Elements entre el i-èssim i j-èssim d'un BST — X21271_ca

Donada la classe *dicc* que permet gestionar diccionaris on només hi guardem claus úniques usant arbres binaris de cerca (BST), cal implementar el mètode

```
vector<Clau> ijessim(nat i, nat j) const;
// Pre: El diccionari no està buit. 1 <= i <= j <= quants()
// Post: Retorna les claus entre la posició i-èssima i j-èssima
// (comptant-les en ordre ascendent).
```

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 del mètode dins del mateix fitxer. La resta de mètodes públics i privats ja estan implementats.

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

template <typename Clau>
class dicc {

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

    // Destructora
    ~dicc();

    // Insereix la clau k en el diccionari. Si ja hi era, no fa res.
    void insereix(const Clau &k);

    // Retorna quants elements (claus) té el diccionari.
    nat quants() const;

    // Pre: El diccionari no està buit. 1 <= i <= j <= quants()
    // Post: Retorna les claus entre la posició i-èssima i j-èssima
    // (comptant-les en ordre ascendent).
    vector<Clau> ijessim(nat i, nat j) const;

  private:
    struct node {
      Clau _k;          // Clau
      node* _esq;       // fill esquerre
      node* _dret;      // fill dret
      nat _n;           // Nombre de nodes del subarbre
      node(const Clau &k, node* esq = NULL, node* dret = NULL);
```

```
    };
    node *_arrel;

    static void esborra_nodes(node* m);
    static node* insereix_bst (node *n, const Clau &k, bool &ins);

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


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

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 del mètode *ijessim* (el que normalment estarien separats en els fitxers *.hpp* i *.cpp*).
Per testejar la classe disposeu d'un programa principal que processa blocs que contenen un diccionari amb claus enteres seguit de comandes per calcular les claus entre la posició i-èssima i j-èssima.

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 (i j són naturals majors que 0):

- ijessim i j

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 un cop inserit tots els seus elements.
- 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 del mètode *ijessim*. Podeu ampliar la classe amb mètodes privats. Seguiu estrictament la definició de la classe de l'enunciat.
El mètode *ijessim* almenys ha de tenir cost logarítmic respecte el nombre de claus del diccionari (en el cas mig) per superar els jocs de prova privats. I cost lineal respecte el nombre de claus a retornar $(j - i + 1)$.

Exemple d'entrada 1

```
5
ijessim 1 1
----------
5 2
ijessim 1 1
ijessim 2 2
ijessim 1 2
```

Exemple de sortida 1

```
1
ijessim 1 1: 5
----------
```

Exemple d'entrada 2

```
5 -3 8 2 -1 7 -7 -6
ijessim 1 1
ijessim 2 2
ijessim 3 3
ijessim 4 4
ijessim 5 5
ijessim 6 6
ijessim 7 7
ijessim 8 8
ijessim 1 2
ijessim 2 3
ijessim 3 4
ijessim 4 5
ijessim 5 6
ijessim 6 7
ijessim 7 8
ijessim 1 3
ijessim 2 4
ijessim 3 5
ijessim 4 6
ijessim 5 7
ijessim 6 8
ijessim 1 4
ijessim 2 5
ijessim 3 6
ijessim 4 7
ijessim 5 8
ijessim 1 5
ijessim 2 6
ijessim 3 7
ijessim 4 8
ijessim 1 6
ijessim 2 7
ijessim 3 8
ijessim 1 7
ijessim 2 8
ijessim 1 8
```

```
2
ijessim 1 1: 2
ijessim 2 2: 5
ijessim 1 2: 2 5
```

Exemple de sortida 2

```
8
ijessim 1 1: -7
ijessim 2 2: -6
ijessim 3 3: -3
ijessim 4 4: -1
ijessim 5 5: 2
ijessim 6 6: 5
ijessim 7 7: 7
ijessim 8 8: 8
ijessim 1 2: -7 -6
ijessim 2 3: -6 -3
ijessim 3 4: -3 -1
ijessim 4 5: -1 2
ijessim 5 6: 2 5
ijessim 6 7: 5 7
ijessim 7 8: 7 8
ijessim 1 3: -7 -6 -3
ijessim 2 4: -6 -3 -1
ijessim 3 5: -3 -1 2
ijessim 4 6: -1 2 5
ijessim 5 7: 2 5 7
ijessim 6 8: 5 7 8
ijessim 1 4: -7 -6 -3 -1
ijessim 2 5: -6 -3 -1 2
ijessim 3 6: -3 -1 2 5
ijessim 4 7: -1 2 5 7
ijessim 5 8: 2 5 7 8
ijessim 1 5: -7 -6 -3 -1 2
ijessim 2 6: -6 -3 -1 2 5
ijessim 3 7: -3 -1 2 5 7
ijessim 4 8: -1 2 5 7 8
ijessim 1 6: -7 -6 -3 -1 2 5
ijessim 2 7: -6 -3 -1 2 5 7
ijessim 3 8: -3 -1 2 5 7 8
ijessim 1 7: -7 -6 -3 -1 2 5 7
ijessim 2 8: -6 -3 -1 2 5 7 8
ijessim 1 8: -7 -6 -3 -1 2 5 7 8
```

Informació del problema

Autoria: Jordi Esteve

Generació: 2026-01-25T14:19:47.441Z