Sexto Concurso Local de Programación ACM ICPC

Universidad Autónoma Metropolitana Unidad Azcapotzalco

Examen final, 12 de octubre de 2009

Inicio: 10:00 Fin: 13:00

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. 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 `uam09acm@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, 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: Los dos lados del hotcake (100 puntos)

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

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

José sigue siendo muy buen cocinero, pero también sigue siendo desordenado. Por ejemplo, cuando hace hotcakes todos le quedan de tamaños distintos. Además, a veces se le quema más el lado de arriba del hotcake y en otras el lado de abajo. Como al final debe darle una buena presentación a todo lo que sirve, con frecuencia tiene que volver a acomodar el platillo que ha preparado. En el caso de los hotcakes él desea que queden apilados de modo que los más pequeños queden encima de los más grandes y que los lados más quemados queden siempre hacia arriba. Como no va a usar las manos para acomodarlos (ya habíamos aclarado que tiene las tiene limpias) decide usar una pala de la siguiente manera: mete la pala abajo de cierta cantidad de hotcakes y de un solo movimiento los voltea de modo que quedan exactamente en el orden inverso de como estaban. Por ejemplo, si preparó **n** = 5 hotcakes y sus tamaños respectivos son -3, 1, -4, 2 y 5 (de arriba hacia abajo en la pila, los positivos tienen el lado quemado hacia arriba y los negativos hacia abajo) entonces puede meter la pala bajo los primeros 4 hotcakes y voltearlos, de modo que ahora quedan en el orden -2, 4, -1, 3, 5. Escribe un programa que le ayude a José a acomodar sus hotcakes en tan pocos movimientos **m** como sea posible.

Entrada: El archivo de texto `dlh.ent` contendrá un entero **n**, seguido de un renglón con **n** enteros $\mathbf{a}_1$, $\mathbf{a}_2$, ..., $\mathbf{a}_n$ (con valores absolutos positivos y distintos) y separados por espacios. Puedes suponer que $1 \leq \mathbf{n} \leq 8$ y que los demás enteros están en el rango de **-n** a **n**.

Salida: El archivo de texto `dlh.sal` deberá contener un entero $\mathbf{m} \leq 3\mathbf{n}$ seguido de **m** enteros $\mathbf{b}_1$, $\mathbf{b}_2$, ..., $\mathbf{b}_m$ separados por espacios, indicando bajo cuántos hotcakes se metió la pala en cada movimiento.

Evaluación: 1 punto si $\mathbf{m} \leq 3\mathbf{n}$ y los **m** movimientos indicados dejan la pila de hotcakes en orden de menor a mayor con los lados quemados hacia arriba. En ese caso, $9\mathbf{M}/\mathbf{m}$ puntos adicionales, donde **M** es la cantidad mínima de movimientos necesaria para acomodar la pila de hotcakes.

Ejemplo de archivo de entrada	Ejemplo de archivo de salida
5 -3 1 -4 2 5	6 4 2 4 3 1 2

Problema 2: Buscando divisiones exactas (100 puntos)

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

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

Se tiene un vector **A** = (a_1, a_2, ..., a_n) con **n** enteros positivos y un vector **B** = (b_1, b_2, ..., b_m) con **m** enteros positivos. Se sabe que todos los elementos de **A** son menores que todos los elementos de **B**.

Dado también un entero positivo **C** se sabe que existe al menos un elemento $\mathbf{a_i}$ y un elemento $\mathbf{b_j}$ tales que $\mathbf{C} = \mathbf{b_j}/\mathbf{a_i}$. Tu labor es encontrar una pareja de valores $\mathbf{a_i}$ y $\mathbf{b_j}$ que satisfagan esta igualdad.

En el ejemplo mostrado abajo **C** = 3 y eso se puede lograr de varias formas, por ejemplo 27/9 o 18/6. Cualquiera de estas dos es una respuesta correcta.

Entrada: El archivo de texto `bde.ent` contendrá tres enteros **n**, **m** y **C** separados por espacios, seguidos de un renglón con **n** enteros positivos $\mathbf{a_1}$, $\mathbf{a_2}$, ..., $\mathbf{a_n}$ separados por espacios, seguidos de un renglón con **m** enteros positivos $\mathbf{b_1}$, $\mathbf{b_2}$, ..., $\mathbf{b_m}$ separados por espacios. Puedes suponer que **n** y **m** están en el rango de 1 a 1,000,000. También puedes suponer que todos los elementos de **A** y de **B** están en el rango de 1 a 4,000,000,000.

Salida: El archivo de texto `bde.sal` deberá contener dos enteros $\mathbf{a_i}$ y $\mathbf{b_j}$ separados por espacios.

Evaluación: 10 puntos si la salida es correcta.

Ejemplo de archivo de entrada	Ejemplo de archivo de salida
10 5 3 3 1 4 1 5 9 2 6 5 3 27 18 28 81 82	6 18

Problema 3: Salto con obstáculos (100 puntos)

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

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

Dado un tablero de ajedrez de **n** por **n** se desea encontrar el mínimo numero **s** de saltos que debe dar un caballo para llegar desde un origen determinado por una letra 'C', hasta un destino determinado por una letra 'D'. Sin embargo, algunas celdas marcadas por un asterisco están bloqueadas y el caballo no puede saltar hacia éstas. Todas las demás celdas están marcadas con un punto.

Recuerden que un salto válido para un caballo consiste en desplazarse una posición a lo largo de un eje y dos posiciones a lo largo del otro eje, pero hay que recordar que el caballo salta, por lo que no importa si para saltar a una posible posición debe saltar sobre otras que están bloqueadas.

Entrada: El archivo de texto `sco.ent` contendrá un entero **n**, seguido de **n** renglones con exactamente **n** caracteres. Puedes suponer que $3 \leq \mathbf{n} \leq 1000$, que hay exactamente una C y una D y que todos los demás caracteres son asteriscos o puntos.

Salida: El archivo de texto `sco.sal` deberá contener un entero **s,** seguido de **s**+1 renglones cada uno con dos enteros positivos $\mathbf{r}_i$, $\mathbf{c}_i$ indicando las posiciones por las que pasó el caballo en orden, desde la posición inicial del caballo hasta la posición final del caballo.

Evaluación: 1 punto si el recorrido mostrado es válido. En ese caso, 9**m**/**s** puntos adicionales, donde **m** es la cantidad mínima de movimientos necesaria para llevar a cabo el recorrido.

Ejemplo de archivo de entrada	Ejemplo de archivo de salida
8 ******** *******. ******** **D***.* ******** ***.*.** ******** ****C***	2 8 5 6 4 4 3