Exercícios

1. Implemente uma função que tenha como valor de retorno o comprimento de uma lista encadeada, isto é, que calcule o número de nós de uma lista. Essa função deve obedecer ao protótipo:

```
int comprimento (Lista* l);
```

2. Considerando listas encadeadas de valores inteiros, implemente uma função que retorne o número de nós da lista que possuem o campo `info` com valores maiores do que `x`. Essa função deve obedecer ao protótipo:

```
int maiores (Lista* l, int x);
```

3. Implemente uma função que retorne o último valor de uma lista encadeada de inteiros. Essa função deve obedecer ao protótipo:

```
int ultimo (Lista* l);
```

4. Implemente uma função que receba duas listas encadeadas de valores reais e transfira para o final da primeira lista os elementos da segunda. No final, a primeira lista representará a concatenação das duas listas e a segunda lista estará vazia. Essa função deve obedecer ao protótipo:

```
void concatena (Lista* l1, Lista* l2);
```

5. Considerando listas de valores inteiros, implemente uma função que receba como parâmetros uma lista encadeada e um valor inteiro `x`, e retire da lista todas as ocorrências de `x`. Essa função deve obedecer ao protótipo:

```
void retira_n (Lista* l, int x);
```

6. Considerando listas de valores inteiros, implemente uma função que receba como parâmetro uma lista encadeada e um valor inteiro `x` e divida a lista em duas, de tal forma que a segunda lista, criada dentro da função, comece no primeiro nó logo após a primeira ocorrência de `x` na lista original. A função deve ter como valor de retorno a lista criada, mesmo que ela seja vazia. A Figura 14.10 ilustra este procedimento.

   Essa função deve obedecer ao protótipo:

```
Lista* separa (Lista* l, int x);
```

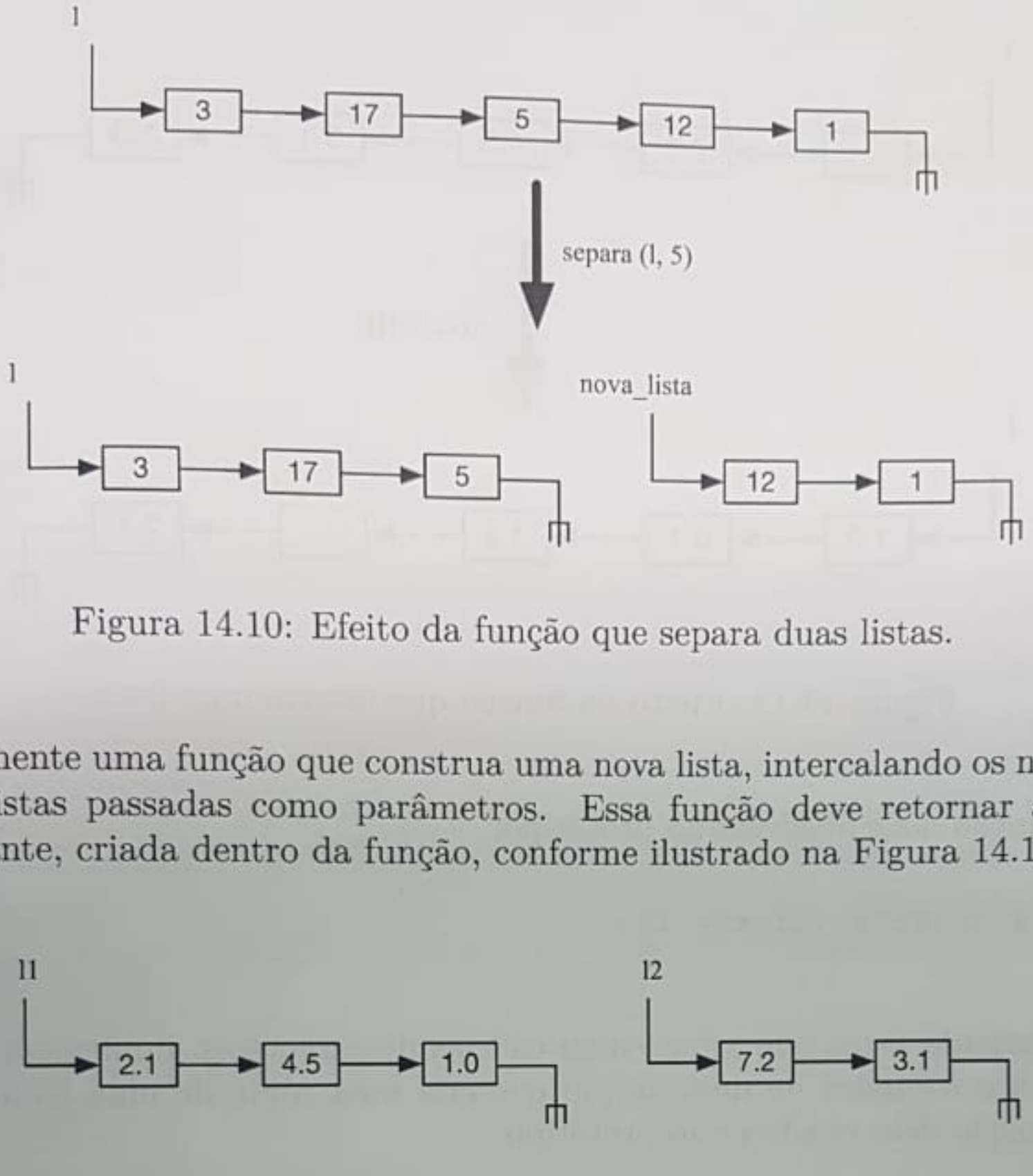

Figura 14.10: Efeito da função que separa duas listas.

