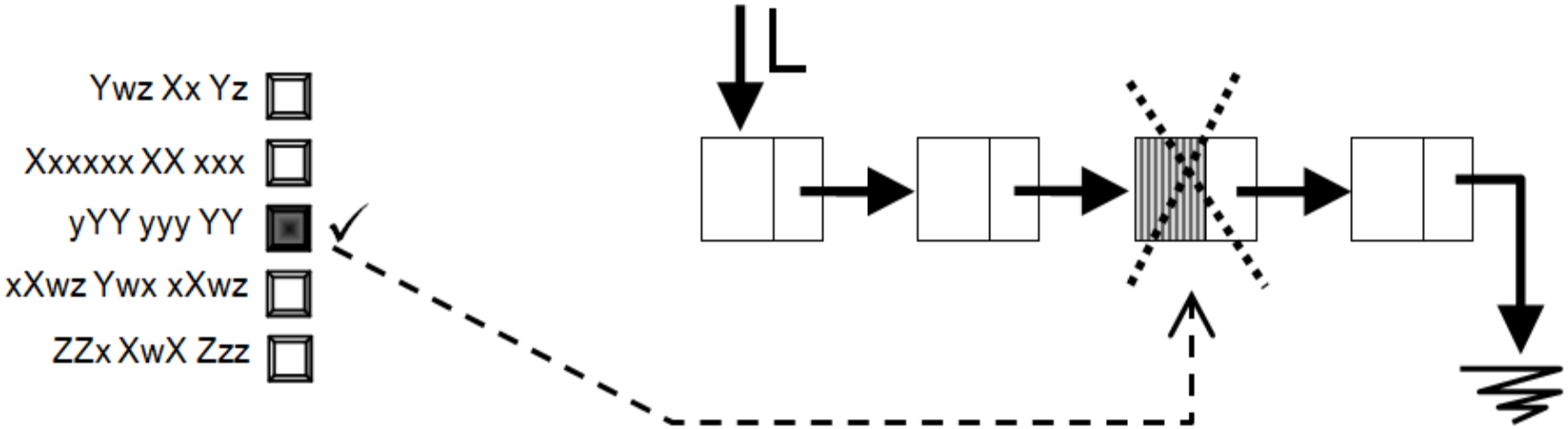
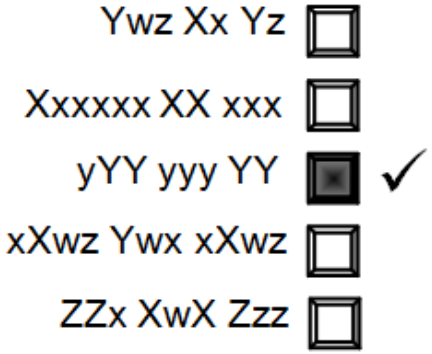
Estruturas de Dados com Jogos



Capítulo 6

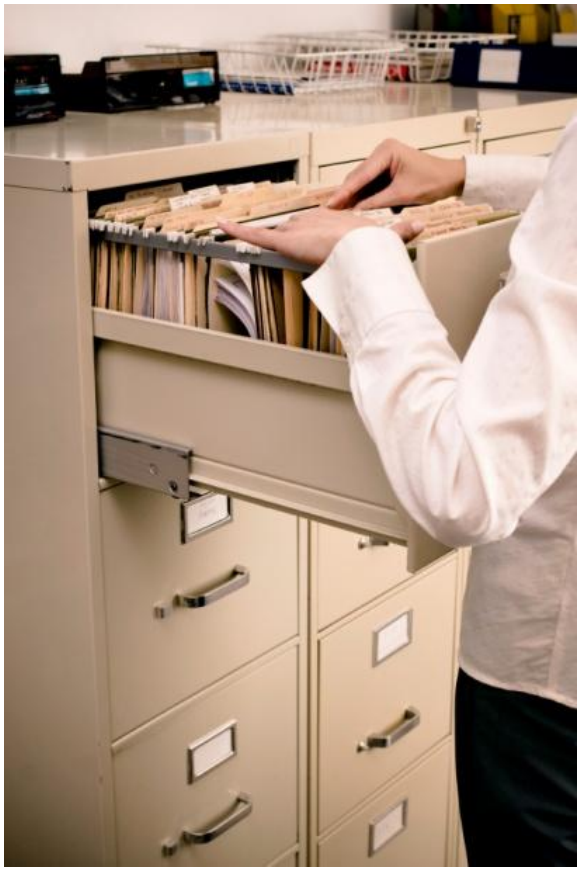
Listas Cadastrais



Seus Objetivos neste Capítulo

- Entender o que é e para que serve uma estrutura do tipo Lista Cadastral;
- Desenvolver habilidade para manipular Listas Cadastrais através de seus operadores primitivos;
- Ganhar experiência na elaboração de algoritmos sobre listas encadeadas, implementando Listas Cadastrais como listas encadeadas ordenadas, listas não ordenadas, listas circulares, listas com elementos repetidos e outras variações;
- Iniciar o desenvolvimento do seu jogo referente ao Desafio 3.

Atualizando um Cadastro de Funcionários

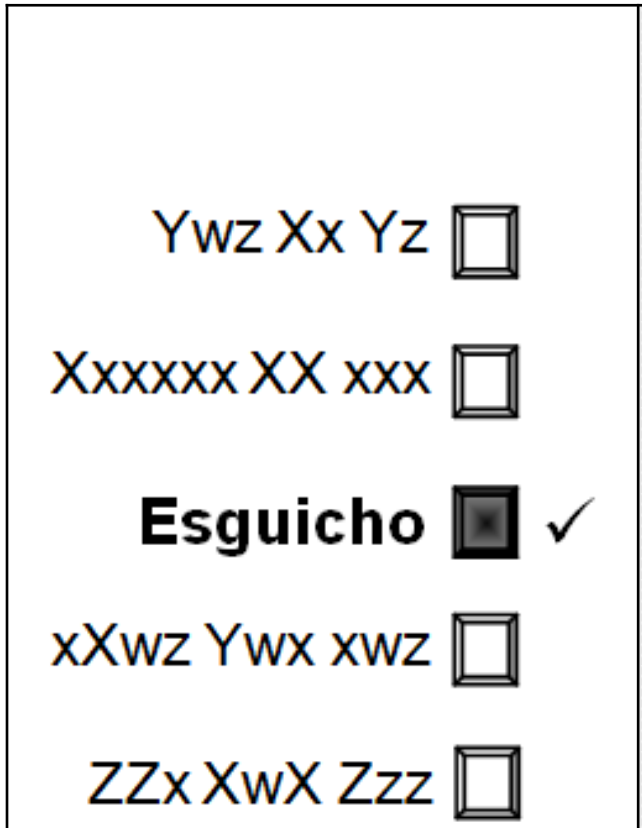


iStockPhoto

Se o funcionário de nome ***Moacir*** deve ser desligado da empresa:

- **Procuramos** a pastinha do ***Moacir****; e*
- **Retiramos**, especificamente, a pastinha que guarda os dados do ***Moacir*** - esteja ela onde estiver.

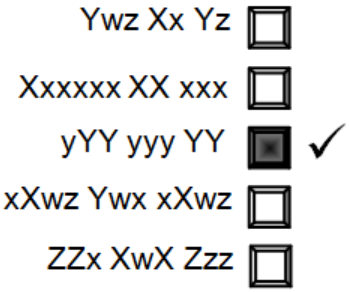
Atualizando uma Lista de Compras



Se queremos retirar o item *Esguicho* da Lista:

- Procuramos na Lista o item cujo valor é ***Esguicho***;
- Retiramos da Lista, especificamente, o item cujo valor é ***Esguicho*** - esteja esse item no começo, no meio ou no final da Lista.

O Que É uma Lista Cadastral?



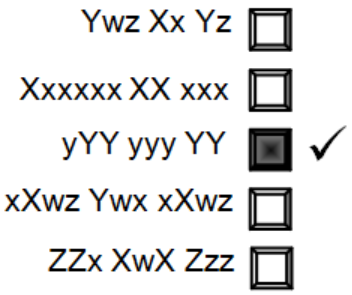
Definição: Lista Cadastral

Em uma estrutura de armazenamento denominada Lista Cadastral, a inserção, a retirada, e o acesso aos elementos do conjunto ocorrem em função do valor dos elementos, e não em função da posição dos elementos no conjunto.

Lista Cadastral, Cadastro ou Lista?

