Agafar elements del front d'un altre Queue en el mateix ordreX79029_ca

En aquest exercici estendrem la classe `Queue` afegint un nou mètode anomenat `take`. Aquest mètode té, com a paràmetres, un altre `Queue`, i un natural `k`, i la seva crida té com a efecte que movem els `k` elements del front de l'altre `Queue` al final del paràmetre implícit, i en el mateix ordre.
Més específicament, suposem que un `Queue t` té contingut $[a_1, a_2, \ldots, a_n]$ (on els elements els representem en ordre des del front fins el final, i en particular a_1 és l'element del front), i que un altre `Queue q` té contingut $[b_1, b_2, \ldots, b_m]$. Llavors, una crida `t.take(q, k)` té com a efecte que `t` passi a contenir $[a_1, \ldots, a_n, b_1, \ldots, b_k]$ i `q` passi a contenir $[b_{k+1}, \ldots, b_m]$.
Per exemple, si `t` conté $[3, 1, 5]$ i `q` conté $[9, 4, 6, 8, 2]$, llavors la crida `t.take(q, 3)` té com a efecte que `t` passi a contenir $[3, 1, 5, 9, 4, 6]$, i que q passi a contenir $[8, 2]$.
En el cas particular que `k` sigui més gran que `m`, llavors es mouen tots els elements de `s` a `t`. És a dir, `t` passa a contenir $[a_1, \ldots, a_n, b_1, \ldots, b_m]$, i `q` passa a contenir `[]`.
Per exemple, si `t` conté $[3, 1, 5]$ i `q` conté $[9, 4, 6, 8, 2]$, llavors la crida `t.take(s, 10)` té com a efecte que `t` passi a contenir $[3, 1, 5, 9, 4, 6, 8, 2]$, i que q passi a contenir `[]`.
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 de buscar dins `queue.hh` les següents línies:

```
// Pre:  Sigui [a1,...,an] el contingut del paràmetre implícit (des del front
//       Sigui [b1,...,bm] el contingut de q.
//       k>=0
// Post: En el cas en que k>=m, aquest és el resultat:
//          [a1,...,an,b1,...,bm] és el contingut del paràmetre implícit.
//          [] és el contingut de q.
//       En canvi, en el cas k<m, aquest és el resultat:
//          [a1,...,an,b1,...,bk] és el contingut del paràmetre implícit.
//          [b{k+1},...,bm] és el contingut de q.
// Descomenteu les següents dues linies i implementeu el mètode:
// void take(Queue<T> &q, int k) {
// ...
// }
```

Descomenteu les linies que s'indiquen i implementeu el mètode.
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: En aquest exercici es prefereix una solució basada en manegar punters abans que una solució basada en cridar a mètodes primitius de la pròpia classe (`push, pop, front`). De fet, manegar punters serà més ràpid, i fer-ho d'una altra forma possiblement provocarà que no supereu els jocs de proves privats, quedant-vos així amb la meitat de la nota.

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.
Se suposa que la seqüència d'entrada serà correcta (sense pop ni front sobre cua buida), ni farà coses extranyes com cridar a take de la pròpia cua (així que no cal que tracteu aquest cas).
El programa principal que us oferim ja s'encarrega de llegir aquestes entrades i fer les crides als corresponents mètodes de la classe cua. Només cal que feu els canvis abans esmentats.

