divisor es el entero positivo más grande que los divide a ambos exactamente y se puede calcular con el algoritmo de Euclides (el cual puedes investigar cómo funciona casi en cualquier lugar). Si el máximo común divisor de **p** y **q** es igual a 1, entonces se dice que **p** y **q** son primos relativos. La función φ(**n**) de Euler es igual a la cantidad de enteros en el intervalo del 1 al **n** que son primos relativos con **n**. Por ejemplo, φ(9) = 6 debido a que 9 es primo relativo con los seis enteros 1, 2, 4, 5, 7 y 8. Escribe un programa que, dados dos enteros **p** y **q**, encuentre el máximo valor **m** que toma la función φ(**n**) de entre todos los enteros **n** en el intervalo de **p** a **q**, además del menor valor de **n** en ese intervalo tal que φ(**n**) = **m**. Por ejemplo, si **p** = 4 y **q** = 10 entonces φ(4) = 2, φ(5) = 4, φ(6) = 2, φ(7) = 6, φ(8) = 4, φ(9) = 6 y φ(10) = 4 por lo que **m** = 6 y **n** = 7.

Entrada: Dos números enteros **p** y **q** tales que $1 \leq \mathbf{p} \leq \mathbf{q} \leq 1{,}000{,}000$.

Salida: Un número entero **m** y un numero entero **n**.

Evaluación: 2 puntos por el valor correcto de **m** y 3 puntos por el valor correcto de **n**.

Ejemplo de entrada	*Ejemplo de salida*
`4 10`	`6 7`

Problema 5: Uno, dos, tres (60 puntos)
Código fuente: udt.c, udt.cpp, udt.java

Imagina que estás parado en el punto (0, 0) de un plano cartesiano. A partir de allí, puedes dar un paso de tamaño 1 en alguna dirección, luego un paso de tamaño 2 en alguna dirección, después un paso de tamaño 3 en alguna dirección, etc. Las cuatro direcciones posibles son arriba, abajo, izquierda y derecha. El objetivo es que llegues al punto de coordenadas (**a**, **b**) en la menor cantidad **n** de pasos que te sea posible. Escribe un programa que te ayude a encontrar un posible camino de (0, 0) a (**a**, **b**) que cumpla esas condiciones.

Entrada: Dos números enteros **a** y **b** tales que $-1{,}000 \leq \mathbf{a}, \mathbf{b} \leq 1{,}000$.

Salida: Un número entero **n** seguido de **n** parejas de enteros $\mathbf{x}_i$, $\mathbf{y}_i$ separados por espacios, las cuales corresponden con los **n** puntos a los que llegas después de cada paso.

Evaluación: 1 punto si los **n** pasos indicados cumplen las condiciones pedidas. En ese caso, $5\mathbf{m}/\mathbf{n}$ puntos adicionales, donde **m** es la cantidad mínima de pasos necesaria para llegar de (0, 0) a (**a**, **b**). El primer ejemplo recibiría 1+5*3/3 = 6 puntos mientras que el segundo recibiría 1+5*3/7 = 3 puntos.

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

`1 1`	`3`
	`0 1`
	`-2 1`
	`1 1`
		`7`
		`-1 0`
		`-2 -1`
		`1 -1`
			`1 -5`
			`1 0`
			`1 -6`
			`1 1`

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