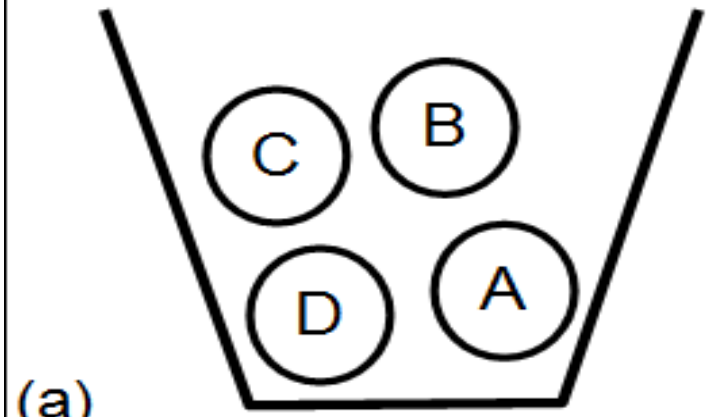
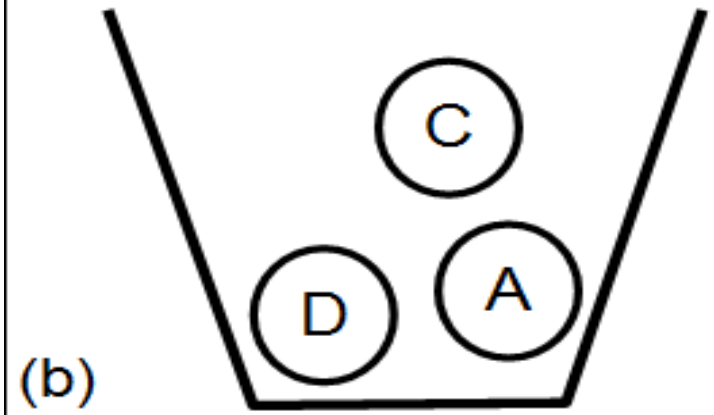
Os três termos podem ser utilizados. Mas não confunda a estrutura de armazenamento *Lista Cadastral* com as *Listas Encadeadas* - técnica de alocação encadeada de memória.

Operações de uma Lista Cadastral

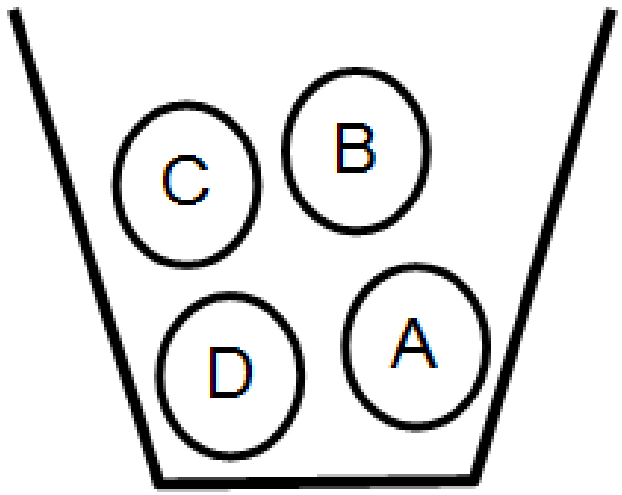


EstáNaLista (L, X)
Insere (L, X, Ok)
Retira(L, X, Ok)
Vazia(L)
Cheia(L)
Cria(L)
PegaOPrimeiro(L, X, TemElemento)
PegaOPróximo(L, X, TemElemento)

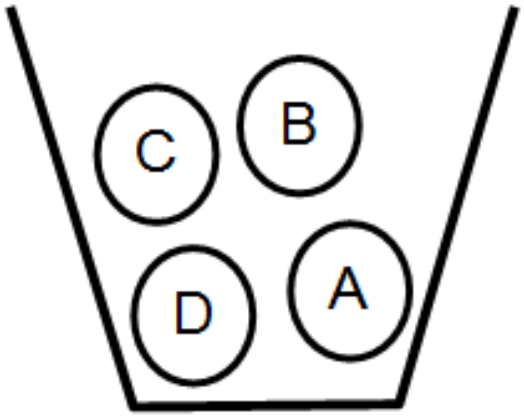
Operações Insere, Retira e EstáNaLista

Lista L	Operação
 (a)	Insere(L, 'A', Ok) EstáNaLista(L, 'F') EstáNaLista(L, 'B') Retira(L, 'F', Ok)
 (b)	Retira(L, 'B', Ok)

Operações PegaOPrimeiro e PegaOPróximo

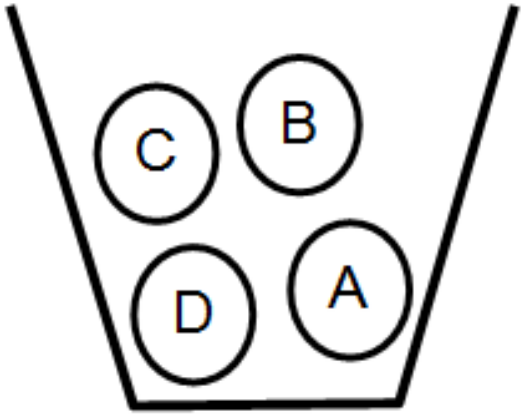
Lista L	Operação
	PegaOPrimeiro(L, X, TemElemento)
	PegaOPróximo(L, X, TemElemento)
	PegaOPróximo(L, X, TemElemento)
	PegaOPróximo(L, X, TemElemento)
	PegaOPróximo(L, X, TemElemento)
	PegaOPrimeiro(L, X, TemElemento)

Exercício 6.1 Imprimir Todos os Elementos de uma Lista

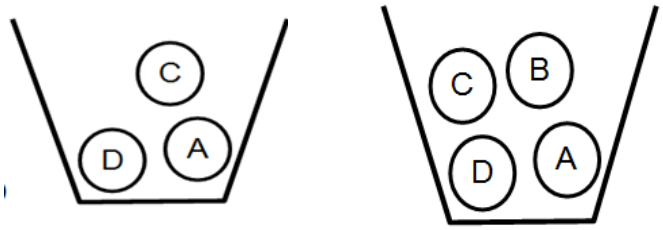


ImprimeTodos (parâmetro por referência **L** do tipo Lista);

/* Imprime todos os valores armazenados na Lista L */




```
ImprimeTodos (parâmetro por referência L do tipo Lista) {
/* Imprime todos os valores armazenados na Lista L  */

Variável X do tipo Char;
Variável TemElemento do tipo Boolean;
PegaOPrimeiro( L, X, TemElemento );     // … se existir
Enquanto (TemElemento == Verdadeiro)  Faça {
                        { Imprime( X );
                        PegaOPróximo( L, X, TemElemento ); }
} // fim ImprimeTodos
```



Exercícios

Exercício 6.2 Intersecção de Duas Listas

Intersecção (parâmetros por referência L1, L2, L3 do tipo Lista);

/* Recebe L1 e L2, e cria L3 contendo a intersecção entre L1 e L2. Ou seja, L3 terá todos os elementos que pertencem tanto a L1 quanto a L2 */

Exercício 6.3 Posição do Elemento no Conjunto

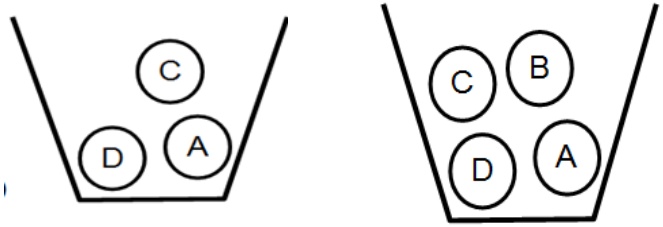
Inteiro PosiçãoNoConjunto (parâmetro por referência L do tipo Lista, parâmetro X do tipo char);

/* Retorna valor numérico que indica a posição de X na Listra L. Por exemplo, retorna 1 se X for o primeiro elemento da Lista, e 0 se X não estiver em L */

Exercício 6.4 Estão em L2 e Não Estão em L1

EmL2MasNãoEmL1 (parâmetros por referência L1, L2, L3 do tipo Lista)

/* Recebe L1 e L2, e cria L3 contendo todos os elementos que estão em L2 mas que não estão em L1 */