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<int> q0 , q1 ;
q0 .push( "a" );
q0 .push( "b" );
q0 .push( "c" );
q0 .push( "d" );
q1 .push( "e" );
q1 .push( "f" );
q1 .push( "g" );
q1 .push( "h" );
cout<< q0 <<endl;
cout<< q1 <<endl;
cout<< q0 .size()<<endl;
cout<< q1 .size()<<endl;
cout<< q0 .front()<<endl;
cout<< q1 .front()<<endl;
q0 .take( q1 , 2 )<<endl;
cout<< q0 <<endl;
cout<< q1 <<endl;
cout<< q0 .size()<<endl;
cout<< q1 .size()<<endl;
cout<< q0 .front()<<endl;
cout<< q1 .front()<<endl;
q1 .take( q0 , 3 )<<endl;
cout<< q0 <<endl;
cout<< q1 <<endl;
cout<< q0 .size()<<endl;
cout<< q1 .size()<<endl;
cout<< q0 .front()<<endl;
cout<< q1 .front()<<endl;
q0 .take( q1 , 1 )<<endl;
cout<< q0 <<endl;
cout<< q1 <<endl;
cout<< q0 .size()<<endl;
cout<< q1 .size()<<endl;
cout<< q0 .front()<<endl;
cout<< q1 .front()<<endl;
q1 .take( q0 , 0 )<<endl;
cout<< q0 <<endl;
cout<< q1 <<endl;
cout<< q0 .size()<<endl;
cout<< q1 .size()<<endl;
cout<< q0 .front()<<endl;
cout<< q1 .front()<<endl;
q0 .take( q1 , 4 )<<endl;
cout<< q0 <<endl;
cout<< q1 <<endl;
cout<< q0 .size()<<endl;
cout<< q1 .size()<<endl;
cout<< q0 .front()<<endl;
q1 .take( q0 , 9 )<<endl;
cout<< q0 <<endl;
cout<< q1 <<endl;
cout<< q0 .size()<<endl;
cout<< q1 .size()<<endl;
cout<< q1 .front()<<endl;
q0 .take( q1 , 1000000000 )<<endl;
cout<< q0 <<endl;
cout<< q1 <<endl;
cout<< q0 .size()<<endl;
cout<< q1 .size()<<endl;
cout<< q0 .front()<<endl;
```

Exemple de sortida 1

```
a b c d
e f g h
4
4
a
e
a b c d e f
g h
6
2
a
g
d e f
g h a b c
3
5
d
g
d e f g
h a b c
4
4
d
h
d e f g
h a b c
4
4
d
h
d e f g h a b c

8
0
d

d e f g h a b c
0
8
d
d e f g h a b c

8
0
d
```

Exemple d'entrada 2

```
Queue<int> q0 , q1 , q2 ;
cout<< q1 .size()<<endl;
q0 .push( "bd" );
q0 .take( q2 , 1 );
cout<< q1 <<endl;
q1 .take( q2 , 0 );
q1 .push( "adb" );
cout<< q1 .front()<<endl;
q1 .push( "ccc" );
q0 .push( "ad" );
q0 .take( q2 , 0 );
cout<< q2 .size()<<endl;
q2 .push( "dcb" );
q1 .take( q2 , 0 );
q2 .pop();
q1 .push( "bb" );
q0 .push( "bcb" );
q2 .take( q0 , 3 );
cout<< q0 <<endl;
q2 .push( "c" );
q2 .take( q0 , 0 );
q2 .push( "daa" );
q1 .push( "cc" );
cout<< q0 .size()<<endl;
q2 .push( "db" );
q2 .push( "a" );
q2 .push( "ac" );
q0 .push( "ada" );
q0 .push( "bd" );
q2 .take( q1 , 5 );
cout<< q1 <<endl;
cout<< q0 <<endl;
q2 .push( "dd" );
q1 .take( q0 , 2 );
q1 .push( "aba" );
cout<< q2 <<endl;
q2 .push( "b" );
q2 .push( "aa" );
q2 .push( "ac" );
cout<< q0 <<endl;
q2 .pop();
q2 .take( q1 , 3 );
q1 .push( "bad" );
q0 .push( "cd" );
q1 .push( "b" );
q2 .push( "d" );
q0 .take( q1 , 4 );
cout<< q1 .size()<<endl;
q2 .push( "aac" );
cout<< q0 .size()<<endl;
cout<< q0 <<endl;
q0 .push( "dcd" );
q2 .push( "ba" );
q2 .push( "ccb" );
q0 .push( "bdd" );
cout<< q2 .front()<<endl;
q2 .pop();
q2 .push( "c" );
cout<< q2 .size()<<endl;
cout<< q0 .size()<<endl;
q0 .push( "acc" );
cout<< q2 .front()<<endl;
cout<< q0 .size()<<endl;
q2 .take( q1 , 0 );
cout<< q0 <<endl;
q2 .push( "dca" );
q0 .push( "bcb" );
q1 .push( "d" );
q2 .push( "db" );
cout<< q0 <<endl;
```

