Estructuras de Datos y Algoritmos

Grado en Desarrollo de Videojuegos. Facultad de Informática (UCM)

Convocatoria extraordinaria. 15 de junio de 2023

Normas de realización del examen

1. El examen dura **3 horas**.
2. La entrega se realiza en el juez automático de los laboratorios accesible desde la url `http://exacrc` (cada ejercicio en su correspondiente problema del juez con el prefijo EDA-Junio23). Para acceder debes usar el usuario/contraseña que has recibido al comienzo del examen.
3. Al principio de cada fichero .cpp debe aparecer, en un comentario, tu nombre y apellidos, dni, usuario del juez y puesto de laboratorio. También debes incluir unas líneas explicando qué has conseguido hacer y qué no.
4. Todo lo que no sea código C++ (explicaciones, respuestas a preguntas, etc.) debe ir en los propios ficheros en comentarios debidamente indicados.
5. Tus soluciones serán evaluadas por el profesor independientemente del veredicto del juez automático. Para ello, el profesor tendrá en cuenta **exclusivamente** el último envío que hayas realizado de cada ejercicio.
6. Puedes acceder a la referencia de C++ en `http://exacrc/cppreference`

Ejercicio 1. Repetir los elementos de una cola (3 puntos)

Se pide implementar una operación interna (como método genérico) en la clase `queue` vista durante el curso que transforme la cola de manera que se repitan sus elementos de acuerdo a las multiplicidades indicadas en un vector de entrada (que tendría la misma longitud que la cola). En concreto, sea `q` la cola y `mult` el vector de multiplicidades:

- Si `mult[i]` = 0 entonces el elemento i-ésimo se elimina de la cola `q`.
- Si `mult[i]` = 1 entonces el elemento i-ésimo de `q` se queda como estaba.
- Si `mult[i]` = v, siendo $v > 1$, entonces se añaden $v - 1$ copias tras el elemento i-ésimo de `q`.

Por ejemplo, si la cola de entrada tiene los elementos $[23, 10, 5, 9]$ y el vector de multiplicidades es $[2, 0, 1, 3]$, entonces tras invocar al método pedido la cola quedaría como $[23, 23, 5, 9, 9, 9]$.

Se valorará la complejidad en tiempo y en espacio de la solución, las cuales debes indicar y justificar. Es importante que se cree la cantidad mínima de nuevos nodos para construir el resultado. Además, no se permite cambiar el valor contenido en los nodos ya existentes en la cola.

Entrada

La entrada comienza con un número indicando cuántos casos vienen a continuación. Cada caso de prueba ocupa tres líneas. La primera línea contiene un número con la longitud de cola (y del vector), la segunda línea contiene los elementos de la cola y la tercera los elementos del vector de multiplicidades, en ambos casos en orden inverso (este aspecto no es importante y queda resuelto en la plantilla proporcionada).

Salida

Para cada caso de prueba debe escribirse una línea con la cola resultante tras llamar al método, delimitada entre corchetes y con sus elementos separados por comas (de nuevo esto queda resuelto en la plantilla proporcionada).

Entrada de ejemplo

```
3
4
9 5 10 23
3 1 0 2
3
1 4 3
5 1 5
3
2 6 1
0 0 0
```

Salida de ejemplo

```
[23, 23, 5, 9, 9, 9]
[3, 3, 3, 3, 3, 4, 1, 1, 1, 1, 1]
[]
```

Ejercicio 2. Fuga de prisión (3 puntos)

Se ha detectado la fuga de un preso de una prisión de alta seguridad. Cada preso se identifica mediante una letra del alfabeto. Se utilizan letras consecutivas y se conoce la primera y la última letra utilizada. Los presos se colocan en línea de manera ordenada y hay que encontrar lo más rápido posible al preso que falta. Se pide implementar un algoritmo eficiente que determine la letra del preso fugado y justificar su coste.

Entrada

La entrada comienza con un número indicando cuántos casos viene a continuación. Cada caso de prueba ocupa dos líneas. En la primera se indican las letras del primer y último preso. En la segunda se indican las letras de los presos presentes en el recuento, ordenados en orden ascendente.

Salida

Para cada caso de prueba debe escribirse una línea con la letra del preso fugado.