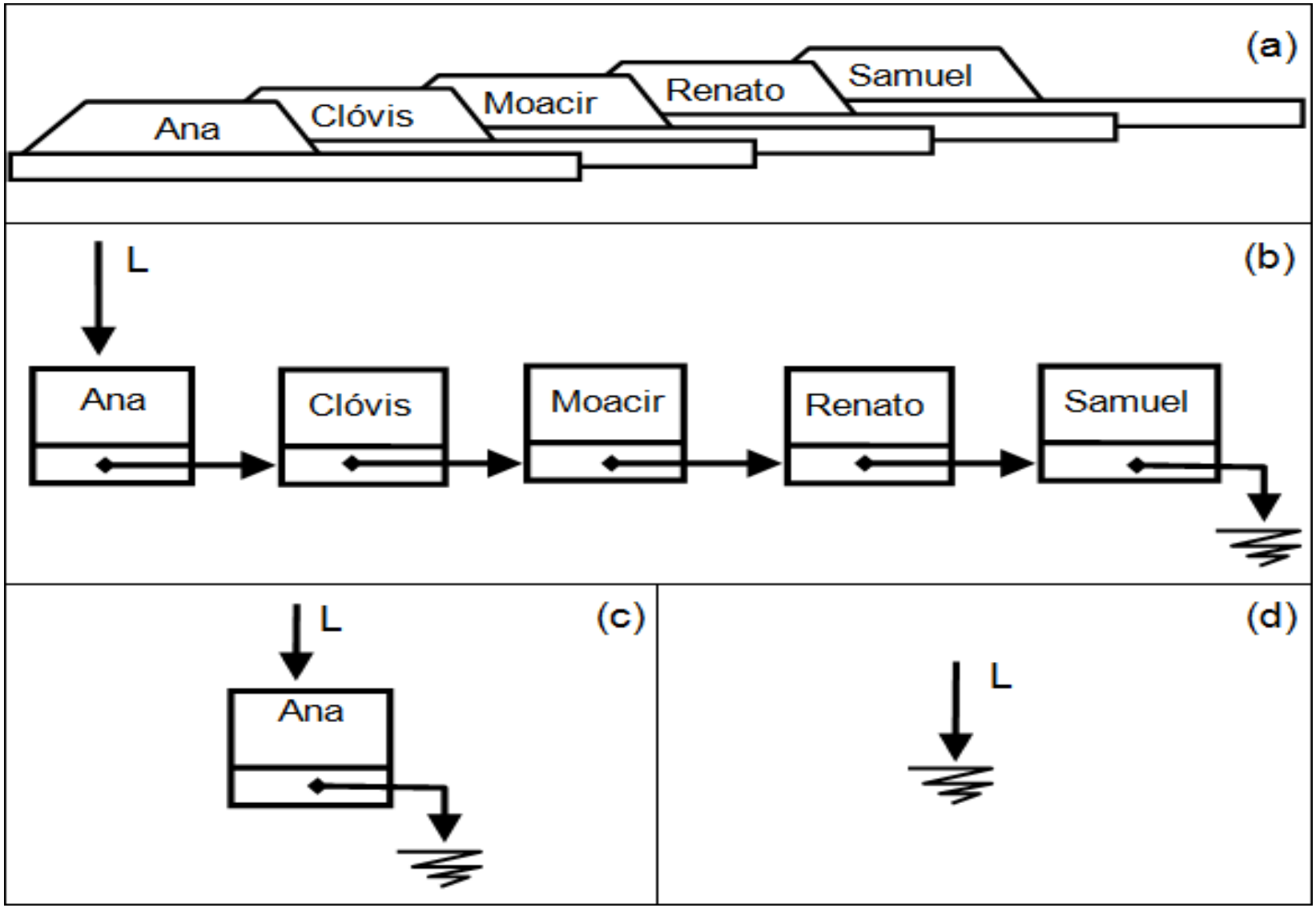


Intersecção (parâmetros por referência **L1**, **L2**, **L3** do tipo Lista) {

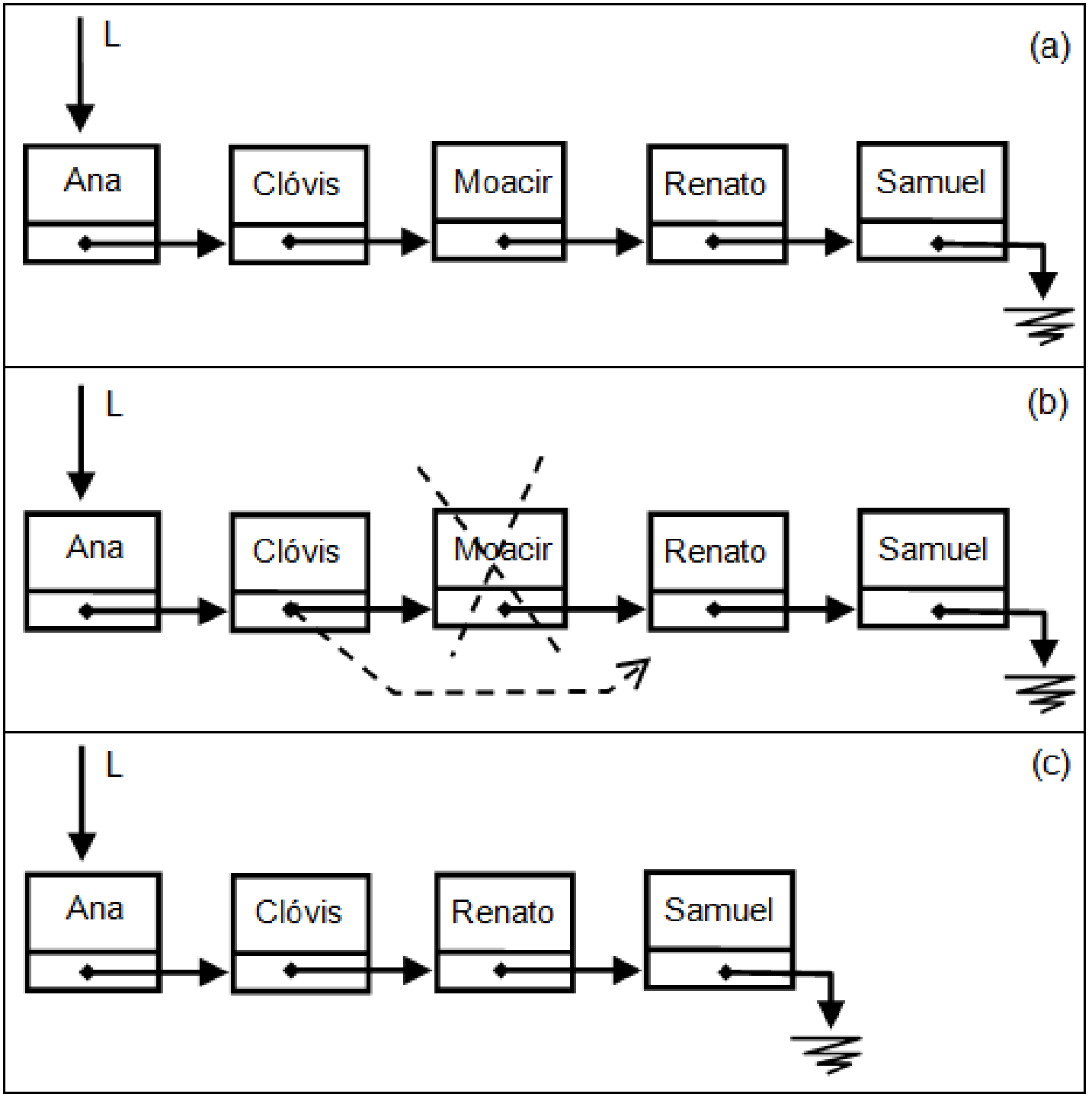
/* Recebe L1 e L2, e cria L3 contendo a intersecção entre L1 e L2. Ou seja, L3 terá todos os elementos que pertencem a L1 e também a L2 */

```
Variável X do tipo Char;
Variável TemElemento do tipo Boolean;
Variável Ok do tipo Boolean;
Cria(L3);                          // cria a Lista L3, vazia

/* para cada elemento de L1, verifica se está também em L2; se estiver, insere em L3 */
PegaOPrimeiro( L1, X, TemElemento );          // pega o primeiro de L1
Enquanto (TemElemento == Verdadeiro) Faça {
        Se (EstaNaLista(L2, X)==Verdadeiro) // Elemento X de L1 está também em L2?
        Então Insere (L3, X, Ok);           // Se estiver, insere X em L3
        PegaOPróximo( L1, X, TemElemento ); }    // pega o próximo de L1

} // fim Intersecção
```

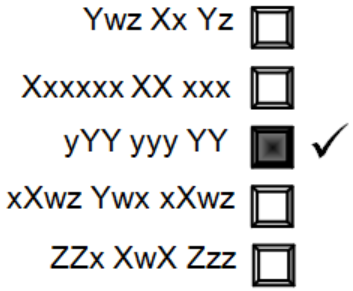
Lista Cadastral Sem Elementos Repetidos como uma Lista Encadeada Ordenada



