Os trens de Dennis

Uma das grandes novidades de São Mateus (se você não está na cidade, talvez não esteja sabendo) é a nova linha ferrea, controlada pela Sama-Trens que liga a cidade à capital do estado (Vitória). Como a demanda de viagens cresceu muito, a empresa teve que comprar mais trens para conseguir atender os passageiros. Com a aquisição de novos trêns, a empresa contratou o jovem Dennis Ritchie para organizar os vagões que serão utilizados em cada viagem. Apesar de ser trabalhosa, sua tarefa é relativamente simples, juntar os vagões de dois trens para formar um único trem. Por exemplo, se o primeiro trem possui 5 vagões e o segundo 3, o novo trem deve possuir 8 vagões.

Cada vagão possui um identificador (um número inteiro) e os vagões de um trem estão sempre em ordem não-decrescente (um requistio da empresa). Assim, Dennis deve formar o novo trem respeitando esse requisito da Sama-Trens. Mesmo Dennis sendo um grande conhecedor de programação, ele está muito cansado e te pediu ajuda para construir um programa que realiza a junção dos trens. Para a brincadeira ficar mais emocionante, ele quer que seja usado listas encadeadas para resolver o problema.

Tarefa

Sua tarefa, nesse problema, é implementar uma função que recebe duas listas ordenadas e retone uma nova lista contendo a junção das listas passadas como argumento. A nova lista também deve estar ordenada (de forma não-decrescente). Para padronizar, cada nó da lista é definido por:

```c
typedef struct no {
    int id; //Idenficador do vagão
    struct no *proximo; //Próximo vagão
} No;
```

E a lista é dada por:

```c
typedef struct lista {
    No *inicio;
} Lista;
```

Você deve implementar/completar a função:

```c
Lista *juntaLista(Lista *l1, Lista *l2) {
    Lista *resultado = criaLista();
    //
    return resultado;
}
```

A função para criar uma lista é dada abaixo:

```c
Lista *criaLista() {
    Lista *lista = malloc(sizeof(Lista));
    lista->inicio = NULL;
    return lista;
}
```

Obs.: Para esse problema, você deve se preocupar apenas com a implementação da função `juntaLista`. Baixe os aquivos (`Lista.h` e `mEP3.c`), disponiblizados no AVA, para facilitar.

Regras

- Você deve implementar/modificar apenas a função `juntaLista`, não sendo permitido a criação de outras funções;
- Assim como Dennis não pode "criar" um novo vagão, você também não pode criar/alocar nenhum novo nó na sua função. Você deve apenas manipular os ponteiros dos nós de `l1` e `l2` para que estejam em `resultado`;
- As listas `l1` e `l2` não precisam estar intactas após a execução da função.

Descrição da entrada

A entrada é composta por quatro linhas. A primeira indica o número de vagões do primeiro trem (t_1). A segunda linha, contém t_1 valores inteiros, separados por um espaço, representando a identificação dos vagões (os valores estão em ordem não-decrescente). A terceira linha, contém o número de vagões do segundo trem (t_2). Por fim, a última linha contém a identificação (valores inteiros em ordem não-decrescente) de cada vagão do segundo trem. Observe o

exemplo abaixo:

```
3
1 3 5
5
2 4 6 8 10
```

Descrição da saída

A saída do programa deverá ser a lista resultante da união da duas listas passadas como argumento para a função `juntaLista`. Para o exemplo anterior, a saída deverá ser da seguinte forma:

```
1 -> 2 -> 3 -> 4 -> 5 -> 6 -> 8 -> 10 -> NULL
```

Submeter Solução

Nome arquivo: mEP3.c

Tempo limite: 0.5 seg. (por caso de teste)

Linguagem: C

Flags de Compilação:

```
-lm -O0 -std=c11 -Wall -Werror -Wextra -Wno-sign-compare -Wno-unused-parameter -Wno-unused-variable -Wshadow
```

Testes abertos: 3

Selecionar arquivo | Browse | **Testar** | **Submeter**

Por que você precisa imprimir a saída **exatamente** como está sendo pedido no problema?