Listas Encadeadas: Inserção

Limite de tempo: 1s
Limite de memória: 256MB

Autor: Daniel Saad Nogueira Nunes

Implemente a inserção em posições arbitrárias de listas encadeadas através do seguinte procedimento:

```
void list_insert(list_t* l, int data, size_t i);
```

No caso o parâmetro `i` corresponde ao índice da inserção do novo elemento.

- Se $i = 0$, equivale à uma inserção na cabeça.
- Se $i = n$, em que n é o tamanho da lista, equivale à uma inserção na cauda.
- Se $0 < i < n$, então equivale à inserção de um novo elemento na posição i.

Por exemplo, se a lista é `1->2->4->5` e o elemento 3 vier a ser inserido na posição 2, então a lista resultante deverá ser `1->2->3->4->5`.

Leve em consideração a seguinte definição de lista encadeada:

```
typedef struct list_node_t {
    int data;                 /*Dado da lista*/
    struct list_node_t *next; /*ponteiro para o próximo elemento*/
} list_node_t;

typedef struct list_t {
    list_node_t *head; /*Cabeça da Lista*/
    list_node_t *tail; /*Cauda da Lista*/
    size_t size;       /*tamanho da lista*/
} list_t;
```

Assuma que as seguintes funções estão disponíveis:

```
size_t list_size(list_t *l); // retorna o tamanho da lista
bool list_empty(list_t *l); // retorna verdadeiro sse a lista está vazia
```

Observação: não se esqueça de tratar os casos em que a lista é vazia e atualizar o tamanho da lista resultante.

Notas

Só é necessário implementar a função pedida no enunciado. Não é necessário realizar a leitura dos dados ou escrever a função `main`. Funções auxiliares, que tenham a ver apenas com a função solicitada, podem ser implementadas, caso deseje.

Se estiver programando em `C` ou `C++` é necessário incluir o cabeçalho "`grader.h`" na sua solução:

```
#include "grader.h"
```