```
cout<< q1 <<endl;
q2 .pop();
q0 .push( "aa" );
q0 .pop();
q0 .pop();
q1 .push( "d" );
cout<< q1 .size()<<endl;
q0 .take( q2 , 14 );
cout<< q0 <<endl;
q0 .push( "b" );
q1 .take( q0 , 11 );
q0 .pop();
cout<< q0 <<endl;
cout<< q1 <<endl;
cout<< q2 <<endl;
```

Exemple de sortida 2

```
0

adb
0

0

ada bd
bd ad bcb c daa db a ac adb ccc bb cc dd

0
3
cd bad b
ad
22
5
bcb
6
cd bad b dcd bdd acc
cd bad b dcd bdd acc bcb
d
2
b dcd bdd acc bcb aa c daa db a ac adb ccc bb cc dd b a
ccc bb cc dd b aa ac ada b
d d b dcd bdd acc bcb aa c daa db a ac
bd aba d aac ba ccb c dca db
```

Exemple d'entrada 3

```
Queue<int> q0 , q1 ;
q0 .push( "a" );
cout<< q0 <<endl;
cout<< q1 <<endl;
cout<< q0 .size()<<endl;
cout<< q1 .size()<<endl;
q0 .take( q1 , 2 )<<endl;
cout<< q0 <<endl;
cout<< q1 <<endl;
cout<< q0 .size()<<endl;
cout<< q1 .size()<<endl;
q1 .take( q0 , 3 )<<endl;
cout<< q0 <<endl;
cout<< q1 <<endl;
cout<< q0 .size()<<endl;
cout<< q1 .size()<<endl;
q0 .take( q1 , 1 )<<endl;
cout<< q0 <<endl;
cout<< q1 <<endl;
cout<< q0 .size()<<endl;
cout<< q1 .size()<<endl;
q1 .take( q0 , 0 )<<endl;
cout<< q0 <<endl;
cout<< q1 <<endl;
cout<< q0 .size()<<endl;
cout<< q1 .size()<<endl;
q0 .take( q1 , 4 )<<endl;
cout<< q0 <<endl;
cout<< q1 <<endl;
cout<< q0 .size()<<endl;
```

```
cout<< q1 .size()<<endl;
q1 .take( q0 , 9 )<<endl;
cout<< q0 <<endl;
cout<< q1 <<endl;
cout<< q0 .size()<<endl;
cout<< q1 .size()<<endl;
q0 .take( q1 , 1000000000 )<<endl;
cout<< q0 <<endl;
cout<< q1 <<endl;
cout<< q0 .size()<<endl;
cout<< q1 .size()<<endl;
```

Exemple de sortida 3

```
a

1
0
a

1
0

a
0
1
a

1
```

```
0
a

1
0
a

1
0

a
0
1
a

1
0
```

Observació

Avaluació sobre 10 punts: (Afegiu comentaris si el vostre codi no és prou clar)

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

Entenem com a solució ràpida una que és correcta 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. Per exemple, una solució que superi tots els jocs de proves però que manegui incorrectament la memòria serà invalidada i tindrà nota 0.
Una solució basada en cridar a mètodes primitius de la pròpia classe possiblement serà lenta i, en cas que no ho sigui, pot tenir una certa penalització en la nota.

Informació del problema

Autoria: PRO2

Generació: 2026-01-27T18:55:46.662Z