Primer Concurso Local de Programación ACM ICPC
Universidad Autónoma Metropolitana Unidad Azcapotzalco

Examen eliminatorio, 16 de Octubre de 2004

Instrucciones: Abajo encontrarás los enunciados de cinco problemas. Para cada problema que resuelva deberá dejar en el directorio de trabajo `c:\acm` los archivos de un programa (tanto en código fuente como en ejecutable). El nombre de estos archivos es de la forma `xxxNN.zzz`, donde `xxx` es el nombre del problema, `NN` es un número de identificación que se le dará al comenzar el examen y `zzz` es la extensión, según el tipo de archivo. Si así lo desea, puedes resolver cada problema en un lenguaje distinto. Tu programa deberá leer e imprimir exactamente los datos que se indican (ni más ni menos). En particular, tu programa no deberá borrar la pantalla, escribir ningún tipo de letrero, usar la unidad `screen`, la biblioteca `conio`, etc. Cada problema tiene un valor de 40 puntos, los cuales se obtendrán de las salidas que su programa entregue para cada una de 10 entradas distintas. Que su programa funcione con el ejemplo no quiere decir que su programa funcionará siempre.

Cuestiones técnicas: En todas las computadoras debe existir un directorio de trabajo llamado c:\acm (en caso contrario, éste se puede crear con las tres instrucciones (a) `cd c:\` (b) `mkdir acm` (c) `cd acm`). Quienes programen en C o C++ deberán usar el **DJGPP**, el cual se puede ejecutar con la instrucción `rhide` desde el directorio de trabajo. Quienes programen en Pascal deberán usar el **Borland Turbo Pascal**, el cual se puede ejecutar con la instrucción `turbo` desde el directorio de trabajo. Quienes programen en Java deberán usar el **Sun Java SDK**. Si algo no funciona, repórtalo inmediatamente.

```
/* :-) Los organizadores te deseamos éxito en tu examen (-: */
```

Problema 1: Extremos de una función cúbica

Código fuente: `efcNN.c, efcNN.cpp, efcNN.java, efcNN.pas`
Programa ejecutable: `efcNN.exe`

Escriba un programa que determine el valor máximo **p** y el valor mínimo **q** que toma la función $f(x) = \mathbf{a}x^3 + \mathbf{b}x^2 + \mathbf{c}x + \mathbf{d}$ en los enteros que se encuentran en el intervalo dado por **m** y **n** (incluyéndolos). Además, deberá determinar los puntos **r** y **s** en los cuales se alcanzan estos valores (es decir, tales que $f(\mathbf{r}) = \mathbf{p}$ y $f(\mathbf{s}) = \mathbf{q}$). En caso de que haya varios puntos **r** y **s** con esa propiedad, se deben dar aquellos cuyos valores sean los menores posibles.

Entrada: Seis números enteros **a**, **b**, **c**, **d**, **m**, **n** (con $\mathbf{m} \leq \mathbf{n}$) separados por espacios y todos ellos en el intervalo de -1000 a 1000 (incluyéndolos).
Salida: Cuatro números enteros **p**, **q**, **r**, **s**, separados por espacios.
Evaluación: 1 punto por cada uno de **p**, **q**, **r**, **s**.

Ejemplo de entrada	Ejemplo de salida
`1 0 -12 1 -2 3`	`17 -15 -2 2`

Problema 2: Triángulos de lados pares e impares

Código fuente: `tpiNN.c, tpiNN.cpp, tpiNN.java, tpiNN.pas`
Programa ejecutable: `tpiNN.exe`

Tres números positivos **a**, **b**, **c** son las longitudes de los lados de un triángulo si **a** + **b** > **c**, **b** + **c** > **a** y **c** + **a** > **b**. Escriba un programa que determine la cantidad de triángulos distintos (dos triángulos son distintos si no se pueden girar de ninguna forma para que coincidan) tales que las longitudes de sus lados son números enteros impares (llamemos a este número **p**) y la cantidad de triángulos distintos tales que las longitudes de sus lados son números enteros pares (llamemos a este número **q**). En ambos casos, las longitudes de los lados deben estar en el intervalo dado por **m** y **n** (incluyéndolos).