Retira Elemento

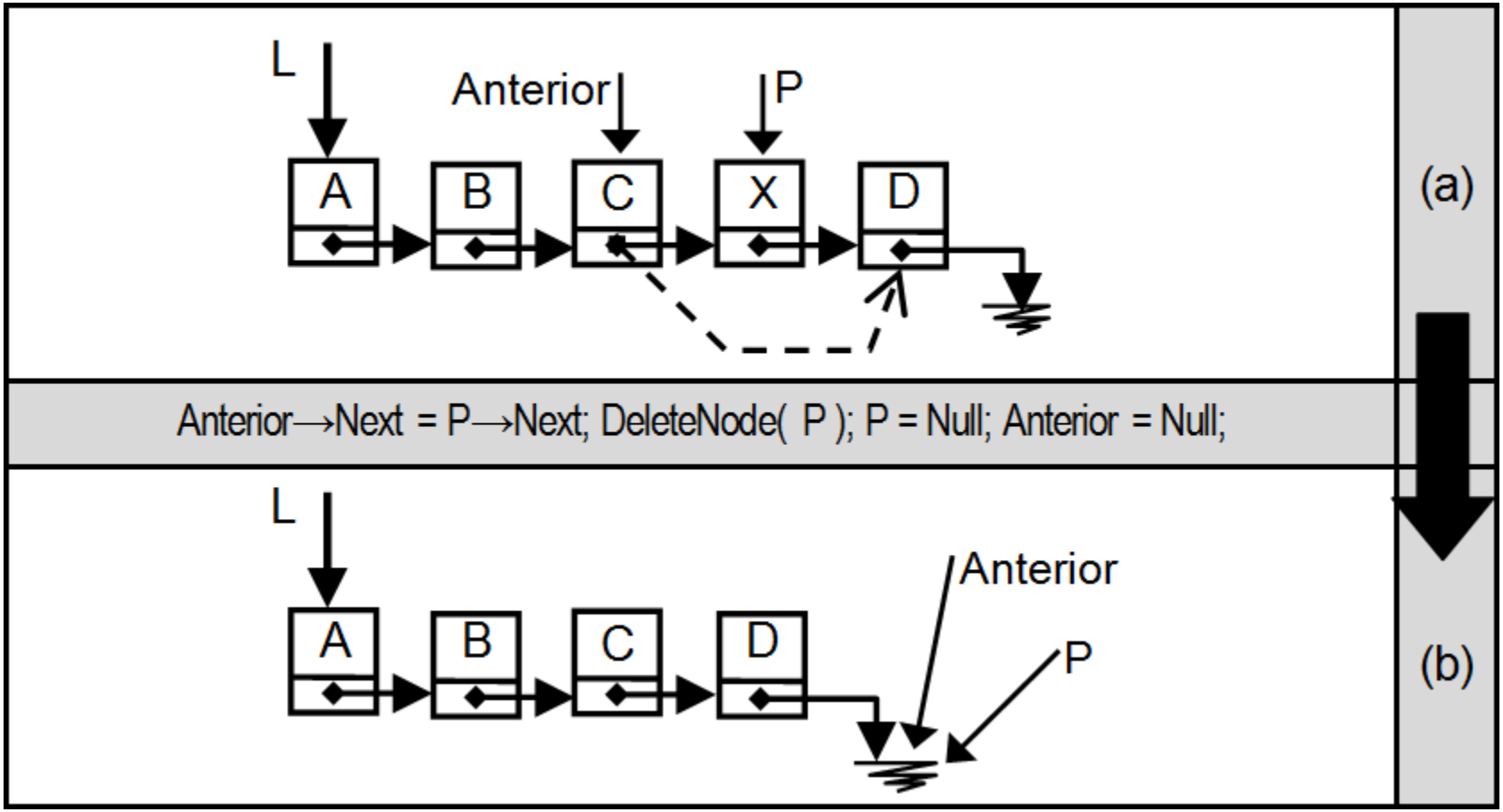


Algoritmo Retira Elemento - Primeira Versão:



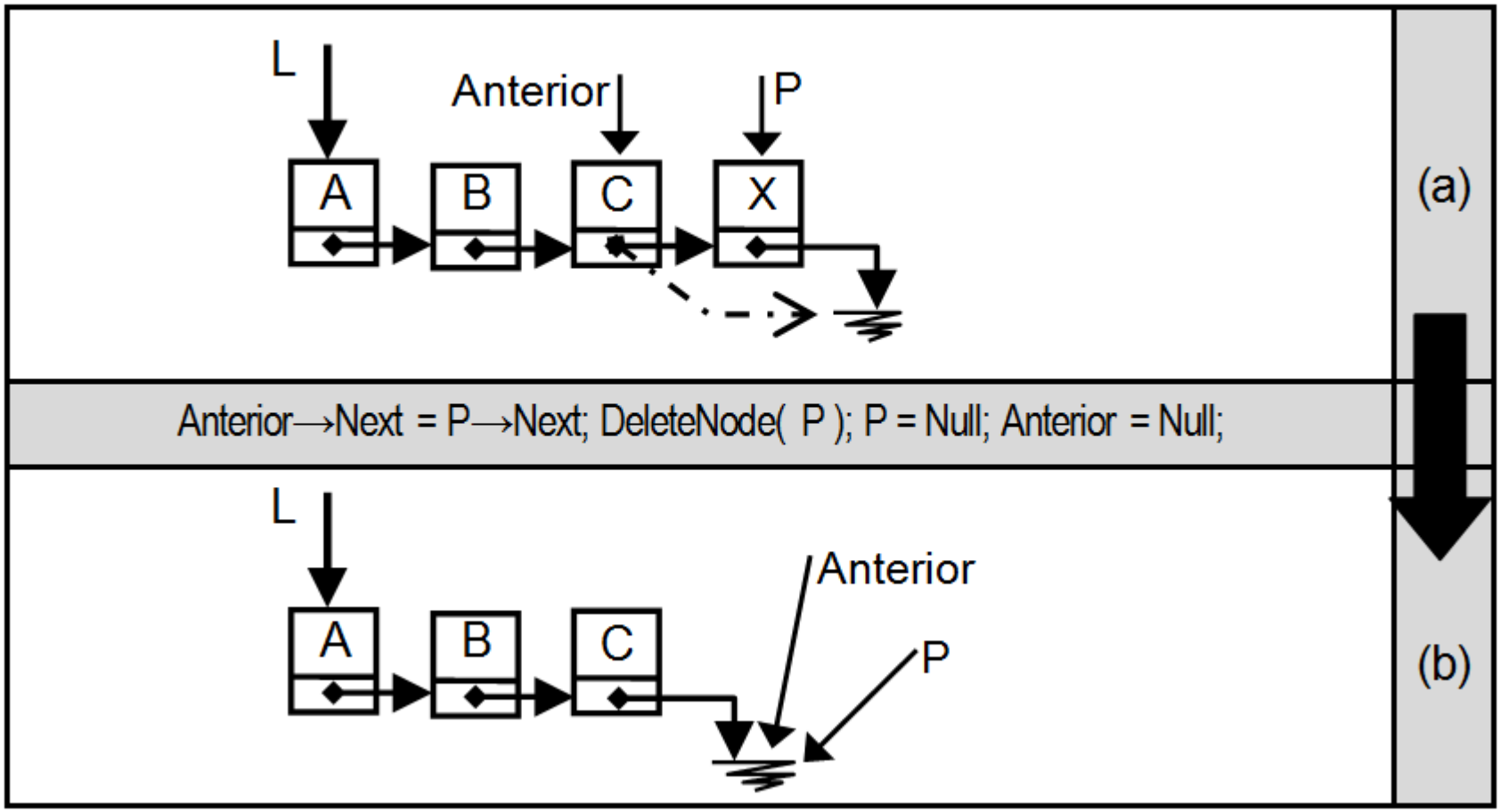
- Passo 1: Procuramos na Lista L o Nó que contém o valor X - sendo X o elemento a ser removido;
- Passo 2: Eliminamos da Lista L o Nó que guarda o valor X.

