Curso Acelerado de Diseño de Algoritmos

Universidad Autónoma Metropolitana

Examen diagnóstico, 18 de julio de 2013 (10:00 a 13:00)

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.

Deberás enviar por correo electrónico el código fuente de todos los problemas que resuelvas a la dirección `uam13acm@gmail.com` indicando claramente estos datos: Nombre completo, matrícula, trimestre de ingreso y fecha de nacimiento. 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, hacer pausa, 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 18 de julio de 2013 a las 13:00.

Problema 1: Verificando rectángulos latinos (20 puntos)

Código fuente: vrl.c, vrl.cpp, vrl.java

Un *rectángulo latino* es una cuadrícula de **R** renglones y **C** columnas donde cada cuadrito tiene un entero del 1 al **R**+**C** y además todos los enteros de cada renglón son distintos y todos los enteros de cada columna son distintos. Tu tarea es verificar si una matriz corresponde con un rectángulo latino. Para ello deberás decir si cada renglón y columna están bien. En el ejemplo mostrado abajo, los primeros dos renglones están mal ya que tienen enteros repetidos y el tercer renglón está mal ya que tiene enteros fuera de rango.

Entrada: Dos enteros **R** y **C** seguidos de una matriz de enteros con **R** renglones y **C** columnas. Puedes suponer que $1 \leq \mathbf{R} \leq 100$, $1 \leq \mathbf{C} \leq 100$ y que las entradas de la matriz son enteros entre 1 y 1000.

Salida: Un renglón con **R** enteros, uno por cada renglón de la matriz (0 si el renglón está mal, 1 si el renglón está bien) seguido de un renglón con **C** enteros, uno por cada columna de la matriz (0 si la columna está mal, 1 si la columna está bien).

Evaluación: 1 punto si todos los renglones están bien revisados y 1 punto si todas las columnas están bien revisadas.

Ejemplo de entrada	*Ejemplo de salida*
3 4 3 1 2 3 1 5 6 5 4 9 5 8	0 0 0 1 0 1 0

Problema 2: Prueba del nueve (30 puntos)
Código fuente: pdn.c, pdn.cpp, pdn.java

El romano Hipólito usaba este método para verificar el resultado de la multiplicación **A** × **B** = **C**: Si **A**, **B** o **C** tienen más de un dígito, entonces reemplaza **A**, **B** y **C** por la suma de los dígitos de **A**, **B** y **C**, respectivamente. Repite este proceso hasta que **A**, **B** y **C** tengan sólo un dígito. Ahora se calcula **D** = **A** × **B** y repite el proceso con **D**. Si al final **C** y **D** son iguales, se considera que la multiplicación está "bien". Por ejemplo, si **A** = 314, **B** = 159 y **C** = 46623, entonces **A** toma el valor 8 en el proceso, **B** toma los valores 15 y 6, y **C** toma los valores 21 y 3. Finalmente **D** = 8 × 6 = 48 que luego toma los valores 12 y 3. Ya que **C** = **D** entonces se consideraba que la multiplicación estaba "bien" (aunque en realidad 314 × 159 = 49926).

Entrada: Tres enteros **A**, **B** y **C**. Puedes suponer que cada uno tiene entre 1 y 1000 dígitos.

Salida: Tres enteros: el valor final de **C**, el valor final de **D** y el resultado **R** de la verificación (0 = "mal", 1 = "bien").

Evaluación: 1 punto por cada valor calculado correctamente.

Ejemplos de entrada	*Ejemplo de salida*	*Explicación*
314 159 49926	3 3 1	Al final **A** = 8, **B** = 6, **C** = 3 y **D** = 3, como **C** = **D** se responde "bien".
314 159 46623	3 3 1	Al final **A** = 8, **B** = 6, **C** = 3 y **D** = 3, como **C** = **D** se responde "bien".
314 159 46629	9 3 0	Al final **A** = 8, **B** = 6, **C** = 9 y **D** = 3, como **C** ≠ **D** se responde "mal".

Problema 3: Tarifas de taxi (40 puntos)
Código fuente: tdt.c, tdt.cpp, tdt.java

Generalmente los usuarios de taxis no quedan conformes con la tarifa indicada por los taxímetros pues la mayoría piensa que varios de los taxis en la ciudad utilizan taxímetros alterados. Un taxímetro funciona de la siguiente manera: en el momento que un usuario aborda la unidad, el conductor del taxi presiona un botón que reinicializa el taxímetro marcando una tarifa inicial **B** llamada *banderazo* y a partir de ahí se realiza un incremento de un peso cada que se avanza una distancia **M** de metros o bien cada **S** segundos, lo que ocurra primero. Una vez que se realiza el incremento, los contadores de metros y segundos son inicializados nuevamente a 0. Los valores de **B**, **M** y **S** siempre son visibles al momento de abordar el taxi.

Betito ha desarrollado una aplicación que utiliza el sistema GPS de los celulares y le permite conocer cuál fue el tiempo total del viaje así como la posición, en metros, en la que se encontraba durante cada segundo del recorrido a partir de la posición inicial que siempre se toma como 0. Escribe un programa que, con base en los datos registrados por la aplicación de Betito y los valores conocidos **B**, **M** y **S**, permita verificar a los usuarios de esta aplicación si efectivamente el costo indicado en el taxímetro es el reportado por la aplicación.

