Noveno Concurso Local de Programación ACM ICPC

Universidad Autónoma Metropolitana

Examen eliminatorio, 25 a 29 de mayo de 2012

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 `uam12acm@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 29 de mayo de 2012 a las 22:00.

Problema 1: Año bisiesto marciano (20 puntos)
Código fuente: abm.c, abm.cpp, abm.java

Finalmente los humanos han acondicionado Marte para poder vivir en él y hay muchas personas que desean vivir allá. Sin embargo para poder organizar adecuadamente la actividad humana en dicho planeta se necesita elegir un calendario. La gente que vivirá en Marte ha votado por un calendario parecido al de la Tierra por lo que su año (que tiene una duración de 686 días) también se dividirá en 12 meses y las semanas tendrán 7 días.

Se ha decidido que en Marte todos los meses tengan 57 días, excepto febrero que tendrá 59 días. Desafortunadamente algo parecido al año bisiesto también existirá en aquel planeta y cuando el año sea múltiplo de cinco entonces febrero tendrá un día menos. Ya que se necesita empezar a coordinar las actividades futuras en el planeta vecino te han pedido que escribas un programa que dada una fecha marciana te indique qué día de la semana es. En Marte el 1 de enero de 2001 será considerado lunes.

Entrada: Tres enteros **D**, **M**, **A** que son el día, mes y año marcianos. Puedes suponer que $1 \leq \mathbf{D} \leq 59$, $1 \leq \mathbf{M} \leq 12$, $2001 \leq \mathbf{A} \leq 100000$ y que la fecha es válida.

Salida: Un entero $1 \leq \mathbf{S} \leq 7$ que corresponde al día de la semana, donde **S** = 1 significa lunes, **S** = 2 significa martes, etc.

Evaluación: 2 puntos por la salida correcta.

Ejemplo de entrada	*Ejemplo de salida*
`1 1 2001`	`1`

Problema 2: Segmentos con puntos intermedios (30 puntos)
Código fuente: spi.c, spi.cpp, spi.java

Desde chico has sido muy bueno trazando rectas y crees saber todo sobre ellas. Hace poco encontraste un papel con muchos puntos dibujados en él y como has estado aburrido, te gustaría saber algo sobre los segmentos que podrías formar usando dichos puntos.

Como sabes, un segmento está descrito por dos puntos **A** y **B** llamados extremos. Sin embargo para que sea más divertido, te has impuesto como condición considerar sólo aquellos segmentos que además pasan por alguno de los demás puntos. Para nosotros el segmento **ABC** será el segmento con extremos **A**, **C** y que además pase por el punto **B**.

Escribe un programa que dadas las coordenadas de **N** puntos calcule el número de segmentos diferentes que cumplan la condición anterior. Sabes que los segmentos **ABC** y **CBA** son en realidad el mismo por lo que sólo lo considerarás una vez, pero considerarás diferentes a los segmentos **ABC** y **ADC** siempre y cuando **B** ≠ **D**.

Entrada: Un entero **N** seguido de **N** coordenadas enteras **X**, **Y**. Puedes suponer que 0 ≤ **N** ≤ 1000, 0 ≤ **X**, **Y** ≤ 1000000 y que todos los puntos tendrán coordenadas diferentes.

Salida: Un entero que sea el número de segmentos diferentes que se cumplen las condiciones anteriormente mencionadas.

Evaluación: 3 puntos por el valor correcto .

Ejemplo de entrada	*Ejemplo de salida*
6 0 0 1 1 1 3 3 3 3 1 2 2	5

Problema 3: Bajo el cielo estrellado (40 puntos)
Código fuente: bce.c, bce.cpp, bce.java

De grande te gustaría ser astrónomo y muchas veces te has preguntado cuántas estrellas hay en el universo. Te queda claro que esta pregunta no la podrás responder únicamente mirando hacia el cielo, pues es posible que el brillo de una estrella cercana oculte a las que están atrás. Lo que sí podrás contestar es cuántas estrellas visibles hay en el firmamento.

Como nosotros miramos el cielo desde abajo, diremos que el brillo de una estrella con coordenadas $\mathbf{X_0}$, $\mathbf{Y_0}$ oculta a todas las estrellas que están arriba (es decir, a aquéllas con coordenadas $\mathbf{X_0}$, $\mathbf{Y_i}$ con $\mathbf{Y_i} > \mathbf{Y_0}$). Escribe un programa que dadas las coordenadas de **N** estrellas calcule las **E** estrellas visibles en el firmamento.

