Problemas propuestos en la I Olimpiada Informática de Santa Cruz de Tenerife

Enero, 2023

Problemas propuestos en la I Olimpiada Informática de Santa Cruz de Tenerife

Volumen 1

Editores

Rafael Herrero-Álvarez, Universidad de La Laguna.
Coromoto León, Universidad de La Laguna.

Fecha de edición

26 de junio de 2023

2023. Universidad de La Laguna. Escuela Superior de Ingeniería y Tecnología. Departamento de Ingeniería Informática y de Sistemas.

Preámbulo

Este documento recoge los enunciados de los problemas propuestos en la *I Olimpiada Informática de Santa Cruz de Tenerife* (OITF), un concurso individual de programación algorítmica para estudiantes de educación secundaria, bachillerato o grado medio, en el que puede participar cualquier centro educativo de esta provincia Canaria.

El objetivo del evento es seleccionar a la persona que junto con el ganador de la *Olimpiada Informática de Las Palmas* constituirá la delegación de la comunidad autónoma de las Islas Canarias en la *Olimpiada Informática Española* (OIE).

En esta edición se propusieron un total de cinco ejercicios que debían resolverse en dos horas haciendo uso de un lenguaje de programación de alto nivel. Los lenguajes permitidos fueron `C++` y `Python`. Las soluciones en Python pueden consultarse en el repositorio de GitHub de la OITF[1].

[1] https://github.com/oliminforcanarias/Soluciones-OITF/tree/master

Organización

Escuela Superior de Ingeniería y Tecnología de la Universidad de La Laguna

Responsables

Coromoto León
Rafael Herrero-Álvarez

Colaboradores

Leopoldo Acosta Sánchez
Rafael Arnay del Arco
Gara Miranda Valladares
Carmen Elvira Ramos Domínguez
Casiano Rodríguez León
Pedro Antonio Toledo Delgado

Centro de Cálculo
Aula Cultural de Pensamiento Computacional

Índice

Ejercicio 1. Verificación en la empresa

1.1. Enunciado

La empresa "Brighter Future" ha encargado a David que implemente un sistema de identificación de los empleados mediante un código alfanumérico de 8 caracteres.

A David se le ha ocurrido que el primer carácter de este código fuese una letra mayúscula, seguido de 7 números. David sabe que se debe verificar la integridad de todo código, por lo que ha añadido una letra de control al final. Ha decidido que esta letra de control se calcule a partir del código alfanumérico utilizando el siguiente algoritmo:

1. Se multiplica cada número del código por un factor de peso que depende de su posición en el código. Los factores de peso que David ha elegido son: 1, 4, 2, 3, 9, 7, 1.
2. Se suman los resultados de las multiplicaciones y se divide el resultado entre 10.
3. El resto de la división anterior se convierte a una letra mayúscula utilizando la siguiente tabla:

Resto	0	1	2	3	4	5	6	7	8	9
Letra	X	D	F	P	L	M	T	H	A	E

Esta es la idea que tiene David, pero no sabe programarla, por lo que ahora te toca a ti. Con el número de empleado de David, S4785112, la letra de control es la D, ya que el algoritmo de verificación sería:

- 4*1 + 7*4 + 8*2 + 5*3 + 1*9 + 1*7 + 2*1 = 81
- 81/10 = 8,1. Como parte entera 8 y de resto 1.
- Teniendo en cuenta la tabla anterior, como el resto es 1, la letra es D.

1.2. Entrada

La empresa le ha pedido a David un programa en el que sea posible introducir un listado de números de empleados, por lo que la entrada consiste en líneas con un único código alfanumérico de 8 caracteres cada una que representa el código del empleado. El primer carácter es una letra mayúscula y los siguientes 7 son números.

Cualquier otra entrada que no tenga longitud 8, que no tenga una letra mayúscula al principio o que lo que sigue a la letra mayúscula no sean números, se considerará un **ERROR**.

1.3. Salida

La salida para este programa será el número del empleado, más la letra correspondiente para cada una de las líneas. Es decir, si recibe 10 números de empleados se debe imprimir sus 10 correspondientes.

