Cuarto Concurso Local de Programación ACM ICPC

Universidad Autónoma Metropolitana Unidad Azcapotzalco

Examen final, 23 de octubre de 2007

Inicio: 13:00 Fin: 16:00

Instrucciones: Abajo 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. Que tu programa funcione con el ejemplo no quiere decir que funcionará siempre. Para cada problema que resuelvas deberás enviar por correo el código fuente de un programa a la dirección `uam07acm@gmail.com` indicando claramente tu nombre completo. De preferencia envía todos tus códigos en un 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, 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 `javac` de la siguiente manera:

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

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

Problema 1: Experimento con microbios (100 puntos)

Código fuente: `ecm.c`, `ecm.cpp`, `ecm.java`

Archivo de entrada: `ecm.ent` Archivo de salida: `ecm.sal`

Un grupo de biólogos computacionales ha diseñado un experimento para decidir si una colonia de microbios es capaz de resolver problemas cuando se le estimula de ciertas formas específicas. En este experimento se construye un recipiente con la forma de una cuadrícula rectangular con **m** renglones y **n** columnas. En cada uno de los cuadritos de la cuadrícula se coloca cierta cantidad de un compuesto químico que le es muy desagradable a los microbios y que, por lo tanto, los microbios preferirían evitar lo más posible. El recipiente está inclinado de tal forma que los microbios no pueden avanzar en direcciones arbitrarias sino que, guiados por la fuerza de gravedad, sólo pueden avanzar hacia abajo o hacia la derecha del cuadrito en donde estén localizados. Por supuesto, los microbios tampoco pueden salir del recipiente. Al principio la colonia de microbios está localizada en el cuadrito correspondiente al primer renglón y primera columna del recipiente. Al final se espera que la colonia de microbios termine en el cuadrito correspondiente al último renglón y última columna del recipiente. En términos de un sistema de coordenadas, los microbios comienzan en la coordenada (1, 1) y terminan en la coordenada (**m**, **n**). Antes de llevar a cabo el experimento, los científicos desean calcular la cantidad **c** de unidades del compuesto químico que deberá soportar la colonia en su trayecto, esto es, la suma de todas las cantidades del compuesto químico que fueron depositadas en todos los cuadritos por los que pasen. En el ejemplo se muestra un recipiente donde el mínimo valor posible de **c** es 17.

Entrada: El archivo de texto `ecm.ent` contendrá dos enteros **m** y **n** separados por un espacio, seguidos de **m** renglones cada uno con **n** enteros separados por espacios. Estos enteros representan las cantidades del compuesto químico. Puedes suponer que $2 \leq \mathbf{m} \leq 100$, $2 \leq \mathbf{n} \leq 100$ y que todas las demás cantidades están entre 1 y 9, incluyéndolos.

Salida: El archivo de texto `ecm.sal` deberá contener un entero **c** seguido de un renglón con **m** + **n** – 2 enteros separados por espacios. Estos enteros representan un trayecto: un 0 representa ir hacia abajo y un 1 representa ir hacia la derecha.

Evaluación: 1 punto si el trayecto va de (1, 1) a (**m**, **n**) sin salir del recipiente. Si **c** es la cantidad del compuesto químico en ese trayecto, 1 punto adicional y además (8*__min__)/**c** puntos donde **min** es la cantidad mínima de unidades del compuesto químico que deberá soportar la colonia en el mejor trayecto. El primer ejemplo mostrado abajo recibiría 1+1+(8*17)/17 = 10 puntos, el segundo ejemplo recibiría 1+1+(8*17)/29 = 6 puntos y el tercer ejemplo recibiría 1+0+0 = 1 punto.

Ejemplo de archivo de entrada	Ejemplos de archivos de salida
3 4 3 1 4 1 5 9 2 6 1 2 3 4	17 1 1 0 0 1
	29 0 1 1 1 0
	17 0 1 1 1 0

Problema 2: Reinas blancas y negras (100 puntos)

Código fuente: `rbn.c`, `rbn.cpp`, `rbn.java`

Archivo de entrada: `rbn.ent` Archivo de salida: `rbn.sal`

Imagina que dos jugadores tienen un tablero de ajedrez de **n** por **n**. El primer jugador coloca una cierta cantidad **p** de reinas blancas sobre el tablero. El segundo jugador debe de colocar el máximo número posible **q** de reinas negras sobre el tablero de modo que ninguna reina blanca ataque a ninguna reina negra. Es posible que las reinas blancas se ataquen entre sí, pero está prohibido que las reinas negras se ataquen entre sí. Recuerde que dos reinas se atacan si están en el mismo renglón, en la misma columna o en la misma diagonal. Sobre el tablero, las reinas blancas se representan con un 1 y las negras con un 2, mientras que los espacios sin reinas se representan con un 0. En el ejemplo se muestra un tablero de 4 por 4 con 2 reinas blancas. Sobre ese tablero se puede poner un máximo de 2 reinas negras.

Entrada: El archivo de texto `rbn.ent` contendrá un entero **n** seguido de **n** renglones con **n** enteros iguales a 0 ó 1 separados por espacios. Puedes suponer que $1 \leq \mathbf{n} \leq 16$ y que siempre se podrá colocar al menos una reina negra.