Caso 1: X no Meio da Lista



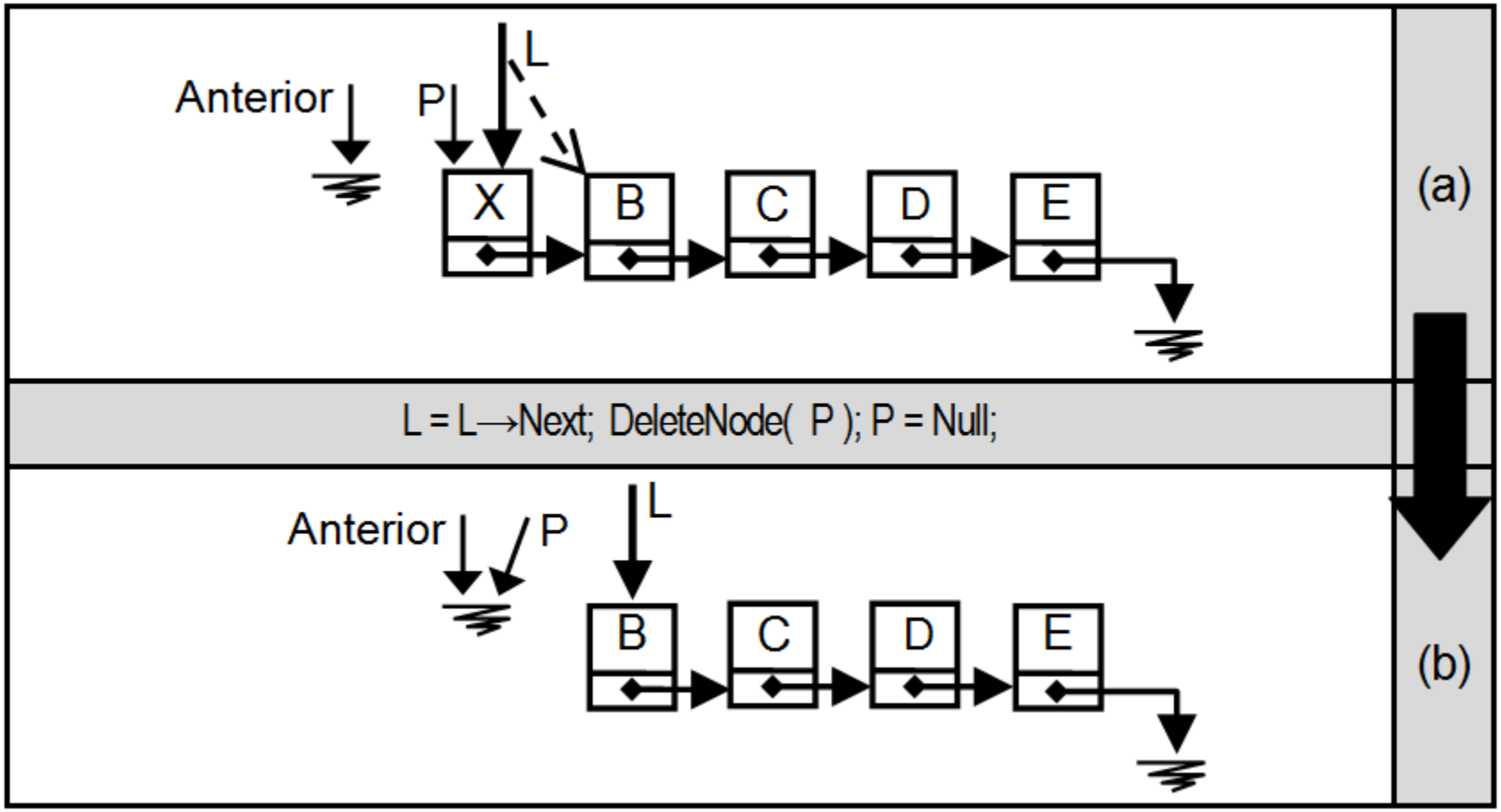
Retira Elemento

Caso 1': X no Final da Lista



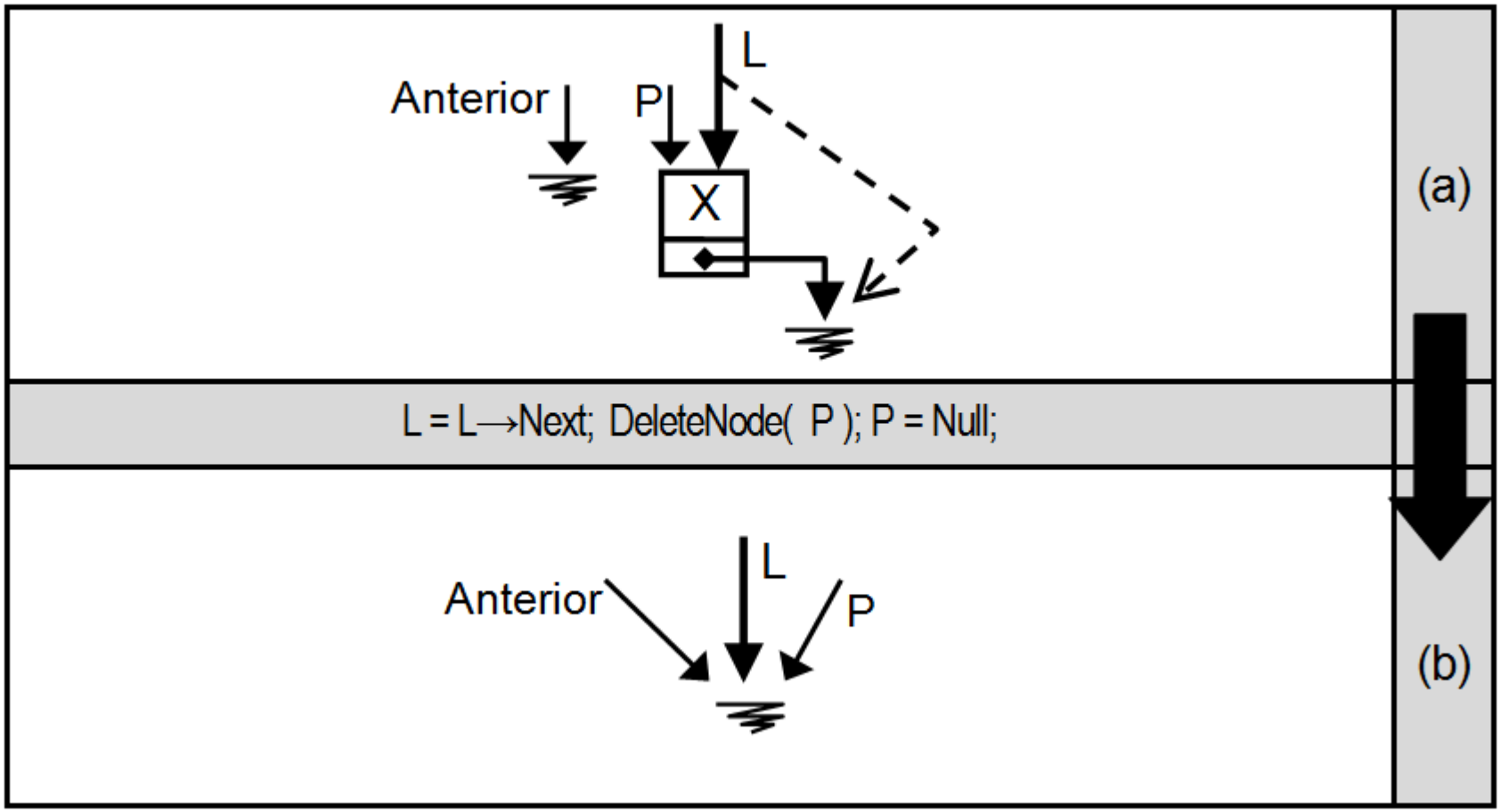
Retira Elemento

Caso 2: X no Inicio da Lista



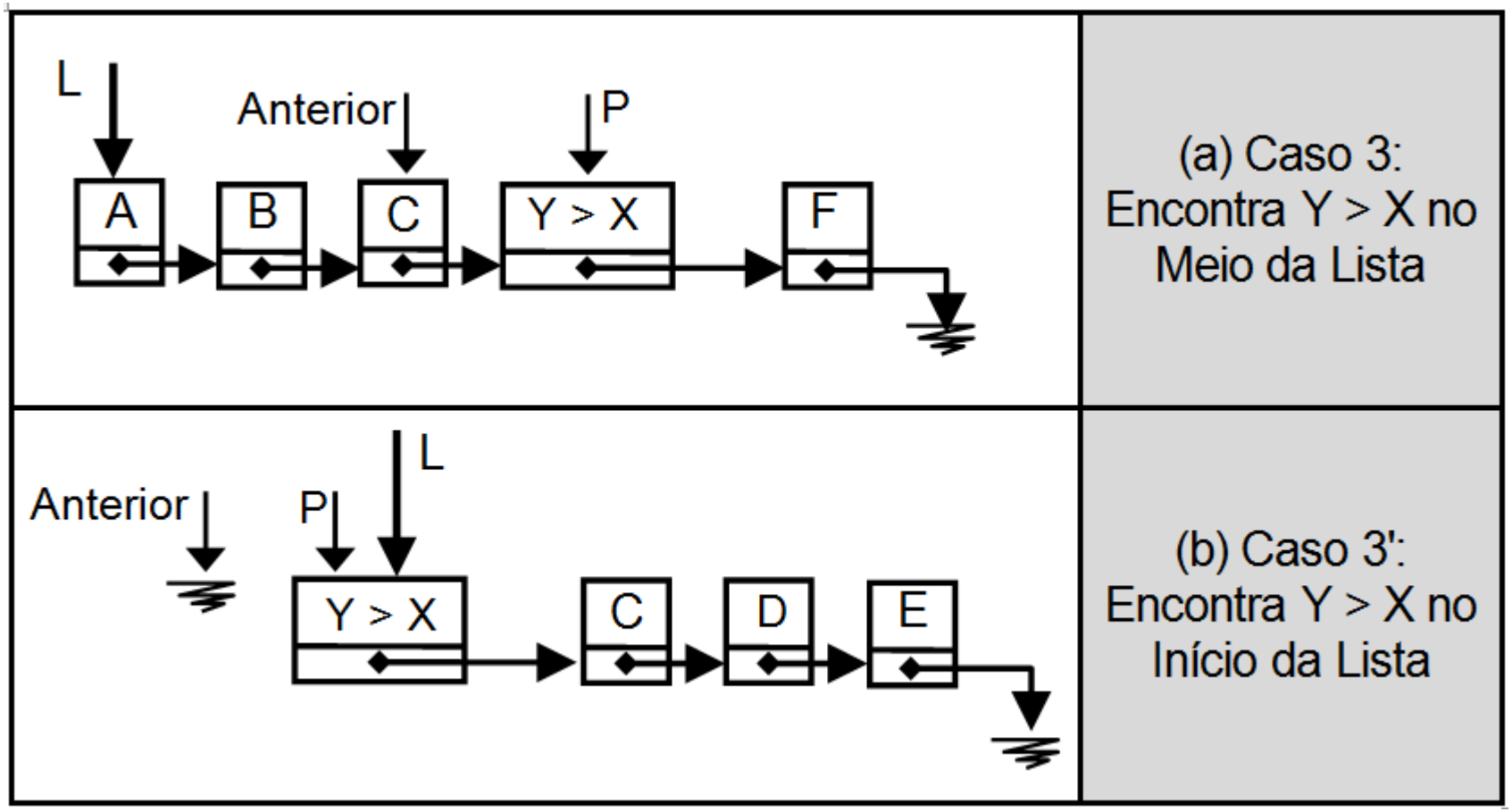
Retira Elemento

Caso 2': X como Único Elemento da Lista



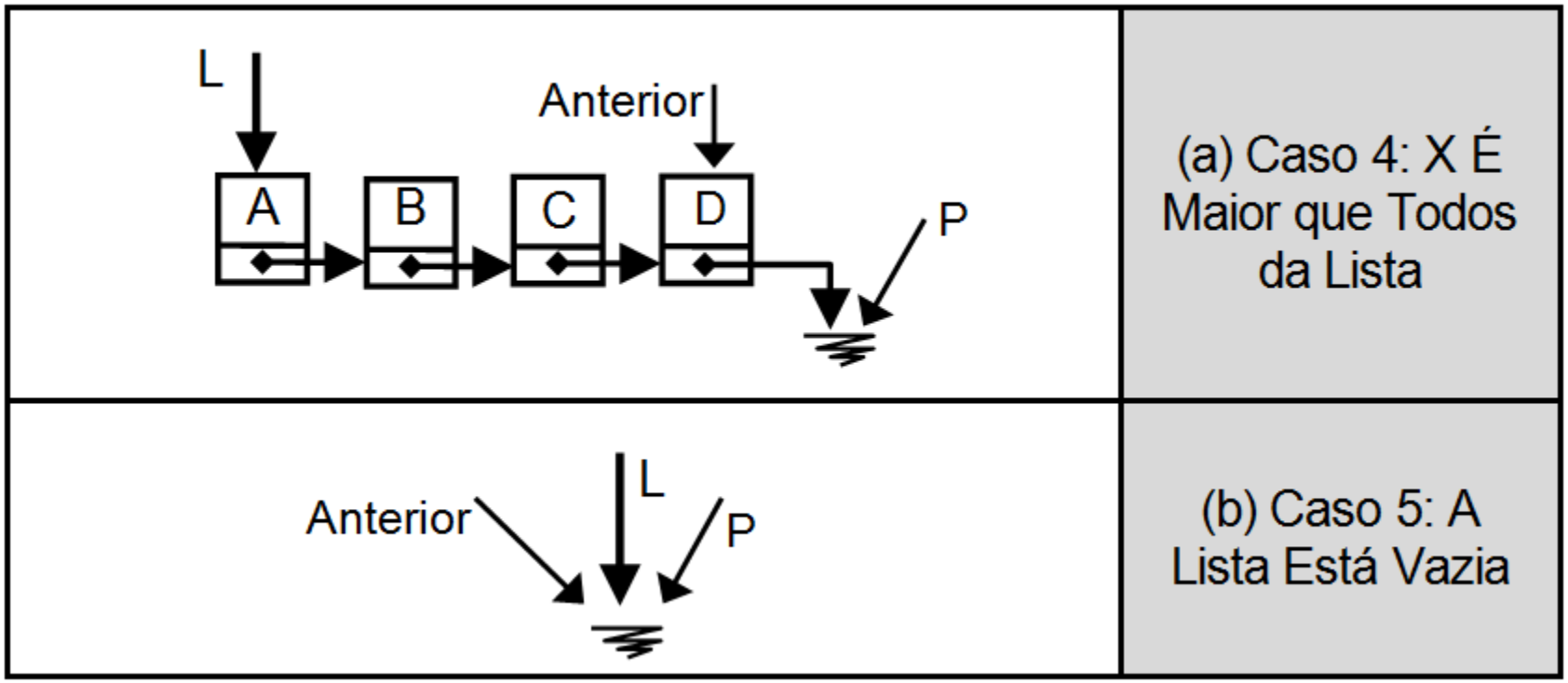
Retira Elemento

Casos 3 e 3': Encontra Y > X no Meio da Lista e Encontra Y > X no Início da Lista



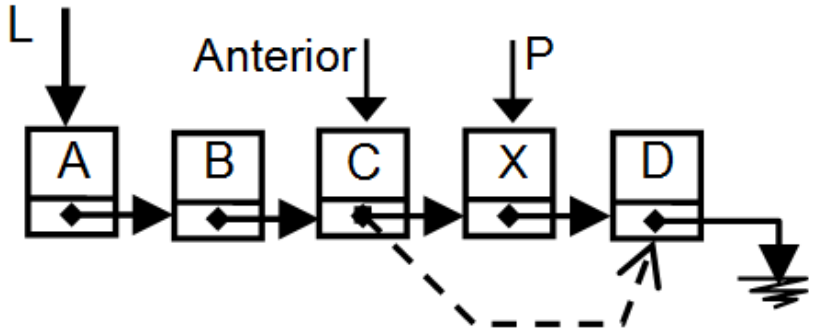
Retira Elemento

Caso 4: X Maior que Todos da Lista e Caso 5: Lista vazia

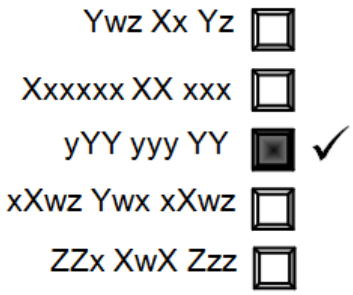


Retira Elemento

Retira Elemento – Segunda Versão:



- Passo 1: Definir variáveis temporárias P e Anterior. Anterior começa em Null, P começa em L. Avança Anterior e P até que P chegue ao Nó que contém X, ou a um Nó que contém um valor Y > X, ou ao final da Lista (Null). Anterior avança sempre uma posição atrás de P.
- Passo 2: Com base na posição de Anterior e P, e com base no valor armazenado no Nó apontado por P (caso P for diferente de Null), identificar o caso (Caso 1, Caso 1', Caso 2, Caso 2', Caso 3, Caso 3', Caso 4 ou Caso 5) e executar as ações necessárias.



Exercício 6.5 Retira Elemento de Lista Cadastral Implementada como Lista Encadeada Ordenada

Retira (parâmetro por referência **L** do tipo Lista, parâmetro **X** do tipo Char, parâmetro por referência **Ok** do tipo Boolean);

