Listas encadeadas - impressão

Considere uma lista encadeada com nó cabeça `le` definida por células

```
typedef struct celula {
   int dado;
   struct celula *prox;
} celula;
```

Sua tarefa nesse exercício é implementar a operação de **impressão** da lista encadeada encabeçada por `le`. Para tanto, você deve submeter um arquivo contendo apenas:

1. Os `#include` necessários para execução das instruções utilizadas no seu código.
2. A definição da `struct celula`.
3. Duas funções (uma iterativa e outra recursiva) que imprimem a lista encadeada. Os protótipos devem ser

```
void imprime (celula *le);
void imprime_rec (celula *le);
```

Exemplos

Se a lista estiver vazia, sua função deve imprimir

```
NULL
```

Se não estiver, os elementos devem ser impressos antes do `NULL` e separados por `->`, da seguinte forma: suponha uma lista com os elementos 1, 2 e 3:

```
1 -> 2 -> 3 -> NULL
```

Atenção: Não deve haver espaço depois do `NULL`.

Author: John L. Gardenghi

Listas encadeadas - inserção

Considere uma lista encadeada com nó cabeça `le` definida por células

```
typedef struct celula {
   int dado;
   struct celula *prox;
} celula;
```

Sua tarefa nesse exercício é implementar a operação de **inserção** na lista encadeada encabeçada por `le`. Para tanto, você deve submeter um arquivo contendo apenas:

1. Os `#include` necessários para execução das instruções utilizadas no seu código.
2. A definição da `struct celula`.
3. Uma função que insere um elemento x no início da lista encadeada, cujo protótipo deve ser:

```
void insere_inicio (celula *le, int x);
```

4. Uma função que insere um elemento x imediatamente antes da primeira ocorrência de um elemento y na lista encadeada. Se y não estiver na lista encadeada, x deve ser inserido ao final. O protótipo dessa função deve ser

```
void insere_antes (celula *le, int x, int y);
```

Author: John L. Gardenghi