Mètode de la classe Stack que deixa una única ocurrència de cada element X64551_ca

Implementa un nou mètode de la classe `Stack` que esborra tots els elements repetits llevat de la primera aparició de cada element (el que està més a dalt de la pila).

Entre els fitxers que s'adjunten en aquest exercici, trobaràs `stack.old.hpp`, a on hi ha una implementació de la classe genèrica `Stack`. En primer lloc, hauràs de fer:

```
cp stack.old.hpp stack.hpp
```

A continuació si obres el fitxer `stack.hpp` al final del mateix trobaràs el mètode que has d'implementar:

```
// Pre: cert.
// Post: Esborra del p.i. els elements de la pila que estan
//   duplicats mantenint la primera aparició de cada element
//   (els que estan més a dalt de la pila).
template <typename T>
void Stack<T>::removeDuplicates() {
   // Aquest és el mètode que has d'implementar
}
```

IMPORTANT: No toquis la resta de la implementació de la classe, excepte si per algun motiu, consideres que necessites afegir algun mètode auxiliar o atribut a la part privada.

També pots trobar entre els fitxers que s'adjunten a l'exercici el fitxer `program.cpp` (programa principal) i el `Makefile` per a compilar i generar l'executable. El programa principal que t'oferim ja s'encarrega de llegir les piles i fer les crides al mètode indicat. **Només cal que implementis el mètode `removeDuplicates`**.

Per a pujar la teva solució, has de crear el fitxer `solution.tar` així:

```
tar cf solution.tar stack.hpp
```

Observació 1

Hauries d'aconseguir implementar el mètode demanat a base d'intercanviar els punters de l'objecte. De fet, una solució a base d'usar `push` i `pop` potser et permetrà passar els jocs de proves, però atès que la solució ha de ser eficient en temps i espai, aquest tipus de solució serà fortament penalitzat.

Observació 2

Recorda que si crees funcions auxiliars, has d'afegir-hi les corresponents **Precondició** (Pre) i **Postcondició** (Post). En els bucles inclou l'**invariant del bucle** (Inv) i la **funció de fita** (FF). En les crides recursives inclou la **hipòtesi d'inducció** (HI) i la **funció de fita** (FF).

Entrada

L'entrada del programa és una seqüència de piles. Per llegir les piles, s'utilitza l'operador `>>` que es troba definit en el fitxer `stack.hpp`.

Sortida

Per cada pila s'escriurà el resultat del mètode `removeDuplicates`. Per escriure les piles, s'ha utilitzat l'operador `<<` que es troba definit en el fitxer `stack.hpp`.

Exemple d'entrada 1

```
25  1 10 2 10 2 3 10 2 4 10 2 5 10 2 6 10 2 7 10 2 8 10 2 9 10 2 10
10  2 4 6 8 10 12 14 16 18 20
18   -1 -2 -3 -4 -5 -6 -7 -8 -9 -1 -2 -3 -4 -5 -6 -7 -8 -9
20   100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100
```

Exemple de sortida 1

```
25 1 10 2 3 4 5 6 7 8 9 10
10 2 4 6 8 10 12 14 16 18 20
18 -1 -2 -3 -4 -5 -6 -7 -8 -9
20 100 100 100 100 100 100 100 100 100 100
```

Exemple d'entrada 2

```
2  -9 -7
1  3
17  1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
17  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2
```

Exemple de sortida 2

```
2 -9 -7
1 3
2 1 0
2 0 2
```

Informació del problema

Autor : Bernardino Casas
Generació : 2025-06-10 10:47:01