/* Caso X for encontrado na Lista L, retira X da Lista e Ok retorna Verdadeiro. Se X não estiver na Lista L, não retira nenhum elemento, e Ok retorna o valor Falso */

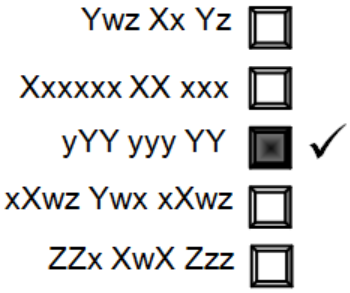
Retira (parâmetro por referência **L** do tipo Lista, parâmetro **X** do tipo Char, parâmetro por referência **Ok** do tipo Boolean) {

```
Variáveis P, Anterior do tipo NodePtr;        // Tipo NodePtr = ponteiro para Nó
Variável AchouX do tipo Boolean;

ProcuraX (L, X, P, Anterior, AchouX);         /*  ProcuraX: executa o passo 1 */

/* Passo 2: se encontrou X, remover da Lista o Nó que contém X */
Se (AchouX == Verdadeiro)                     // se X foi encontrado na Lista
Então {     Se (P != L)                       // se X não estiver no primeiro Nó da Lista
            Então {     Anterior→Next = P→Next; // Casos 1 ou 1': X no meio ou no último
                        DeleteNode( P );              // Nó da Lista L - veja o Quadro 6.14 e
                        P = Null;                     // o Quadro 6.15
                        Anterior = Null }
            Senão {     L = L→Next;           // Casos 2 ou 2': X no primeiro Nó da Lista
                        DeleteNode( P );              // veja os Quadros 6.16 e 6.17
                        P = Null }
            Ok = Verdadeiro;
        }
Senão       Ok = Falso;  // X não foi encontrado - casos 3, 3', 4 ou 5; não retira elemento
}           // fim Retira
```