Entrada: Un entero **N** seguido de **N** coordenadas enteras **X**, **Y**. Puedes suponer que 0 ≤ **N**, **X**, **Y** ≤ 1000000 y que todas las estrellas tendrán coordenadas diferentes.

Salida: Un entero **E** seguido de las coordenadas **X**, **Y** de las **E** estrellas visibles, una por línea y ordenadas de forma creciente con respecto a la componente **X**. Las componentes de las coordenadas deberán ir separadas por un espacio.

Evaluación: 4 puntos por la salida correcta.

Ejemplo de entrada	*Ejemplo de salida*
4 0 7 1 5 3 7 0 5	3 0 5 1 5 3 7

Problema 4: Sumando volúmenes de cubos (50 puntos)
Código fuente: svc.c, svc.cpp, svc.java

Por alguna razón te gustan mucho los cubos y te la pasas jugando horas con ellos. Como todos sabemos, un cubo es una figura tridimensional con ocho vértices aunque si el cubo está orientado con respecto a los ejes sólo se necesitan conocer dos vértices que sean opuestos para poder representarlo (dos vértices son opuestos si sus coordenadas no tienen componentes en común). Te gustaría calcular el volumen total que ocuparían **N** cubos si para cada cubo conoces las coordenadas de dos vértices opuestos. Desafortunadamente los cubos pueden intersectarse por lo que debes cuidar que el volumen de la intersección sólo sea contado una vez.

Entrada: Un entero **N** seguido de **N** líneas donde cada línea contiene seis enteros $\mathbf{X}_1$, $\mathbf{Y}_1$, $\mathbf{Z}_1$, $\mathbf{X}_2$, $\mathbf{Y}_2$, $\mathbf{Z}_2$ que son las coordenadas de dos vértices opuestos de un cubo orientado con respecto a los ejes. Puedes suponer que $0 \leq \mathbf{N} \leq 100$ y que las componentes de las coordenadas estarán en un rango de -100 a 100.

Salida: Un entero **V** que sea la suma del volumen ocupado por los **N** cubos.

Evaluación: 5 puntos por la salida correcta.

Ejemplo de entrada	*Ejemplo de salida*
2 0 0 0 4 4 4 2 2 2 6 6 6	120

Problema 5: Cola bancaria de la suerte (60 puntos)
Código fuente: cbs.c, cbs.cpp, cbs.java

Eres cuentahabiente del banco más popular de la región y por lo general la cola del banco es extremadamente larga. Además, cada cuentahabiente tiene una prioridad entera **P** y la política es que los clientes con prioridad más alta sean atendidos primero (aunque hayan llegado a la cola después) lo que puede ocasionar que algunos cuentahabientes con baja prioridad no sean atendidos.

Para solucionar esto y volver menos monótona la espera en la cola, el banco ha decidido realizar sorteos para que ciertos clientes avancen más rápido. Durante el sorteo se extrae de una tómbola una letra y un entero **I**: todos los clientes que estén formados en ese momento y cuyos nombres inicien con esa letra tendrán su prioridad incrementada (o decrementada si **I** es negativo) en **I** puntos. En caso de que dos cuentahabientes tengan la misma prioridad se atenderá al que haya llegado primero a la cola. Los eventos que suceden en la cola se pueden representar de la siguiente forma:

- **C N P**: llega un cuentahabiente de nombre **N** con prioridad inicial **P**.
- **S L I**: se realizó un sorteo y se extrajo la letra **L** y el entero **I**.
- **A**: se atendió al cuentahabiente de mayor prioridad de la cola.

Entrada: Un entero **E** seguido de **E** eventos. Puedes suponer $1 \leq \mathbf{E} \leq 50000$, que no se atenderá a nadie mientras la cola esté vacía, que la prioridad de los cuentahabientes nunca saldrá del intervalo de -1000000 a 1000000 y que tanto los nombres como las letras del sorteo incluirán únicamente letras mayúsculas, siendo los nombres no mayores a 16 caracteres.

Salida: Los nombres de los cuentahabientes atendidos en el orden en el que se atendieron. Los nombres deberán estar separados por un salto de línea.

Evaluación: 6 puntos por la salida correcta.

Ejemplo de entrada	*Ejemplo de salida*
6 C CARLOS 8 C JORGE 5 S J 10 C JUAN 7 A A	JORGE CARLOS

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