Por ejemplo suponga que se tiene **B** = 10, **M** = 10, **S** = 2, que el recorrido total duró **T** = 8 segundos y que la posición del taxi a cada segundo está dada por (7, 11, 18, 20, 45, 49, 49, 59). Suponga que **P** representa el valor indicado por el contador de metros, **Q** el de segundos y **C** el costo actual del recorrido. En $\mathbf{T_1}$ el taxi se encuentra en el metro 7, por lo tanto **P** = 7, **Q** = 1 y **C** = 10; en $\mathbf{T_2}$ se encuentra en la posición 11, con **P** = 1, **Q** = 0 y **C** = 11 (tenga en cuenta que **P** llego a 10 y esto provoca reiniciar ambos contadores, pero **P** requiere sumar el metro adicional avanzado después de los 10 metros); en $\mathbf{T_3}$ tenemos que **P** = 8, **Q** = 1 y **C** = 11; en $\mathbf{T_4}$, **P** = 0, **Q** = 0 y **C** = 12; en $\mathbf{T_5}$, **P** = 5, **Q** = 0 y **C** = 14; en $\mathbf{T_6}$, **P** = 9, **Q** = 1 y **C** = 14; en $\mathbf{T_7}$, **P** = 0, **Q** = 0 y **C** = 15; finalmente en $\mathbf{T_8}$, los contadores terminan con **P** = 0, **Q** = 0 y **C** = 16.

Entrada: Cuatro enteros **B**, **M**, **S**, **T** que son el banderazo, metros, segundos y el tiempo de recorrido, seguidos de una línea con **T** enteros que denotan la ubicación del taxi al tiempo $\mathbf{T_i}$. Puedes suponer que $1 \leq \mathbf{B, M, S} \leq 1000$, $1 \leq \mathbf{T} \leq 10000000$, $0 \leq \mathbf{T_i} \leq 1000000000$, y que todos los valores $\mathbf{T_i}$ están dados en orden ascendente.

Salida: Un entero **C** que corresponde al costo total del viaje.

Evaluación: 4 puntos por la salida correcta.

Ejemplo de entrada	*Ejemplo de salida*
`10 10 2 8` `7 11 18 20 45 49 49 59`	`16`

Problema 4: Consultando el máximo en un intervalo (50 puntos)
Código fuente: cmi.c, cmi.cpp, cmi.java

Un problema muy común en estadística consiste en encontrar el valor máximo de una lista de números. Sin embargo, es frecuente querer encontrar el valor máximo considerando sólo intervalos de la lista. Escribe un programa que, dada una lista **L** de **N** enteros, calcule el valor máximo de cada uno de los **M** intervalos solicitados. Cada intervalo con elementos $\mathbf{L_I}$, $\mathbf{L_{I+1}}$, ..., $\mathbf{L_J}$ estará especificado por la pareja de enteros **I** y **J**.

Entrada: Un entero **N** seguido de los **N** enteros de la lista, seguidos del entero **M** y las **M** parejas de enteros **I** y **J** que denotan un intervalo de la lista cada una. Puedes suponer que $1 \leq \mathbf{N} \leq 750000$, $1 \leq \mathbf{M} \leq 10000$ y que $1 \leq \mathbf{I} \leq \mathbf{J} \leq \mathbf{N}$.

Salida: **M** enteros, uno por línea, que correspondan con el valor máximo de cada uno de los **M** intervalos respectivamente.

Evaluación: 5 puntos si el valor máximo de todos los intervalos solicitados es correcto.

Ejemplo de entrada	*Ejemplo de salida*
10 3 1 4 1 5 9 2 6 5 3 3 1 10 1 3 7 10	9 4 6

Problema 5: Detectando empalme de materias (60 puntos)
Código fuente: dem.c, dem.cpp, dem.java

Por accidente o por mala costumbre, muchos alumnos inscriben materias en horarios que se empalman. Aunque la escuela no puede evitarlo, a los administrativos del proceso de inscripción sí les gustaría detectar estos casos para por lo menos alertar al alumno de este hecho. Ya que los administrativos no son expertos en programación, te han pedido que los ayudes a resolver el problema. Escribe un programa que lea la información de los grupos a los que se inscribirá el alumno y detecte qué grupos tienen horarios que se empalman.

Entrada: Un entero **G** seguido de la información de los **G** grupos del alumno. La información de cada grupo consiste en el nombre del mismo seguido de una cadena de cinco caracteres que representa qué días hay clases en ese grupo seguido de las horas de inicio y fin de clase en el formato HH:MM. Puedes suponer $0 \leq \mathbf{G} \leq 10000$, que el nombre de cada grupo será único de 5 letras mayúsculas y que las horas de inicio y fin serán válidas.

Salida: Un entero **E** que denote el número de grupos que tienen empalmes seguido de los **E** nombres de estos grupos en orden alfabético y separados por saltos de línea.

Evaluación: 1 punto por el valor correcto de **E**, 2 puntos más si la lista de grupos con empalmes es correcta y 3 puntos más si está en orden.

Ejemplo de entrada	*Ejemplo de salida*
4 PROGR L-M-V 10:00 11:30 FISIC LMM-- 11:30 13:00 MATEM ---J- 10:00 11:30 QUIMI --M-V 11:15 13:00	3 FISIC PROGR QUIMI