Cues quàntiques X60468_ca

En aquest exercici modificarem la classe `Queue` afegint dos nous mètodes `entangle` i `disentangle` i canviant el comportament del mètode `pop` com descrivim a continuació.
El nou mètode `entangle` rebrà una altra `Queue` com a paràmetre. Una crida `q0.entangle(q1)` provocarà que `q0` quedi enllaçat a `q1` de manera que, a partir de llavors, sempre que fem un `pop` sobre `q0`, l'element extret serà afegit al final de `q1`. Una crida `q0.disentangle()` cancel.larà aquest comportament.
Fixeu-vos en aquest exemple per tal d'acabar d'entendre-ho:

```
Queue<string> q0, q1;
q0.push("a");      // q0: a
q0.push("b");      // q0: a,b
q1.push("c");      // q1: c
q1.push("d");      // q1: c,d
q0.entangle(q1);
q0.pop();          // q0: b  q1: c,d,a
q1.pop();          // q0: b  q1: d,a
q1.entangle(q0);
q0.pop();          // q0:    q1: d,a,b
q1.pop();          // q0: d  q1: a,b
q0.disentangle();
q0.pop();          // q0:    q1: a,b
q1.pop();          // q0: a  q1: b
q1.disentangle();
q0.pop();          // q0:    q1: b
q1.pop();          // q0:    q1:
```

Successius `entangle` fan que només l'últim estigui actiu. Per exemple, si hem executat `q0.entangle(q1)` i després `q0.entangle(q2)`, llavors `q0` està enllaçat a `q2` però no a `q1`.
Una crida `q0.disentangle()` cancel.larà l'efecte de l'últim `entangle` actiu. En altres paraules, `q0` queda com a no enllaçat a cap altra cua.
D'entre els fitxers que s'adjunten en aquest exercici, trobareu `queue.hh`, a on hi ha una implementació de la classe genèrica `Queue`. Haureu d'implementar els dos nous mètodes `entangle` i `disentangle` dins `queue.hh` a la part pública de la classe (podeu trobar les capçaleres comentades dins `queue.hh`), i modificar el mètode `pop` convenientment. Necessitareu també algun atribut addicional per tal de recordar si la cua té un `entangle` actiu i amb qui, amb les convenients inicialitzacions.
D'entre els fitxers que s'adjunten a l'exercici també hi ha `main.cc` (programa principal), i el podeu compilar directament, doncs inclou `queue.hh`. Només cal que pugeu `queue.hh` al jutge.
Observació: La solució ràpida de `pop` amb `entanglement` hauria de moure punters i no pas valors.
Observació: En els jocs de proves no es copiaran cues. Per tant, no cal que adapteu els mètodes que copien cues, de manera que no cal decidir si l'entanglement d'una cua s'hereta sobre una còpia.

Entrada

L'entrada del programa comença amb una declaració d'unes quantes cues d'strings (`q0, q1, ...`), i després té una seqüència de comandes sobre les cues declarades. Com que ja us oferim el `main.cc`, no cal que us preocupeu d'implementar la lectura d'aquestes entrades. Només cal que implementeu la extensió de la classe cua abans esmentada.
Podeu assumir que les comandes faran disentangles només sobre cues que tinguin un entangle actiu. Però pot ser el cas que es faci un entangle sobre una cua que ja tingui un entangle actiu. Com mencionavem abans, en aquestes situacions només l'últim entangle aplica.

Sortida

Per a cada comanda d'escriptura sobre la sortida s'escriurà el resultat corresponent. El `main.cc` que us oferim ja fa això. Només cal que implementeu la extensió de la classe cua abans esmentada.

Exemple d'entrada 1

```
Queue<string> q0 , q1 , q2 ;
q0 .push( "a" );
q0 .push( "b" );
q0 .push( "c" );
q0 .push( "d" );
q1 .push( "e" );
q1 .push( "f" );
q1 .push( "g" );
q1 .push( "h" );
q2 .push( "i" );
q2 .push( "j" );
q2 .push( "k" );
q2 .push( "l" );
q0 .pop();
q1 .pop();
q2 .pop();
cout<< q0 <<endl;
cout<< q1 <<endl;
cout<< q2 <<endl;
cout<< q0 .size()<<endl;
cout<< q1 .size()<<endl;
cout<< q2 .size()<<endl;
cout<< q0 .front()<<endl;
cout<< q1 .front()<<endl;
cout<< q2 .front()<<endl;
q0 .entangle( q1 );
q1 .entangle( q2 );
q2 .entangle( q0 );
q0 .pop();
q1 .pop();
q2 .pop();
q0 .push( "m" );
q1 .push( "n" );
q2 .push( "o" );
cout<< q0 <<endl;
cout<< q1 <<endl;
cout<< q2 <<endl;
cout<< q0 .size()<<endl;
cout<< q1 .size()<<endl;
cout<< q2 .size()<<endl;
q1 .disentangle();
q0 .pop();
q1 .pop();
q2 .pop();
cout<< q0 <<endl;
cout<< q1 <<endl;
cout<< q2 <<endl;
cout<< q0 .size()<<endl;
cout<< q1 .size()<<endl;
cout<< q2 .size()<<endl;
q2 .disentangle();
q0 .pop();
q1 .pop();
q2 .pop();
cout<< q0 <<endl;
cout<< q1 <<endl;
cout<< q2 <<endl;
cout<< q0 .size()<<endl;
cout<< q1 .size()<<endl;
cout<< q2 .size()<<endl;
q0 .disentangle();
q0 .pop();
q1 .pop();
q2 .pop();
cout<< q0 <<endl;
cout<< q1 <<endl;
cout<< q2 <<endl;
cout<< q0 .size()<<endl;
cout<< q1 .size()<<endl;
cout<< q2 .size()<<endl;
cout<< q0 .front()<<endl;
cout<< q1 .front()<<endl;
cout<< q2 .front()<<endl;
```