7. Implemente uma função que construa uma nova lista, intercalando os nós de outras duas listas passadas como parâmetros. Essa função deve retornar a nova lista resultante, criada dentro da função, conforme ilustrado na Figura 14.11.

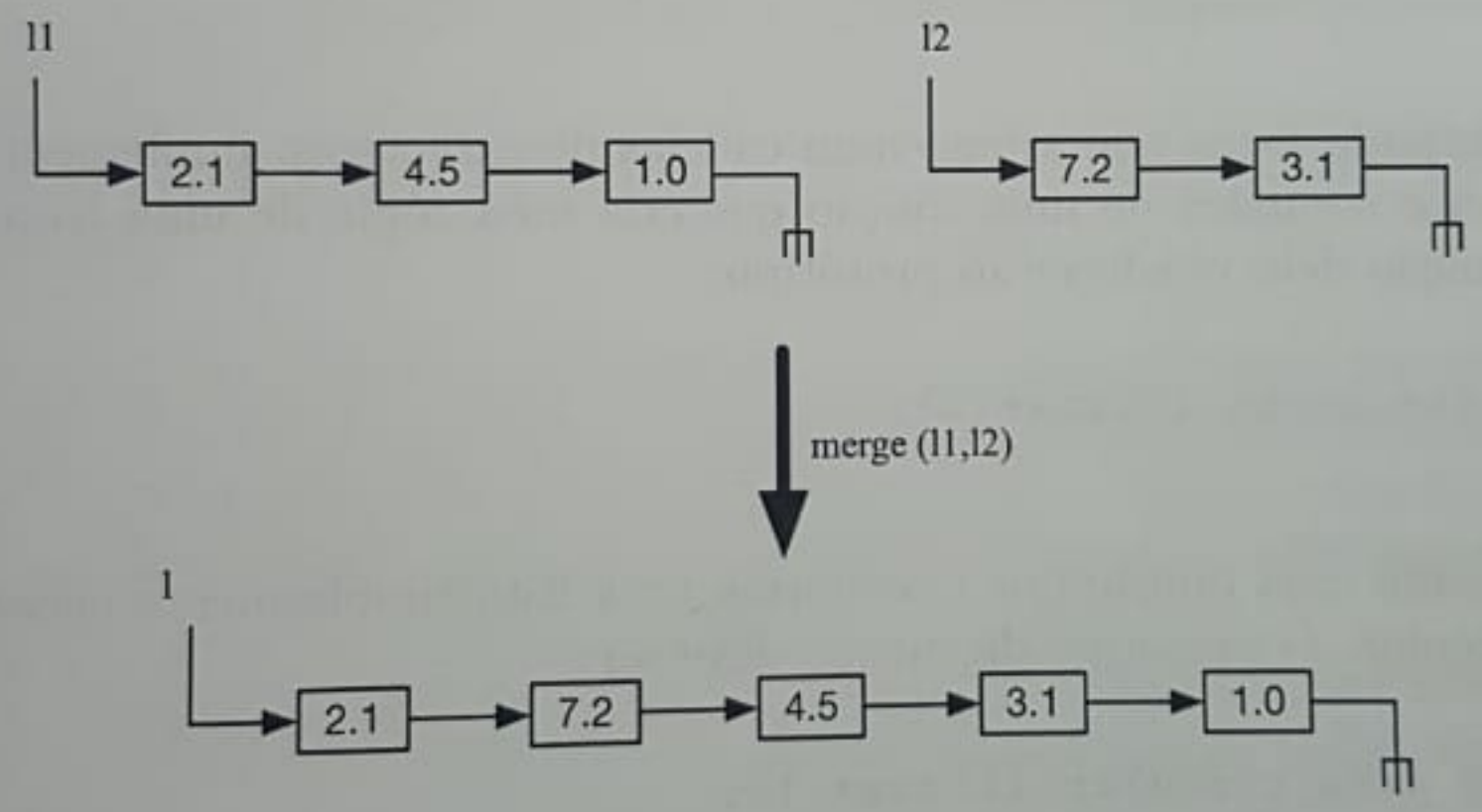

Figura 14.11: Efeito da função que combina duas listas.