Si la entrada no cumple con los criterios expuestos en el apartado anterior, se imprimirá para esa línea un **ERROR**, pero se continuará calculando las demás entradas.

1.4. Entrada de ejemplo

```
S4785112
4785112S
4785112
U6543210
Q0987654789
C5678901
J7890123
N0123456
```

1.5. Salida de ejemplo

```
S4785112D
ERROR
ERROR
U6543210A
ERROR
C5678901E
J7890123P
N0123456L
```

RECUERDA: el sistema espera recibir los datos por la entrada estándar.

Ejercicio 2. Juan y los alienígenas

2.1. Enunciado

Juan es un aficionado a la ciencia ficción y está leyendo un libro sobre una civilización alienígena que utiliza un sistema de numeración completamente diferente al nuestro. Según la historia, en este sistema, el valor de un número se determina multiplicando sus dígitos consecutivos. Por ejemplo, el número 321 tiene un valor de 6, ya que 3 x 2 x 1 = 6.

Juan se ha encontrado con una secuencia de números alienígenas y quiere saber cuál es el valor máximo que puede obtener multiplicando tres dígitos consecutivos. Ayuda a Juan escribiendo un programa que calcule este valor.

2.2. Entrada

Juan espera que la entrada consista en una lista de números enteros. Cada línea representa una numeración alienígena y esta solamente puede contener números enteros. Ningún número será $n < 10^9$.

2.3. Salida

La salida para este programa será un único número entero con el valor máximo que se puede obtener multiplicando tres dígitos consecutivos

En cualquier otro caso, como que haya un mínimo de tres dígitos por línea, que no sea un número entero, que se encuentre un número decimal, negativo o un carácter, se imprimirá un **ERROR**.

Debes tener en cuenta que es posible recibir más de una línea de entrada, por lo que el programa tiene que devolver un valor por la salida para cada una de esas entradas.

2.4. Entrada de ejemplo

```
123
75
.PRT
1111
89763211
11236798
2266755
```

2.5. Salida de ejemplo

```
6
ERROR
```

```
ERROR
1
504
504
252
```

RECUERDA: el sistema espera recibir los datos por la entrada estándar.

Ejercicio 3. Cifrado César con números

3.1. Enunciado

Noelia quiere explicarle a Javi como funciona el cifrado César. Se dice que este cifrado era utilizado por Julio César para comunicarse con sus generales. Su funcionamiento es sencillo y consiste en que, teniendo una cadena de texto, se reemplaza cada uno de sus caracteres por otro que se encuentra un número fijo de posiciones más adelante en el alfabeto.

Noelia sabe que puede complicarse la programación si no se define de manera clara el alfabeto, ya que puede incluir caracteres como signos de puntuación, o de interrogación y exclamación. Por ello, le propone a Javi que lo intente únicamente con números del 0 al 9.

Ejemplo:
Alfabeto: 0 1 2 3 4 5 6 7 8 9
Cadena original: 112
Posición más adelante en el alfabeto: 2
Cadena final: 335

Noelia le explica a Javi que en caso de llegar al final del alfabeto, tendrá que volver a empezar.

Ejemplo:
Alfabeto: 0 1 2 3 4 5 6 7 8 9
Cadena original: 774
Posición más adelante en el alfabeto: 5
Cadena final: 229

3.2. Entrada

Para la entrada de nuestro programa se espera únicamente números enteros positivos de longitudes diferentes, un espacio, y un número. Cualquier otra entrada que no tenga números o incluya alguna letra se considerará un **ERROR**. Ningún número será $n < 10^9$.

Cada línea de la entrada representa a un número diferente, seguido de un espacio y un número que indica la posición más adelante en el alfabeto 0, 1, 2, 3, 4, 5, 6, 7, 8 y 9, por el que se debe reemplazar cada dígito.

3.3. Salida

La salida para este programa será un único número con la misma cantidad de dígitos que el recibido en la entrada, pero modificado según la cantidad de posiciones que se quiere avanzar.

En cualquier otro caso se imprimirá un **ERROR**.