ProcuraX (parâmetros por referência **L** do tipo Lista, parâmetro por referência **X** do tipo Char, parâmetros por referência **P**, **Anterior** do tipo NodePtr, parâmetro por referência **AchouX** do tipo Boolean) {

```
P = L;                  // P começa em L
Anterior = Null;  // Anterior começa em Null

/* avança P e Anterior até encontrar X, ou Y > X, ou Null... Anterior corre atrás de P */
Enquanto ((P != Null) E (P→Info < X)) Faça
            { Anterior = P;
            P = P→Next; }
Se ( (P != Null) E (P→Info == X) )
Então AchouX = Verdadeiro;
Senão AchouX = Falso;
}  // fim ProcuraX
```

Exercícios



Exercício 6.6 Insere Elemento

Insere (parâmetro por referência L do tipo Lista, parâmetro X do tipo Char, parâmetro por referência Ok do tipo Boolean);

/* Caso o valor X já não estiver na Lista L, insere X e Ok retorna Verdadeiro. Se X já estiver na Lista L, não insere nenhum elemento, e Ok retorna o valor Falso */

Exercício 6.7 Está na Lista?

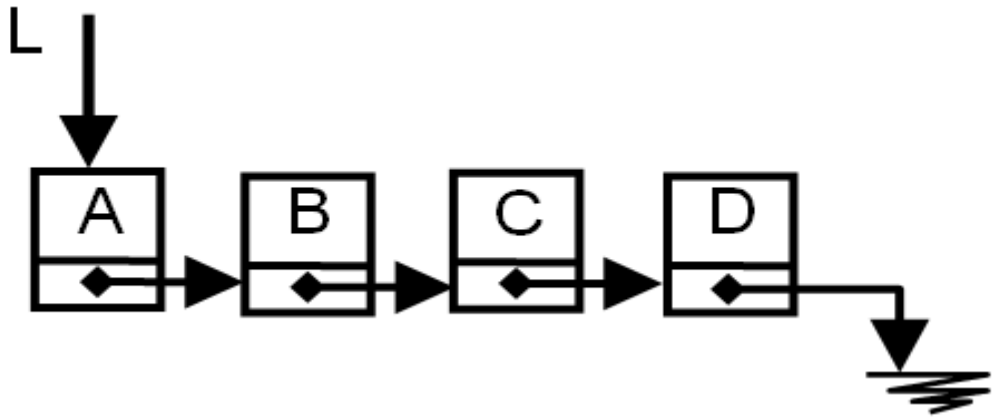
Boolean EstáNáLista (parâmetro por referência L do tipo Lista, parâmetro X do tipo Char);

/* Se X for encontrado na Lista , retorna Verdadeiro; Falso caso contrário */

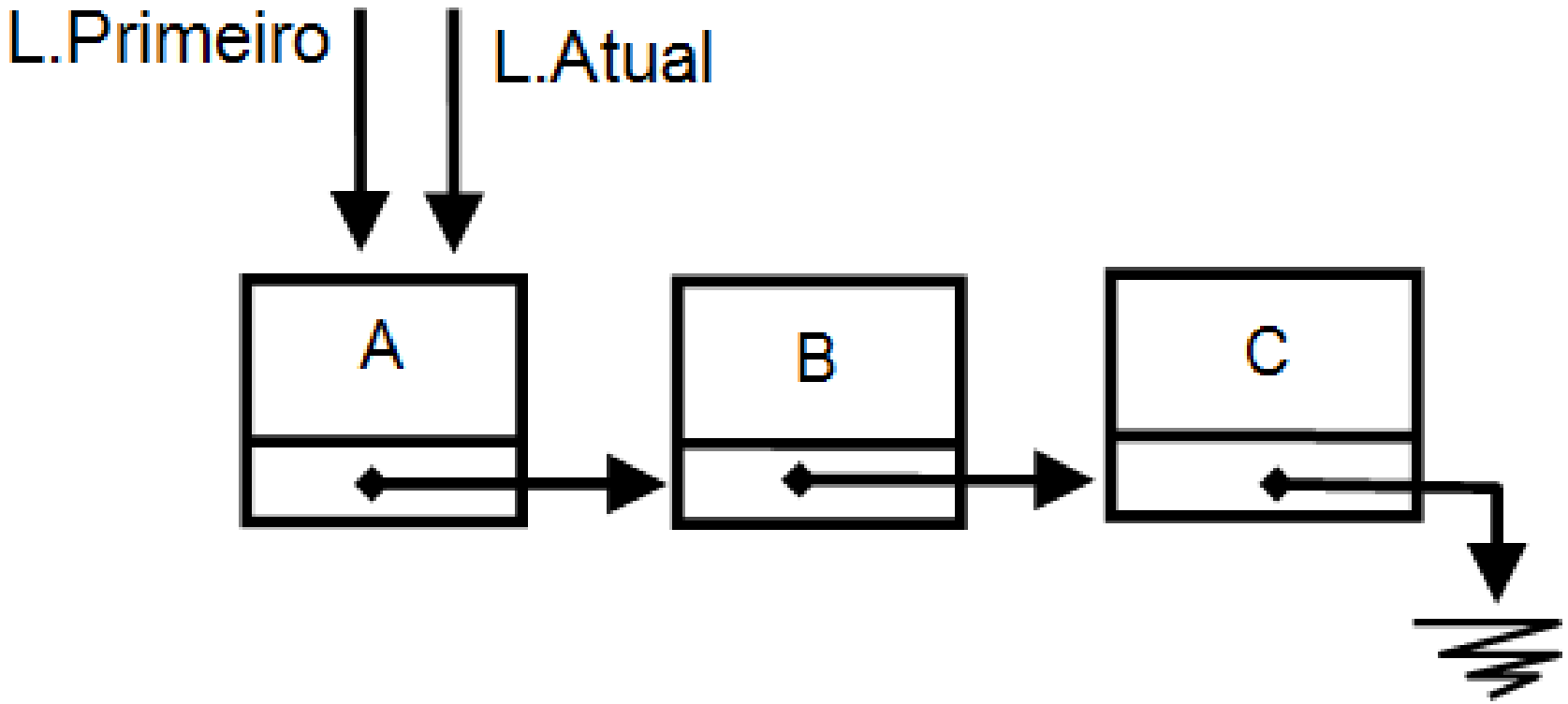
Exercício 6.8, 6.9, 6.10 Operações Cria, Vazia e Cheia

Operações para Percorrer a Lista – Como Implementar?

PegaOPrimeiro (L, X, TemElemento)
PegaOPróximo (L, X, TemElemento)

