

Lista 12 - Exercícios de Lista Duplamente Encadeada

1. Implemente funções para as seguintes operações em uma lista duplamente encadeada:
 - a. Inserir no final da lista
 - b. Remover o primeiro elemento
 - c. Remover o último elemento. **É proibido percorrer a lista. Para encontrar o penúltimo nó, acesse o ponteiro “anterior”.**

Observação: faça uso do descritor da lista para remover ou inserir um novo nó, ou seja, não é necessário percorrer a lista.

2. Implemente funções para as seguintes operações em uma lista duplamente encadeada:

```
//remove da lista todas as ocorrências do parâmetro valor
```

```
void removeTodos(Lista *L, int valor);
```

```
//Retorna uma nova lista, ordenada, com os K maiores elementos de L
```

```
//Se K for maior que o tamanho de L, a lista retornada deve conter todos os
```

```
//elementos de L de forma ordenada.
```

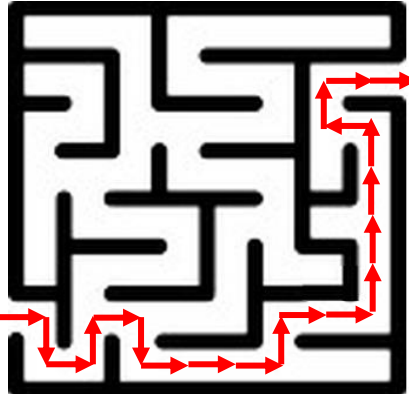
```
Lista* leMaiores(Lista *L, int K);
```

3. A empresa Pet Veterinária LTDA mantém uma lista diária com as informações de todos os animais em atendimento. Para cada animal, as informações de código, idade e peso são registradas. O código de um animal é dado pela ordem de chegada, por exemplo, o primeiro animal possui o código 1, o segundo animal o código 2, etc. Com base nessas informações, crie um programa em C++ para gerenciar a lista de animais como segue:
 - a. Implemente uma lista duplamente encadeada, projetada para armazenar os dados dos animais.
 - b. Crie um descritor para a lista contendo:
 - Ponteiro para o primeiro nó da lista
 - Ponteiro para o último nó da lista
 - Quantidade de nós
 - c. Implemente as seguintes funções:
 - Função para inserir um animal na lista;
 - Função para remover um elemento da lista com base no código do animal;
 - Função para imprimir os animais da lista em ordem de chegada;
 - Função para imprimir os 3 últimos animais da lista:
 - **Não é permitido usar laços (for/while).**
 - Acesse o último elemento da lista com base no descritor.
 - Para o penúltimo elemento, use o ponteiro “ant” do último nó.
 - Para o antepenúltimo elemento, use o ponteiro “ant” do penúltimo nó.
 - Antes de acessar um ponteiro, verifique se o mesmo não é nulo.
 - Função para separar a lista em duas novas listas, com base no peso do animal:
 - Dada uma lista L1 de entrada, a função deve criar duas novas listas L2 e L3, sendo que L2 deve armazenar uma cópia dos animais da lista L1 com peso

menor ou igual a 50kg e L3 uma cópia dos animais da lista L1 com peso maior que 50kg.

4. Considere um labirinto, contendo um ponto de entrada e um ponto de saída. Dado um percurso do ponto de entrada do labirinto até um ponto saída (caminho de ida), escreva um programa em C++ para computar o percurso inverso (caminho de volta).

a. Exemplo dos percursos de ida e volta:

Caminho de ida	Caminho de volta	
direita, baixo, direita, cima, direita, baixo, direita, direita, direita, cima, direita, direita, cima, cima, cima, cima, esquerda, cima, direita, direita.	esquerda, esquerda, baixo, direita, baixo, baixo, baixo, baixo, esquerda, esquerda, baixo, esquerda, esquerda, esquerda, cima, esquerda, baixo, esquerda, cima, esquerda.	

b. Implemente o programa usando uma lista duplamente encadeada.

c. Exemplo de entrada/saída.

```
IDA: direita, baixo, direita, cima, direita, baixo, direita, direita, direita, cima, direit
a, direita, cima, cima, cima, cima, esquerda, cima, direita, direita

VOLTA: esquerda, esquerda, baixo, direita, baixo, baixo, baixo, baixo, esquerda, esquerda,
baixo, esquerda, esquerda, esquerda, cima, esquerda, baixo, esquerda, cima, esquerda
```