Revessar seqüència d'items — X86574_ca

Preliminars
En aquest exercici treballarem sobre la següent estructura de dades, que ens serveix per a mantenir una seqüència de valors dins de items encadenats mitjançant punters.

```
struct Item {
int value;
Item* next;
};
```

Exercici
Implementeu una funció **RECURSIVA** que, donat un `Item*` que apunta a una seqüència d'items encadenats, retorna un altre `Item*` que apunta a una nova seqüència que representa el revessat de la original. En altres paraules, la nova seqüència d'items no comparteix memòria amb la original, però la seva corresponent seqüència de valors és el revessat de la seqüència de valors original.

```
// Pre:  pitem apunta al primer element d'una seqüència correcta d'items encade
//       L'últim element de la seqüència apunta a NULL. El propi pitem podria s
//       cas en el qual no hi hauria elements a la seqüència.
// Post: Retorna un Item* que representa una seqüència d'items nous tals que la
//       corresponent seqüència de valors és el revessat de seqüència de valors
//       La seqüència original no ha canviat.
Item* reverse(Item* pitem);
```

Aquí tenim un exemple de paràmetres entrada i sortida de la funció:

```
reverse([3]->[2]->[5]->[1]->[8]->NULL) = [8]->[1]->[5]->[2]->[3]->NULL
```

Fixeu-vos que l'enunciat d'aquest exercici ja ofereix uns fitxers que haureu d'utilitzar per a compilar: `Makefile, program.cpp, reverse.hpp`. Us falta crear el fitxer `reverse.cpp` amb els corresponents `includes` i implementar-hi la funció anterior. Quan pugeu la vostra solució al jutge, només cal que pugeu un tar construït així:

```
tar cf solution.tar reverse.cpp
```

Possiblement necessitareu crear una funció auxiliar amb més paràmetres per tal de resoldre aquest exercici eficientment.

Entrada

L'entrada té un nombre arbitrari de casos. Cada cas consisteix en una línia amb una llista de valors enters. 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 dues línies, la primera amb la mateixa llista de valors original, i la segona amb la llista revessada resultant. Fixeu-vos en que el programa que us oferim ja s'encarrega d'escriure aquestes dades. Només cal que implementeu la funció abans esmentada.

Exemple d'entrada

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

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

Exemple de sortida

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


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

Informació del problema

Autor : PRO1
Generació : 2022-10-25 20:56:40