Salida: El archivo de texto `rbn.sal` deberá contener dos enteros **p** y **q** separados por un espacio seguidos de **n** renglones con **n** enteros iguales a 0, 1 ó 2 separados por espacios.

Evaluación: 1 punto por el valor correcto de **p**. Si el tablero representado en la salida tiene a las **p** reinas blancas en las posiciones originales y adicionalmente tiene **q** reinas negras satisfaciendo las propiedades pedidas entonces se otorgarán (9***q**)/**max** puntos, donde **max** es el número máximo de reinas negras que se pueden colocar en el tablero original. El primer ejemplo mostrado abajo recibiría 1+(9*1)/2 = 5 puntos, mientras que el segundo ejemplo recibiría 1+(9*2)/2 = 10 puntos.

Ejemplo de archivo de entrada	Ejemplos de archivos de salida	
4 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0	2 1 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 2	2 2 0 0 1 0 1 0 0 0 0 0 0 2 0 2 0 0

Problema 3: El anillo del pirata (100 puntos)

Código fuente: `adp.c`, `adp.cpp`, `adp.java`

Archivo de entrada: `adp.ent` Archivo de salida: `adp.sal`

Una expedición submarina ha encontrado un galeón en el fondo del mar. En éste encontraron un anillo de oro pero no encontraron ninguna otra cosa de valor. Sin embargo, el anillo tenía una inscripción que convenció a la expedición de que el dueño del anillo era un pirata y que él había dejado las instrucciones para encontrar su tesoro escritas en el mismo anillo. Primero descubrieron el lugar desde donde debían comenzar la búsqueda del tesoro. Puedes suponer que ese lugar está en el punto con

coordenadas (0, 0) de un plano. Luego descubrieron que una secuencia de letras alrededor del anillo indicaba en qué dirección debían dar pasos hasta encontrar el sitio donde estaba enterrado el tesoro. Las letras eran, por supuesto, N, E, S y O. Una N significa dar pasos al norte, es decir, incrementar la coordenada y actual. Una E significa dar pasos al este, es decir, incrementar la coordenada x actual. Una S significa dar pasos al sur, es decir, decrementar la coordenada y actual. Una O significa dar pasos al oeste, es decir, decrementar la coordenada x actual. Finalmente descubrieron que el primer paso debía ser de longitud 1, el segundo paso de longitud 2, el tercer paso de longitud 3, etc. Por ejemplo, la secuencia SESOS significa que el tesoro debería estar en la coordenada

$$(0, 0) – (0, 1) + (2, 0) – (0, 3) – (4, 0) – (0, 5) = (-2, -9).$$

Pero el problema es que la secuencia es circular, así que no tienen forma de saber en qué lugar deben comenzar a leerla (SESOS, ESOSS, SOSSE, OSSES, SSESO) y por lo tanto sólo les queda el remedio de cavar en todos los lugares posibles. En nuestro ejemplo hay cinco lugares posibles: (-2, -9), (-2, -11), (3, -8), (3, -10) y (-2, -7). Se desea saber la cantidad **n** de lugares en donde se debe cavar.

Entrada: El archivo de texto `adp.ent` contendrá una cadena de caracteres que contiene únicamente letras N, E, S, O. Puedes suponer que la cadena tendrá al menos 1 y a lo más 1,000 caracteres.

Salida: El archivo de texto `adp.sal` deberá contener un entero **n** seguido de **n** renglones cada uno con una pareja de enteros **x**, **y** separados por espacios. Las parejas de enteros **x**, **y** que representan las coordenadas de los puntos donde se debe cavar deben aparecer ordenadas de menor a mayor por su coordenada **x**. En caso de empate deben aparecer ordenadas de menor a mayor por su coordenada **y**.

Evaluación: Si las **n** parejas encontradas son **distintas** y correctas entonces se otorgarán (9*__n__)/__max__ puntos, donde **max** es el número correcto de lugares donde se debe cavar. Si **n** = **max** se otorgará 1 punto adicional. El primer ejemplo mostrado abajo recibiría 1+(9*5)/5 = 10 puntos, el segundo ejemplo recibiría 0+(9*2)/5 = 3 puntos, mientras que el tercero, el cuarto y el quinto ejemplos recibirían 0 puntos cada uno (el tercero porque las parejas no están en orden, el cuarto porque una pareja es incorrecta y el quinto porque contiene parejas repetidas).

<table>
<tr><th>Ejemplo de
archivo de entrada</th><th>Ejemplos 1 y 2 de
archivos de salida</th><th>Ejemplos 3, 4 y 5 de
archivos de salida</th></tr>
<tr><td rowspan="3">SESOS</td><td rowspan="2">5
-2 -11
-2 -9
-2 -7
3 -10
3 -8</td><td>2
3 -8
-2 -9</td></tr>
<tr><td>2
-2 -9
3 1</td></tr>
<tr><td>2
-2 -9
3 -8</td><td>2
-2 -9
-2 -9</td></tr>
</table>