Entrada: Dos números enteros **m**, **n**, separados por un espacio, tales que 1 ≤ **m** ≤ **n** ≤ 1000.
Salida: Dos números enteros **p**, **q** separados por un espacio.
Evaluación: 2 puntos por el valor de **p** y 2 puntos por el valor de **q**.

Ejemplo de entrada	Ejemplo de salida
3 5	4 1

Problema 3: Segmentos horizontales y verticales

Código fuente: `shvNN.c, shvNN.cpp, shvNN.java, shvNN.pas`
Programa ejecutable: `shvNN.exe`

Escriba un programa que dados dos segmentos (cada uno pudiendo ser horizontal o vertical) determine primero si son horizontales o verticales y luego si se intersectan en algún punto (**x**, **y**) o no. En caso de que los segmentos se intersecten en más de un punto, se deberá determinar el punto (**x**, **y**) cuyas coordenadas sean las menores posibles. En caso de que los segmentos no se intersecten, esto se indicará con **x** = **y** = **-1.** Los segmentos estarán dados por las coordenadas de sus puntos extremos: (**x1**, **y1**) y (**x2**, **y2**) para el primero y (**x3**, **y3**) y (**x4**, **y4**) para el segundo.

Entrada: Ocho números enteros positivos **x1**, **y1**, **x2**, **y2**, **x3**, **y3**, **x4**, **y4**, separados por espacios, todos ellos menores a 1000.
Salida: Cuatro números enteros **h1**, **h2**, **x**, **y**, separados por espacios. El valor de **h1** será 1 si el primer segmento es horizontal y 0 en caso contrario. El valor de **h2** está definido de la misma forma para el segundo segmento.
Evaluación: 1 punto por cada uno de **h1**, **h2**, **x**, **y**.

Ejemplo de entrada	Ejemplo de salida
2 2 4 2 4 1 4 3	1 0 4 2

Problema 4: Números menores que dos mil

Código fuente: `nmdNN.c, nmdNN.cpp, nmdNN.java, nmdNN.pas`
Programa ejecutable: `nmdNN.exe`

Imagine que tiene una lista de números enteros (todos ellos menores que dos mil) a la cual se le han suprimido las separaciones. Por ejemplo, si comenzó con la lista 31, 415, 9, 265 entonces obtuvo la cadena **s** = 314159265. Ahora usted quiere recuperar una lista de números enteros (todos ellos menores que dos mil) agrupando los dígitos tanto como sea posible desde el principio de la cadena. En nuestro caso, obtendrá la lista 314, 1592, 65 con **p** = 3 elementos. Después usted realiza la misma operación pero comenzando desde el final de la lista. En nuestro caso, obtendrá la lista 314, 159, 265 con **f** = 3 elementos. Escriba un programa que dada la cadena de dígitos encuentre cuantos elementos **p** y **f** contienen cada una de las listas.

Entrada: Una cadena **s** que contiene entre 1 y 1000 dígitos.
Salida: Dos números enteros **p** y **f** separados por un espacio.
Evaluación: 2 puntos por el valor de **p** y 2 puntos por el valor de **f**.

Ejemplo de entrada	Ejemplo de salida
314159265	3 3

Problema 5: Matriz con entradas en espiral

Código fuente: `meeNN.c, meeNN.cpp, meeNN.java, meeNN.pas`
Programa ejecutable: `meeNN.exe`

Dados dos enteros **m** y **n**, escriba un programa que construya una matriz con **m** renglones y **n** columnas cuyas entradas sean los números 1, 2, ..., **m*****n** acomodados en espiral, comenzando con el número 1 en la entrada que está en la esquina superior izquierda, siguiendo hacia la derecha, luego hacia abajo, luego hacia la izquierda, luego hacia arriba, y así sucesivamente.

Entrada: Dos números enteros **m**, **n**, separados por un espacio, cuyos valores están entre 1 y 100 (incluyéndolos).
Salida: La matriz requerida (para mayor detalle, ver el ejemplo de salida, aunque no se requiere exactamente el mismo espaciado, sólo el orden).
Evaluación: 4 puntos por la salida correcta.

Ejemplo de entrada	Ejemplo de salida
4 5	1 2 3 4 5 14 15 16 17 6 13 20 19 18 7 12 11 10 9 8

```
/* Deja de leer: Ya no hay más problemas */
```