Ao final da função, as listas originais devem ficar vazias e ser liberadas. Se uma lista tiver mais elementos que a outra, os elementos excedentes são transferidos na mesma ordem para a nova lista. Essa função deve obedecer ao protótipo:

```
Lista* merge (Lista* l1, Lista* l2);
```

8. Implemente uma função que receba como parâmetro uma lista encadeada e inverta o encadeamento de seus nós. Após a execução dessa função, cada nó da lista vai estar apontando para o nó que originalmente era seu antecessor, e o último nó da lista passará a ser o primeiro nó da lista invertida, conforme ilustrado na Figura 14.12.

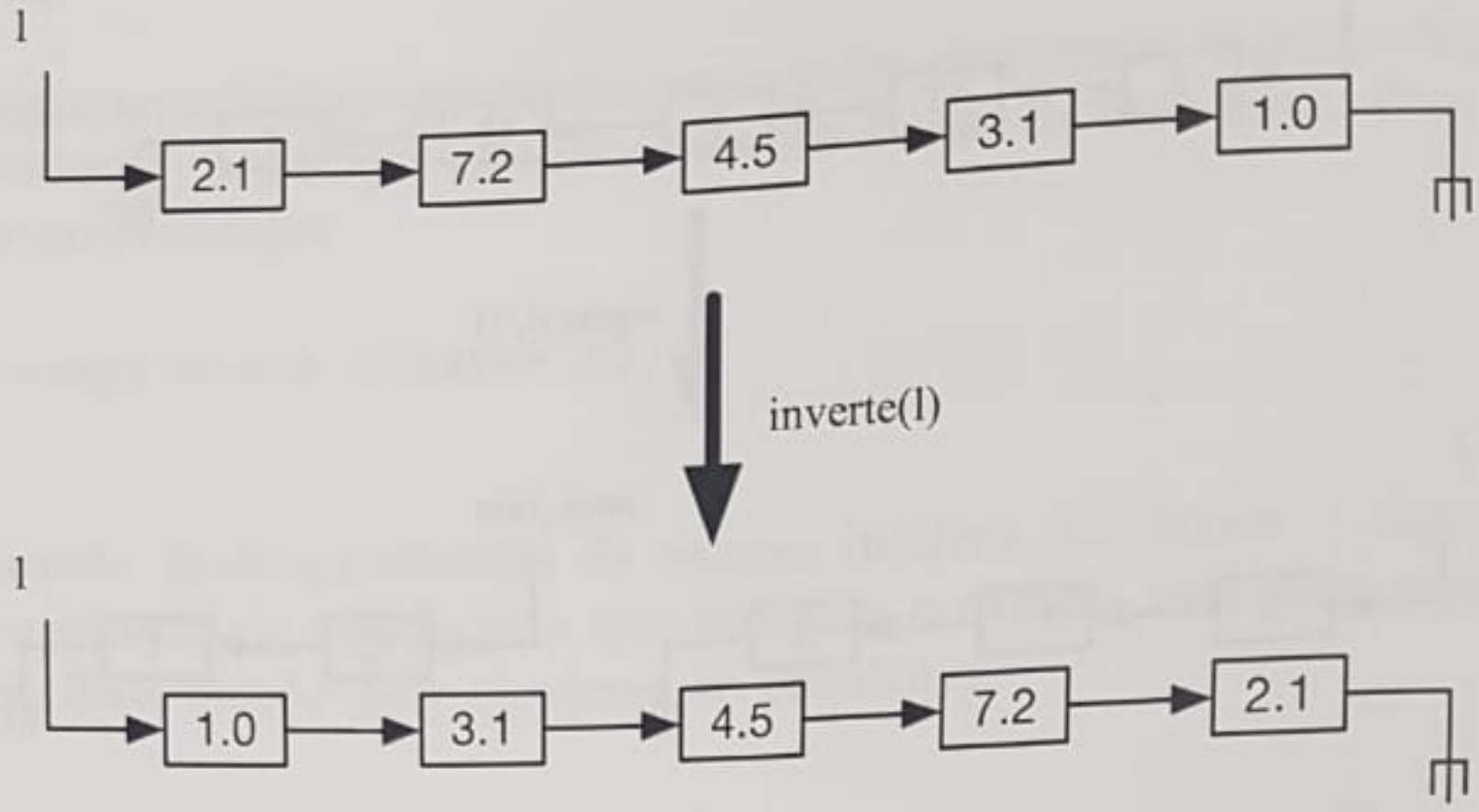

Figura 14.12: Efeito da função que inverte uma lista.

Essa função deve obedecer ao protótipo:

```
void inverte (Lista* l);
```

9. Considerando listas que armazenam cadeias de caracteres, implemente as versões iterativa e recursiva de uma função que cria uma cópia de uma lista encadeada. Essa função deve obedecer ao protótipo:

```
Lista* copia (Lista* l);
```

10. Implemente uma função que transforma uma lista simplesmente encadeada numa lista circular. O protótipo da função deve ser:

```
void para_circular (Lista* l);
```

11. Implemente as funções para retirar elementos do início e do final de uma lista duplamente encadeada. Os protótipos das funções devem ser:

```
void retira_inicio (Lista2* l);
void retira_final (Lista2* l);
```

12. Implemente funções para inserir e retirar um elemento de uma lista circular duplamente encadeada.