C12: Colas en el supermercado

Estructuras de Datos
Facultad de Informática - UCM

La fecha tope de entrega de este ejercicio es el **24 de mayo de 2020**.

El ejercicio debe entregarse a través de los problemas `C12-1` y/o `C12-2` de *DOMjudge*. Ambos problemas juzgan el mismo ejercicio. Sin embargo, `C12-2` introduce requisitos más exigentes relativos al coste asintótico de las operaciones, de modo que algunas soluciones que obtengan el veredicto `CORRECT` en `C12-1` pueden obtener un veredicto `TIMELIMIT` en `C12-2`.

La entrega consiste en un único fichero `.cpp` que se subirá a *DOMjudge*. Puedes subir tantos intentos como quieras. Se tendrá en cuenta el último intento con el veredicto `CORRECT` que se haya realizado antes de la hora de entrega por parte de alguno de los miembros del grupo.

No olvides poner el nombre de los componentes del grupo en el fichero `.cpp`. Solo es necesario que uno de los componentes del grupo realice la entrega.

Evaluación: Este ejercicio se puntuará de 0 a 10. Para poder obtener una calificación superior a 0 es necesario obtener un veredicto `CORRECT` en, al menos, el problema `C12-1`. Para obtener una calificación superior a 7 es necesario obtener un veredicto `CORRECT` en `C12-2`.

En el supermercado de mi barrio han implantado un sistema para agilizar las colas que se forman al pasar por caja. El supermercado tiene varias cajas, cada una con su cola, de modo que los clientes pueden pagar en cualquiera de ellas poniéndose al final de la cola correspondiente. Pues bien, el sistema que han implantado intenta determinar, en todo momento, quién está en cada cola. La directora del supermercado dice que este sistema facilita el trabajo al personal, aunque a mi me parece una excusa bastante cogida por los pelos.

En cualquier caso, este sistema de rastreo de colas parece sencillo, pero se les ha ido de las manos, ya que hay mucha gente impaciente (o indecisa, según se mire) que no es capaz de estarse quieta en una cola. Cuando ven que su cola no avanza, deciden salirse de ella y colocarse al final de otra.

El objetivo de este ejercicio consiste en implementar un TAD que permita gestionar las colas de este supermercado. Para ello define una clase `Supermercado` con las siguientes operaciones:

- `Supermercado(num_cajas)`

  Crea un nuevo supermercado sin clientes, y con el número de cajas indicado.

- `nuevo_cliente(cod_cliente, num_cola)`

  Registra un nuevo cliente en el sistema. El cliente está identificado mediante su código de cliente (`cod_cliente`, de tipo `string`). Este cliente se colocará al final de la cola indicada por el parámetro `num_cola`, que es un número comprendido entre 0 y $N-1$, donde N es el número de cajas del supermercado. Si `num_cola` no está en este rango, se lanzará una excepción `domain_error` con el mensaje `Cola inexistente`. Si el cliente a añadir ya estaba registrado en el sistema, se lanzará una excepción `domain_error` con el mensaje `Cliente duplicado`.

- `cambiar_cola(cod_cliente, num_cola)`

  Se llama a este método cuando un cliente decide salirse de su cola para pasarse a la de otra caja. De este modo, ahora será el último de la cola indicada por `num_cola`, excepto si el cliente ya estaba en esa misma cola, en cuyo caso no se haría nada. Si `num_cola` está fuera de rango, se lanzará una excepción `domain_error` con el mensaje `Cola inexistente`. Si el cliente no estaba registrado en el sistema, se lanzará una excepción `domain_error` con el mensaje `Cliente inexistente`.

- `consultar_cliente(cod_cliente)`

  Devuelve el número de cola en la que se encuentra el cliente dado. Si el cliente no está registrado en el sistema lanza una excepción `domain_error` con el mensaje `Cliente inexistente`.

- `cuantos_en_cola(num_cola)`

  Devuelve un número que indica cuántos clientes hay esperando en la cola pasada como parámetro. Si `num_cola` está fuera de rango, se lanzará una excepción `domain_error` con el mensaje `Cola inexistente`.

