4. Considere a existência de duas listas simplesmente encadeadas, ORDENADAS crescentemente por **codigo**, em que cada nó encadeado é representado pelo tipo NoLista abaixo.

```
typedef struct noLista NoLista;
struct noLista
{   int codigo;
    char nome[51];
    NoLista *prox;
};
```

Escreva a função **MergeListas** que recebe duas listas simplesmente encadeadas (ou seja, os endereços dos primeiros nós das duas listas) e ordenadas crescentemente por "codigo" e retorna o endereço do primeiro nó da lista resultante da união das duas. A lista retornada também estará ordenada crescentemente por código e não pode conter repetição de código. Nenhum nó é criado. NÃO USE FUNÇÕES AUXILIARES (exceto as disponibilizadas pelas bibliotecas padrão de C) e LEVE EM CONSIDERAÇÃO A ORDENAÇÃO DAS LISTAS para tornar sua função mais eficiente.

Por exemplo, se os elementos da lista 1 fossem:

[111,”leo”], [333, “rui”], [446, “cris”], [558, “lara”], [666,”cadu”], [777,”rosa”], [999,”bia”]

E os elementos da lista 2 fossem:

[111,”leo”], [333, “rui”], [440, “edu”], [554, “vera”], [777,”rosa”], [888,”lia”]

A lista resultante teria os seguintes valores:

[111,”leo”], [333, “rui”], [440, “edu”], [446, “cris”], [554, “vera”], [558, “lara”], [666,”cadu”], [777,”rosa”] , [888,”lia”], [999,”bia”]

e seria retornado o endereço do nó com [111,”leo”].

Os nós não usados na lista resultante devem ser liberados. Considere também que não há repetições de "codigo" numa mesma lista e que um determinado "codigo" corresponderá sempre a um mesmo nome.