Inserir i comptar claus repetides en un BST — X90310_ca

Donada la classe *dicc* que permet gestionar diccionaris usant arbres binaris de cerca (BST) on les claus poden estar **repetides**, cal implementar els mètodes

```
void insereix (const Clau &k);
// Pre: Cert
// Post: Insereix la clau k en el diccionari.

nat quantes(const Clau &k) const;
// Pre: Cert
// Post: Retorna el nombre de claus iguals a k
```

Les claus són del tipus *Clau* que admet una relació d'ordre total, és a dir, tenim una operació de comparació < entre claus.
Cal enviar a jutge.org la següent especificació de la classe *dicc* i la implementació dels mètodes dins del mateix fitxer. La resta de mètodes públics i privats ja estan implementats.

```
#include <iostream>
using namespace std;
typedef unsigned int nat;

template <typename Clau>
class dicc {
  // Diccionari implementat en un BST on les claus poden estar repetides.

  public:
    // Constructora per defecte. Crea un diccionari buit.
    dicc ();

    // Destructora
    ~dicc ();

    // Imprimeix el contingut del diccionari: Nombre d'elements i
    // totes les claus de més petita a més gran separades per un espai
    void print () const;

    void insereix (const Clau &k);
    // Pre: Cert
    // Post: Insereix la clau k en el diccionari.

    nat quantes(const Clau &k) const;
    // Pre: Cert
    // Post: Retorna el nombre de claus iguals a k

  private:
    struct node {
      Clau _k;        // Clau
```

```
    node* _esq;     // fill esquerre
    node* _dret;    // fill dret
  };
  node *_arrel;
  nat _n;           // Nombre d'elements del diccionari

  // Elimina els nodes del subarbre apuntat per p
  static void esborra_nodes(node* p);
  // Imprimeix ordenades les claus del subarbre apuntat per p
  static void print(node* p);

  // Aquí va l'especificació dels mètodes privats addicionals
};

// Aquí va la implementació dels mètodes públics i dels mètodes privats addicionals
```

Degut a que jutge.org només permet l'enviament d'un fitxer amb la solució del problema, en el mateix fitxer hi ha d'haver l'especificació de la classe i la implementació dels mètodes *insereix* i *quantes* (el que normalment estarien separats en els fitxers *.hpp* i *.cpp*).
Per testejar la classe disposes d'un programa principal que processa blocs que contenen un diccionari amb claus enteres seguit de comandes per comptar quantes claus són iguals a una donada.

Entrada

L'entrada conté varis blocs separats per línies amb 10 guions (----------). Cada bloc consisteix en una línia que conté una seqüències d'enters, són els elements que tindrà originalment el diccionari. A continuació segueixen vàries comandes, una per línea, amb el següent format (clau és un enter):

- quantes clau

Sortida

Per a cada línia d'entrada, escriu una línia amb el resultat:

- Si la línia és un diccionari, mostra el nombre de claus del diccionari i totes les claus de més petita a més gran separades per un espai.
- Si la línia és una comanda, mostra la comanda, el separador ": " i el resultat.
- Si la línia és el separador de blocs format per 10 guions, mostra els mateixos 10 guions.

Observació

Només cal enviar la classe requerida i la implementació dels mètodes *insereix* i *quantes*. Pots ampliar la classe amb mètodes privats. Segueix estrictament la definició de la classe de l'enunciat.
Els mètodes *insereix* i *quantes* almenys han de tenir cost logarítmic (en el cas mig) per superar els jocs de prova privats.

Exemple d'entrada 1

```
quantes -6
quantes 0
```

Exemple de sortida 1

```
0:
quantes -6: 0
quantes 0: 0
```

Exemple d'entrada 2

```
-6
quantes -6
quantes 0
```

Exemple de sortida 2

```
1: -6
quantes -6: 1
quantes 0: 0
```

Exemple d'entrada 3

```
0 -6
quantes 0
quantes -6
quantes 6
```

Exemple de sortida 3

```
2: -6 0
quantes 0: 1
quantes -6: 1
quantes 6: 0
```

Exemple d'entrada 4

```
-6 -6
quantes 0
quantes -6
quantes 6
```

Exemple de sortida 4

```
2: -6 -6
quantes 0: 0
quantes -6: 2
quantes 6: 0
```

Exemple d'entrada 5

```
5 -3 8 2 -1 7 -7 -6
quantes 8
quantes 5
quantes -7
quantes 0
```

Exemple de sortida 5

```
8: -7 -6 -3 -1 2 5 7 8
quantes 8: 1
quantes 5: 1
quantes -7: 1
quantes 0: 0
```

Exemple d'entrada 6

```
5 -3 8 5 -3 7 5 -8
quantes 8
quantes 5
quantes -3
quantes 0
```

Exemple de sortida 6

```
8: -8 -3 -3 5 5 5 7 8
quantes 8: 1
quantes 5: 3
quantes -3: 2
quantes 0: 0
```

Exemple d'entrada 7

```
5 -5 -3 9 -5 5 -2 1 -3 9 -5 0 4 -5 5 -3 4
quantes -5
quantes -3
quantes -2
quantes 0
quantes 1
quantes 2
quantes 4
quantes 5
quantes 9
----------
4 5 5 5 5 5 5 5 5 5 5 5 5 5
quantes 0
quantes 4
quantes 5
```

Exemple de sortida 7

```
17: -5 -5 -5 -5 -3 -3 -3 -2 0 1 4 4 5 5 5 9 9
quantes -5: 4
quantes -3: 3
quantes -2: 1
quantes 0: 1
quantes 1: 1
quantes 2: 0
quantes 4: 2
quantes 5: 3
quantes 9: 2
----------
14: 4 5 5 5 5 5 5 5 5 5 5 5 5 5
quantes 0: 0
quantes 4: 1
quantes 5: 13
```

Informació del problema

Autoria: Jordi Esteve

Generació: 2026-01-25T17:09:16.803Z