- `clientes_en_cola(num_cola)`

  Devuelve una lista con los clientes que están esperando en una determinada cola, desde el último hasta el primero. Si `num_cola` está fuera de rango se lanzará una excepción `domain_error` con el mensaje `Cola inexistente`.

No olvides indicar el coste de cada una de las operaciones. Ten en cuenta que todas las operaciones deben implementarse de la manera más eficiente posible, desde el punto de vista del coste asintótico en tiempo.

Ninguno de los métodos de la clase `Supermercado` debe realizar operaciones de E/S. El manejo de E/S debe hacerse de manera externa al TAD.

Entrada

La entrada consta de una serie de casos de prueba. Cada caso de prueba comienza con un número N ($1 \leq N \leq 16$) indicando el número de cajas que hay en el supermercado. Después aparecen una serie de líneas. Cada una de ellas contiene el nombre de una operación del TAD, seguida de sus argumentos. Suponemos que los códigos de clientes no contienen espacios, ni tabuladores, ni caracteres de fin de línea.

Cada caso de prueba finaliza con una cadena `FIN` que no realiza ninguna acción.

Salida

Para las operaciones `consultar_cliente`, `cuantos_en_cola` y `clientes_en_cola` debe mostrarse el resultado de la operación, en el formato que se indicará a continuación. El resto de operaciones no muestran nada por pantalla.

- `consultar_cliente`. En caso de éxito, debe imprimir la cadena `El cliente` c `esta en la cola` n, donde c es el código de cliente y n es el número de cola. Observa que la salida no lleva tildes.
- `cuantos_en_cola`. En caso de éxito, debe imprimir la cadena `En la cola` n `hay` m `clientes`, donde n es el número de la cola y m es la cantidad de clientes que hay en la misma.
- `clientes_en_cola`. En caso de éxito, debe imprimir la cadena `En la cola` n `estan:`, seguida de los códigos de los clientes que están en esa cola, separados por un espacio en blanco. De nuevo n es el número de cola.

En cualquier caso, si alguna de las operaciones produce una excepción, debe imprimirse una línea con el mensaje `ERROR:`, seguida de un espacio en blanco y del mensaje de error correspondiente.

Al final de cada caso de prueba debe imprimirse una línea con tres guiones (`---`).

Entrada de ejemplo

```
4
nuevo_cliente 12A 2
nuevo_cliente 23B 2
nuevo_cliente 34C 2
cuantos_en_cola 2
clientes_en_cola 2
cuantos_en_cola 0
clientes_en_cola 0
cambiar_cola 34C 0
cambiar_cola 12A 0
consultar_cliente 34C
cuantos_en_cola 0
clientes_en_cola 0
clientes_en_cola 2
cambiar_cola 12A 2
clientes_en_cola 0
clientes_en_cola 2
FIN
3
nuevo_cliente 12A 2
nuevo_cliente 12A 1
nuevo_cliente 23B 4
cambiar_cola 12A 3
consultar_cliente 23B
cuantos_en_cola 10
FIN
4
nuevo_cliente 12A 2
nuevo_cliente 23B 2
cambiar_cola 12A 2
clientes_en_cola 2
FIN
```

Salida de ejemplo

```
En la cola 2 hay 3 clientes
En la cola 2 estan: 34C 23B 12A
En la cola 0 hay 0 clientes
En la cola 0 estan:
El cliente 34C esta en la cola 0
En la cola 0 hay 2 clientes
En la cola 0 estan: 12A 34C
En la cola 2 estan: 23B
En la cola 0 estan: 34C
En la cola 2 estan: 12A 23B
---
ERROR: Cliente duplicado
ERROR: Cola inexistente
ERROR: Cola inexistente
ERROR: Cliente inexistente
ERROR: Cola inexistente
---
En la cola 2 estan: 23B 12A
---
```

Créditos

Basado en un ejercicio de Alberto Verdejo.