Exemple de sortida 1

```
3 b c d
3 f g h
3 j k l
3
3
3
b
f
j
4 c d j m
4 g h b n
4 k l f o
4
4
4
4 d j m k
4 h b n c
3 l f o
4
4
3
3 j m k
4 b n c d
2 f o
3
4
2
2 m k
3 n c d
1 o
2
3
1
m
n
o
```

Exemple d'entrada 2

```
Queue<string> q0 , q1 , q2 ;
cout<< q1 <<endl;
q1 .push( "bba" );
cout<< q1 .front()<<endl;
cout<< q1 .size()<<endl;
cout<< q1 <<endl;
q1 .push( "a" );
q1 .push( "caa" );
cout<< q1 .size()<<endl;
q1 .push( "aaa" );
q1 .push( "ba" );
q0 .push( "cbc" );
cout<< q2 <<endl;
cout<< q2 .size()<<endl;
q1 .push( "cac" );
q1 .pop();
q0 .push( "c" );
q2 .push( "bab" );
cout<< q0 <<endl;
cout<< q0 .size()<<endl;
cout<< q2 .size()<<endl;
q2 .pop();
q0 .push( "baa" );
cout<< q2 .size()<<endl;
cout<< q2 <<endl;
cout<< q0 <<endl;
q1 .push( "ba" );
cout<< q1 .front()<<endl;
q0 .push( "caa" );
q2 .push( "cc" );
q0 .pop();
q1 .push( "baa" );
q0 .entangle( q2 );
q2 .push( "b" );
cout<< q1 <<endl;
cout<< q0 <<endl;
q2 .push( "cc" );
q1 .push( "c" );
q0 .disentangle();
q0 .entangle( q1 );
cout<< q2 .front()<<endl;
q0 .pop();
cout<< q2 .size()<<endl;
q1 .pop();
q0 .disentangle();
q2 .pop();
cout<< q1 .front()<<endl;
cout<< q2 <<endl;
q2 .push( "bc" );
cout<< q0 <<endl;
q2 .entangle( q0 );
q0 .pop();
q1 .push( "b" );
q2 .push( "bcc" );
q1 .push( "bcb" );
q1 .push( "caa" );
q0 .push( "bbb" );
q2 .pop();
q1 .push( "ba" );
q0 .entangle( q2 );
q0 .pop();
q0 .push( "c" );
q2 .push( "cb" );
q2 .pop();
q2 .push( "ac" );
q2 .pop();
q0 .pop();
q1 .push( "cca" );
q0 .pop();
cout<< q2 <<endl;
q1 .push( "bab" );
q0 .pop();
q2 .push( "c" );
q0 .pop();
q2 .pop();
q2 .disentangle();
q0 .pop();
cout<< q2 .front()<<endl;
q0 .pop();
```

```
q2 .push( "b" );
cout<< q2 <<endl;
cout<< q0 <<endl;
cout<< q1 <<endl;
cout<< q2 <<endl;
```

Exemple de sortida 2

```
0
bba
1
1 bba
3
0
0
2 cbc c
2
1
0
0
3 cbc c baa
a
7 a caa aaa ba cac ba baa
3 c baa caa
cc
3
caa
2 b cc
2 baa caa
6 bcc caa cb ac bbb b
caa
11 caa cb ac bbb b c c cc bc bcc b
0
14 caa aaa ba cac ba baa c c b bcb caa ba cca bab
11 caa cb ac bbb b c c cc bc bcc b
```

Observació

Avaluació sobre 10 punts:

- Solució lenta: 5 punts.
- solució ràpida: 10 punts.

Entenem com a solució ràpida una que és correcta, on la operació `pop` amb `entanglement` té cost **CONSTANT** (fins i tot si els strings involucrats son grans), i capaç de superar els jocs de proves públics i privats. Entenem com a solució lenta una que no és ràpida, però és correcta i capaç de superar els jocs de proves públics.

Informació del problema

Autoria: PRO2

Generació: 2026-01-27T18:53:42.305Z