Algoritmos y Estructuras de Datos – PARCIAL 3 –11/2/2025

Problema: “Procesamiento de Lista Enlazada”
Completar el programa “esqueleto” que encontrará luego del enunciado, con el código de la función **Parcial3()** y todo el código que considere necesario, para que cumpla con la siguiente consigna:

La función Parcial3(...) debe recibir una lista enlazada L1 (que inicialmente contiene una cantidad de nodos >= 1) con los datos de los participantes inscriptos a un torneo de kick boxing. Para cada uno de ellos se dispone de la siguiente información:

- Nombre (string de hasta 25 caracteres)
- Altura (valor entero en cm)
- Peso (valor entero en 50..200 kg)

La lista L1 se carga al comienzo del programa por teclado, y los datos se ingresan ordenados por el nombre de los participantes (aunque algunos, cuando se inscribieron, completaron el formulario más de una vez, y por lo tanto, luego de la carga inicial, pueden quedar en L1, uno, dos ó mas nodos para el mismo participante. (Ver ejemplos!!!)

Su tarea es codificar la función *Parcial3()*, con los parámetros que considere necesarios, para:

- La lista deberá procesarse, para que sólo quede un nodo en la misma, para cada uno de los participantes inscriptos (no hay dos participantes con el mismo nombre).
- Cuando un participante tiene más de un nodo con datos, sólo debe quedar en la lista un nodo para dicho participante (y si las alturas y/o los pesos registrados en sus nodos son diferentes, se debe guardar en cualquier caso, el valor mayor de cada uno de ellos).
- La función debe retornar los dos valores que se imprimen, luego de la llamada a la misma en la función main(), y que tienen el siguiente significado:
  - R1: cantidad de participantes inscriptos (luego del proceso de la lista)
  - R2: suma total de los pesos de todos los inscriptos

Entrada
Los valores para cargar la lista: en cada línea los datos de un participante (nombre, altura y peso, separados por un espacio) hasta EOF. **Aclaración**: la lista inicial nunca será vacía.

Salida
Un renglón con R1 y R2.

Ejemplos

Entrada	**Salida**
Abel 160 60 Ruperto 199 105	// la lista quedará // (Abel 160 60) -> (Ruperto 199 105) 2 165
Abel 160 60 Abel 160 65 Ruperto 200 105 Ruperto 199 105	// la lista quedará // (Abel 160 65) -> (Ruperto 200 105) 2 170
Abel 160 71 Abel 160 75 Abel 160 74 Ruperto 200 105 Zoilo 199 125	// la lista quedará // (Abel 160 75) -> (Ruperto 200 105) -> (Zoilo 199 125) 3 305

Ejercicio en OmegaUp (30 pts)
Tiempo de Resolución: 60 minutos. - Puntaje Mínimo Requerido: 15/30 puntos.

Observación: º El código fuente enviado se corregirá para asegurar el borrado de los nodos duplicados de la lista original.

Esqueleto

```
#include <stdlib.h>
#include <iostream>
using namespace std;
struct Nodo {
      string nombre;
      int altura, peso;
      struct Nodo *sig;
};

typedef Nodo * NodoPtr;
void insertar( NodoPtr & sPtr, string NOMBRE, int ALTURA, int
PESO);
void mostrarLista( NodoPtr actual );
void Parcial3( ... );

int main() {
      NodoPtr L1 = NULL;
      string dato1;
      int dato2, dato3;
      int cantidad=0, sumapesos=0;

      while ( cin >> dato1 >> dato2 >> dato3)
            insertar( L1, dato1, dato2, dato3 );

      Parcial3(...);

      cout << cantidad << " " << sumapesos << endl;

      return 0;
}

void insertar( NodoPtr & sPtr, string NOMBRE, int ALTURA, int
PESO){
NodoPtr nuevo;

if (sPtr == NULL) {
      nuevo = new Nodo;
      if ( nuevo != NULL ) {
      nuevo->nombre = NOMBRE;
      nuevo->altura = ALTURA;
      nuevo->peso = PESO;
      nuevo->sig = NULL;
      sPtr = nuevo;}
      else cout << "No hay espacio";
}
      else insertar (sPtr->sig, NOMBRE, ALTURA, PESO);
```

```
}

void Parcial3( ... ){

}

void mostrarLista( NodoPtr P ) {
while ( P != NULL ) {
      cout <<P->nombre <<" " << P->altura <<" " <<P->peso <<" -> ";
      P= P->sig;
      }
cout << "NULL" << endl;
}
```

Resolver el siguiente problema en OmegaUp:

https://omegaup.com/arena/problem/Procesamiento-Lista-Enlazada/