Problema Etiquetadora defectuosa

En la empresa tenemos un juego de apuestas con dinero ficticio, en donde una vez se define el ganador, se le imprime un tiquete con el monto ganado. Un día, al momento de querer imprimir, nos dimos cuenta de que el dígito **4** no se imprimía. Esto no es algo bueno, dado que siempre el monto es un número que contiene al menos un **4** entre sus dígitos. Por suerte, a una persona se le ocurrió que podíamos entregar dos cheques con montos A y B, tal que **A+B = N**, donde **N** es el monto total del tiquete a pagar. Se necesita entonces formular una función que nos permita imprimir A y B de forma que podamos pagar al ganador.

Input

Monto **N** a pagar a la persona.

Output

Valores **A** y **B** tal que **A+B = N**

Restricciones

1 ≤**N**≤ 10^100.
Al menos un dígito de **N** es **4**

Ejemplo

Entrada: N = 9463
Saluda: A = 6352, B=3111

Problema Alarmas de CPU

Cada vez que se tiene un nuevo servicio, hay que preocuparse de colocar las alarmas correctas para avisar si es que existe algún mal funcionamiento. Queremos implementar una nueva alarma que nos pueda avisar si es que un servicio está consumiendo más CPU de lo habitual. Para ello tenemos una máquina externa que recibe el uso del CPU de la máquina, y dependiendo del algoritmo que apliquemos, poder alertar si es que existe alguna anomalía en el sistema.

Al equipo se le ocurrió que el procedimiento detrás de esto debía ser que cada vez que recibimos un valor **N** de %CPU de la máquina, vamos calcular la mediana **M** de los **E** últimos valores que recibimos anteriormente, y luego vamos a verificar si se cumple que (**N** >= 2**M)**. Si este test sale positivo, esto quiere decir que detectamos una anomalía en el % de uso de CPU y debemos informar al área correspondiente. Para calcular la mediana **M**, se deben ordenar los valores correspondientes de menor a mayor y luego obtener el valor que se encuentra justo al medio. Si **E** es un número par, entonces la mediana en este caso debiese ser el promedio de los 2 valores que se encuentran al medio.

Se pide implementar una función **healthcheck(N,E)** que lleve a cabo el proceso de detectar la anomalía, considerando que se entregan constantemente **N** y **E,** y debe responder **1** si todo se encuentra en orden o **0** si es que existe alguna anomalía. Considere que para los primeros casos, no existen valores, por lo que no debería entregar una alarma hasta tener todos los datos necesarios.

Input

Se entrega **N** el porcentaje de CPU utilizado por la máquina, y **E** la ventana que se requiere inspeccionar.

Output

1 si todo está en orden. 0 si existe anomalía.

Restricciones
N entero tal que 0 <= **N** <= 100
E entero tal que 0 <= **E** <= 10^100

Ejemplo

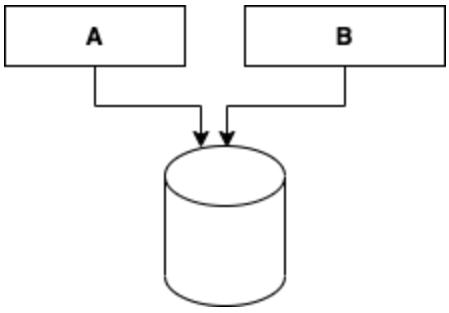
Considerando **E** = 3. Se entregan en orden los valores [2,5,5,12,3,4].

- healthcheck(2,3) => Mediana([2]) <= 2? => Salida(1)
- healthcheck(5,3) => Mediana([2,5]) => Salida(1)
- healthcheck(5,3) => Mediana([2,5,5]) => Salida(1)
- healthcheck(12,3) => Mediana([2,5,5]) => Salida(0) **// Anomalia**
- healthcheck(3,3) => Mediana([5,5,12]) => Salida(1)
- healthcheck(4,3) => Mediana([5,12,3]) => Salida(1)

Problema mails repetidos

Se ha implementado un nuevo sistema de mensajería que utiliza una arquitectura de múltiples servidores para procesar tickets y enviar un mail, indicando el nombre del servidor que lo procesó y el valor que se generó, para luego ser consumido por el cliente. Lamentablemente, luego de una semana desde su partida, se han detectado envío de información errónea y mails repetidos, por lo que se nos asignó para descubrir y arreglar el problema a tiempo.

Supongamos que tenemos una arquitectura de dos servidores y una base de datos en común, en donde los servidores (**A y B**) comparten el mismo código de fuente. Este es un esquema simplificado de la arquitectura:



La estructura de la tabla de **tickets** es (int: id ; string: data ; string: mail ; int: state), donde cada valor representa una columna. El **id** es el identificador autoincremental del ticket el cual es único en la base de datos; **data** es un string que debemos procesar para generar información que será enviada al mail; **mail** es la dirección de correo en

donde debe llegar la información; **state** es un número que indica 1 si está pendiente de procesar y 2 si ya fue procesado.

Es esencial que cada fila sea consumida una única vez dado que se debe enviar un sólo mail al cliente. Para ello se utilizó la siguiente función que procesa los tickets (en pseudocódigo):

```
// Nombre de la maquina guardado
Global machine_uuid;

//  Retorna un string a partir de data
Function processData(string data){
        ….
        return (string)data_modified;
}

// Envía el mail hacia el cliente
Function sendMail(string data){
        …
        return (bool) sent; // True si fue enviado con éxito, False si no
}

function processTickets(){
        (array)pendingTickets = DB(“SELECT * FROM tickets WHERE state = 1”);
        foreach(pendingTickets as ticket) {
                (string)processed_data = processData(ticket->data);
                (string)mail = ticket->id + “ “ + machine_uuid + “ “ + processed_data;
                if(sendMail(data)){
                        DB(“UPDATE tickets SET data = ?, state = 2 WHERE id =
                        ?”,[processed_data, ticket->id]);
                }
        }
}
```

Debe contestar las siguientes preguntas. Considerar que puede realizar cambios no sólo en el código, sino que puede modificar el diagrama y la estructura de la base de datos:

1. ¿Por qué se está enviando dos veces el mail?

2. ¿Qué diferencia hay entre tener 1 o 2 servidores procesando los mails?
3. ¿Cuales son los puntos de fallo en la solución?
4. Reescriba la solución y explique por qué es mejor que la que existe.

Modelo de datos

1. Un colegio necesita un sistema para administrar sus cursos. El sistema tiene que suportar que se le ingresen varios cursos. Cada curso tendrá un profesor a cargo y una serie de alumnos inscritos. Cada profesor, así como cada alumno puede estar en más de un curso. Además cada curso tendrá una cantidad no determinada de pruebas, y el sistema debe permitir ingresar la nota para cada alumno en cada prueba. Todas las pruebas valen lo mismo.

Escriba a continuación las tablas que utilizaría para resolver este problema con los campos y llaves de éstas. Intente hacer el sistema lo más robusto posible, pero sin incluir datos adicionales a los que se plantean acá.

2. Escriba un Query que entregue la lista de alumnos para el curso “programación”.

3. Escriba un Query que calcule el promedio de notas de un alumno en un curso.

4. Escriba un Query que entregue a los alumnos y el promedio que tiene en cada ramo.

5. Escriba un Query que lista a todos los alumnos con **más de un ramo** con promedio rojo.