Fusió ordenada des d'una altra llista — X94603_ca

Heu d'implementar una funció que rep dues llistes d'enters l_1, l_2 com a paràmetre per referència. Ambdues llistes se suposen ordenades creixentment. La funció haurà d'insertar tots els elements de l_2 dins de l_1, de manera que l_1 continui quedant ordenat creixentment. En altres paraules, el valor final de l_1 haurà de ser el resultat de fusionar ordenadament els valors inicials de l_1 i l_2.

Important: Heu de garantir que els elements que la llista l_1 contenia inicialment queden inalterats. En particular, la funció no els pot eliminar i tornar a afegir després.
Aquesta és la capcelera:

```
// Pre: l1 i l2 estan ordenades creixentment. Siguin L1 i L2 els seus valors in
// Post: l1 conté la fusió ordenada de L1 i L2.
//       A més a més, els elements inicials de la llista han persistit i
//       no han canviat de valor.
void mergeIntoFirstList(list<int> &l1, list<int> l2);
```

Aquí tenim un exemple de comportament de la funció:

```
mergeIntoFirstList(L1 = [1,3,3,5], [2,3,4,7,8]) => L1 = [1,2,3,3,3,4,5,7,8]
```

Fixeu-vos que l'enunciat d'aquest exercici ja ofereix uns fitxers que haureu d'utilitzar per a compilar: `main.cc`, `mergeIntoFirstList.hh`. Us falta crear el fitxer `mergeIntoFirstList.cc` amb els corresponents `includes` i implementar-hi la funció anterior. Només cal que pugeu `mergeIntoFirstList.cc` al jutge.

Entrada

L'entrada té un nombre arbitrari de casos. Cada cas consisteix en dues llistes d'enters, cadascuna en una línia diferent. Fixeu-vos en que el programa que us oferim ja s'encarrega de llegir aquestes entrades. Només cal que implementeu la funció abans esmentada.

Sortida

Per a cada cas, la sortida conté el corresponent resultat de la funció. Fixeu-vos en que el programa que us oferim ja s'encarrega d'escriure aquesta sortida. Només cal que implementeu la funció abans esmentada.

Exemple d'entrada

```
3 5 5 6 6 7
0 1 2 7 9
0 2 3 6 6 6
0 2 2 3 5 7 7 8 9 9
2 3 7 8 9
1 1 2 3 4 7 8 9 9
0 0 1 3 5 6 6
0 1 2 4 5 5 6 7
3 4 5 5 6 7 9
4 4 4 5 7
```

```
0 3 4 6 7 8 8 8
0 2 2 4 6 6 6 8 9 9
0 4 5 8 9
0 1 1 2 2 2 6 6 7 7
0 4 5 9 9 9
1 1 5 7 7 7 7 9
3 3 4 5 6 6 7 8 9
0 2 3 5 8 8 9 9
1 1 3 5 5 6 6 8 9
0 0 1 3 4 4 4 4 8 8
```

Exemple de sortida

```
0 1 2 3 5 5 6 6 7 7 9
0 0 2 2 2 3 3 5 6 6 6 7 7 8 9 9
1 1 2 2 3 3 4 7 7 8 8 9 9 9
0 0 0 1 1 2 3 4 5 5 5 6 6 6 7
```

```
3 4 4 4 4 5 5 5 6 7 7 9
0 0 2 2 3 4 4 6 6 6 6 7 8 8 8 8 9 9
0 0 1 1 2 2 2 4 5 6 6 7 7 8 9
0 1 1 4 5 5 7 7 7 7 9 9 9 9
0 2 3 3 3 4 5 5 6 6 7 8 8 8 9 9 9
0 0 1 1 1 3 3 4 4 4 4 5 5 6 6 8 8 8 9
```

Observació

La vostra funció i subfuncions que creeu han de treballar només amb llistes. Avaluació sobre 10 punts:

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

Entenem com a solució ràpida una que és correcta, de cost lineal 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.
Una solució que altera, o elimina els elements originals de la primera llista i els torna a afegir més tard rebrà un 0.

Informació del problema

Autor : PRO2
Generació : 2024-03-29 19:48:47