Debes tener en cuenta que es posible recibir más de una línea de entrada, por lo que el programa tiene que devolver un valor por la salida para cada una de esas entradas.

3.4. Entrada de ejemplo

```
123 0
7745 3
34R
1111
1116 -3
88889 14
88889 4.3
88889 5
```

3.5. Salida de ejemplo

```
123
0078
ERROR
ERROR
ERROR
22223
ERROR
33334
```

RECUERDA: el sistema espera recibir los datos por la entrada estándar.

Ejercicio 4. Números perfectos

4.1. Enunciado

María es una apasionada de las matemáticas, se pasa todo el día haciendo cálculos y disfruta con ellos. Recientemente descubrió los números perfectos, aquellos números enteros cuya suma de sus divisores propios es igual al mismo número.

Asombrada por la belleza y perfección de esos números decidió pedirle ayuda a su amigo Julen, que es un crack de la programación, para que le hiciese un programa capaz de detectar si un número es perfecto o no.

Un ejemplo de número perfecto es el 6, donde sus divisores propios son el 3, el 2 y el 1, y su suma (3+2+1) es igual al mismo número, 6.

4.2. Entrada

La entrada de nuestro programa espera únicamente números enteros. En caso de que no se introduzca algo que cumpla esta condición, por ejemplo un número negativo, con decimales o un carácter, se tratará como **ERROR**. Ningún número será $n < 10^9$.

Cada línea de la entrada contendrá un número diferente.

4.3. Salida

Se espera que la salida incluya tres opciones, **PERFECTO**, **NO PERFECTO** o **ERROR**.

Recuerda que el programa recibe varías líneas de entrada y para cada una de ellas espera una línea de salida.

4.4. Entrada de ejemplo

```
6
25
4.3
76
-25.3
28
Maria
```

4.5. Salida de ejemplo

```
PERFECTO
NO PERFECTO
ERROR
```

```
NO PERFECTO
ERROR
PERFECTO
ERROR
```

RECUERDA: el sistema espera recibir los datos por la entrada estándar.

Ejercicio 5. De excursión por Tenerife

5.1. Enunciado

Lola y Ramiro son dos amigos que quieren hacer una excursión por la isla de Tenerife. Como ambos son muy aficionados a la informática, han decidido desarrollar un programa que les ayude a planificar su ruta.

El programa debe leer una lista de puntos de interés que han seleccionado y, a partir de ellos, generar una ruta que pase por todos los puntos de interés y calcule la distancia total recorrida.

Para simplificar el problema, consideraremos que la isla de Tenerife es un tablero de ajedrez **infinito** y que los puntos de interés se encuentran en las casillas del tablero. Cada punto de interés se representará por un par de coordenadas (fila y columna) que indican su posición en el tablero.

5.2. Entrada

La entrada del programa consistirá en líneas con dos números enteros x y y que representan las coordenadas de cada punto de interés. Ningún número será $n < 10^9$.

5.3. Salida

La salida del programa debe ser de dos líneas, la primera con el texto **PUNTOS DE INTERES:** , seguido de un número entero que indique la cantidad de puntos de interés por los que pasaran Lola y Ramiro, y otra línea con el texto **DISTANCIA:** , seguido de un un número entero que representa la distancia total recorrida por la ruta generada.

En cualquier otro caso, como que se introduzcan caracteres, que no se incluya un par de coordenadas (dos números enteros por línea), o que se incluyan números negativos o con decimales, se deberá imprimir **ERROR**.

5.4. Entrada de ejemplo

1.
```
1 2
3 1
2 3
```

2.
```
2
2 1
RT 65
```

3.
```
-42 1.3
6 95 8
```

4.
```
2 1
7 9
```

```
12 9
4 11
```

5.5. Salida de ejemplo

1.
```
PUNTOS DE INTERES: 3
DISTANCIA: 6
```

2.
```
ERROR
```

3.
```
ERROR
```

4.
```
PUNTOS DE INTERES: 4
DISTANCIA: 28
```

RECUERDA: el sistema espera recibir los datos por la entrada estándar.