Entrada de ejemplo

```
12
a e
a b c e
c h
d e f g h
c h
c e f g h
c h
c d f g h
c h
c d e g h
c h
c d e f h
c h
c d e f g
d h
e f g h
d h
d f g h
d h
d e g h
d h
d e f h
d h
d e f g
```

Salida de ejemplo

```
d
c
d
e
f
g
h
d
e
f
g
h
```

Ejercicio 3. Gestión de una tienda (4 puntos)

Nos piden implementar un sistema para gestionar la adquisición y venta de productos por parte de una tienda. De vez en cuando la tienda recibe nuevos productos identificados por un código (cadena sin espacios) y señalados con una fecha. Las unidades del producto adquirido se guardan en el almacén, salvo que haya clientes que hubieran intentado comprar ese determinado producto cuando no había existencias y se hubieran colocado en la lista de espera. En ese caso, los clientes son servidos en riguroso orden de llegada. También puede haber venta de productos en existencia. En ese caso, se venden siempre las unidades con una fecha menor. El sistema estará encapsulado en un TAD `Tienda` que proporciona las siguientes operaciones:

- constructora: al comienzo el almacén está vacío y no hay clientes en espera.
- `adquirir(COD, F, CANT)`: gestiona la adquisición de `CANT` unidades (un entero no negativo) del producto `COD` con fecha `F`. Si hubiera clientes esperando la llegada de este producto, devuelve un vector con los identificadores de los clientes que han podido ser servidos, en el orden en que hicieron la petición. El resto de unidades (si las hay) se guardan en el almacén.
- `vender(COD, CLI)`: gestiona la venta de una unidad del producto `COD` al cliente `CLI` (un string sin espacios). Si hay existencias, la operación devuelve `true` y la fecha del producto vendido (la menor entre las disponibles). Si no hay existencias, devuelve `false` y añade al cliente a la lista de espera de este producto (un cliente puede aparecer varias veces en la lista de espera).
- `cuantos(COD)`: devuelve cuántas unidades tiene la tienda del producto `COD` (independientemente de la fecha). Si se trata de un producto que nunca se ha adquirido simplemente se devuelve `0`.
- `hay_esperando(COD)`: indica si hay clientes en la lista de espera del producto COD. Si se trata de un producto que nunca se ha adquirido simplemente se devuelve `false`.

Se pide elegir una representación adecuada e implementar de manera eficiente las operaciones del TAD descrito, utilizando los contenedores vistos de la librería estándar de C++. En la plantilla se proporciona el tipo `Fecha` con operadores de orden y de E/S. Indica y justifica brevemente el coste de cada una de las operaciones. Ninguno de los métodos del TAD debe realizar operaciones de E/S (el manejo de E/S se hace en la función `resuelveCaso`).

Entrada

La entrada consta de una serie de casos de prueba. Cada caso consiste en una serie de líneas donde se muestran las operaciones a realizar, sobre una tienda inicialmente vacía: el nombre de la operación seguido de sus argumentos. Cada caso termina con una línea con la palabra FIN.

Salida

Para cada caso de prueba, se escribirá una línea por operación de la siguiente manera:

- `adquirir`: muestra `PRODUCTO ADQUIRIDO` seguido de los códigos de los clientes que estuvieran en la lista de espera (si los había) y hayan podido llevarse una unidad;
- `vender`: si hay existencias en ese momento se muestra `VENDIDO` y la fecha del producto vendido; si no, se muestra `EN ESPERA`;
- `cuantos`: muestra el número devuelto por la operación;
- `hay_esperando`: si hay clientes esperando que llegue ese producto escribe `SI`, y en caso contrario escribe `NO`.

Observa que ninguna operación debe generar error. Cada caso termina con una línea con tres guiones (—).

Entrada de ejemplo

```
vender lapiz Ana
adquirir lapiz 10/06/19 3
vender lapiz Pedro
adquirir boli 20/06/19 3
adquirir boli 15/06/19 2
vender boli Pedro
vender boli Luis
vender boli Marta
cuantos boli
hay_esperando boli
FIN
vender boli Ana
hay_esperando boli
hay_esperando lapiz
vender boli Pedro
vender boli Luis
adquirir boli 20/06/19 2
cuantos boli
hay_esperando boli
FIN
```

Salida de ejemplo

```
EN ESPERA
PRODUCTO ADQUIRIDO Ana
VENDIDO 10/06/19
PRODUCTO ADQUIRIDO
PRODUCTO ADQUIRIDO
VENDIDO 15/06/19
VENDIDO 15/06/19
VENDIDO 20/06/19
2
NO
---
EN ESPERA
SI
NO
EN ESPERA
EN ESPERA
PRODUCTO ADQUIRIDO Ana Pedro
0
SI
---
```