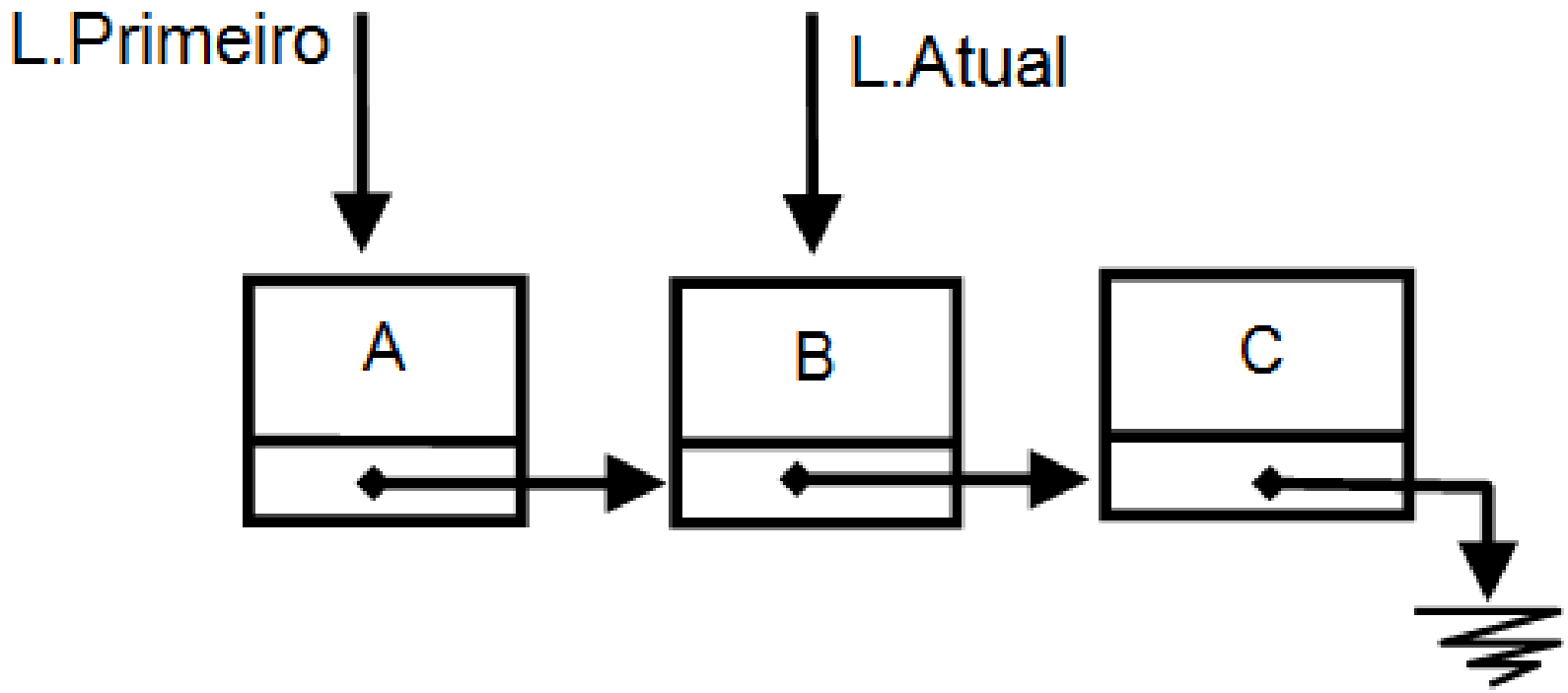


Operação
PegaOPrimeiro (L, X, TemElemento)



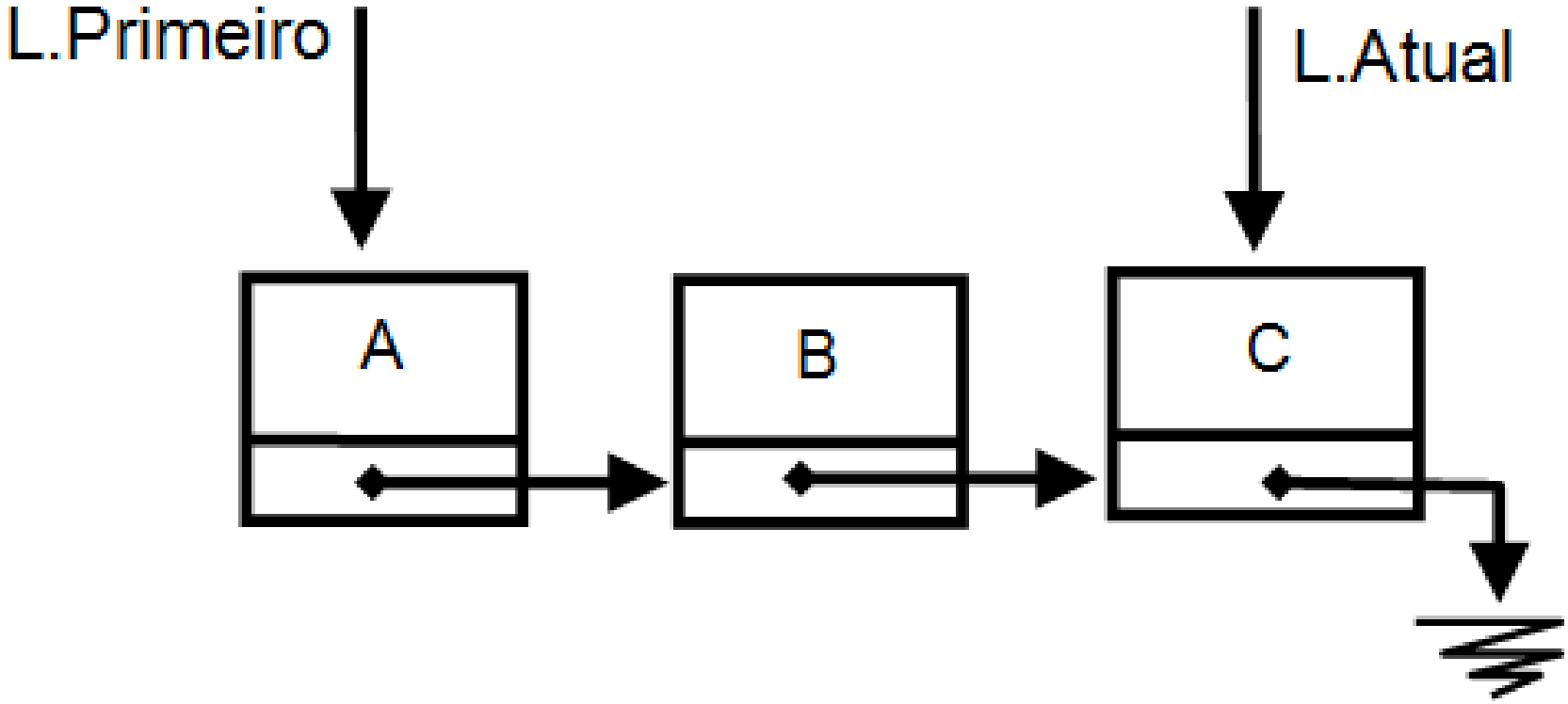
Resultado	
TemElemento: Verdadeiro	X: A

Operação
PegaOProximo (L, X, TemElemento)



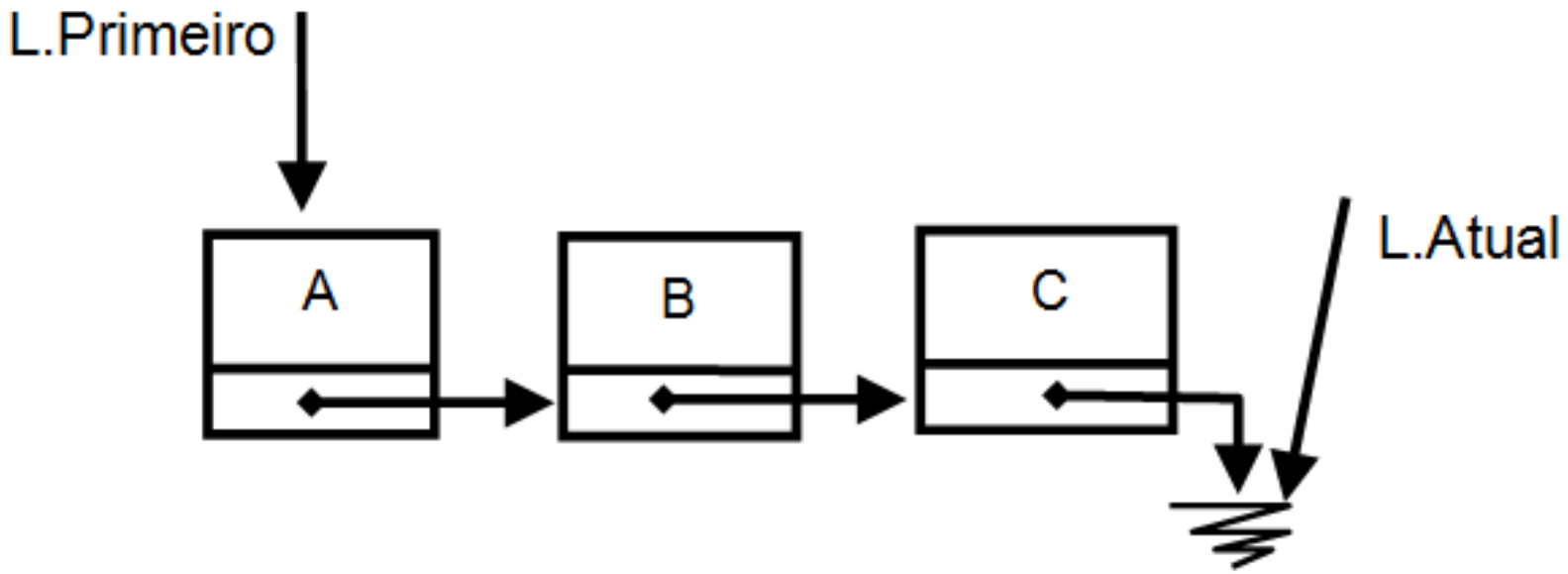
Resultado	
TemElemento: Verdadeiro	X: B

Operação
PegaOProximo (L, X, TemElemento)



Resultado	
TemElemento: Verdadeiro	X: C

Operação
PegaOProximo (L, X, TemElemento)



Resultado	
TemElemento: Falso	X: ?

Exercícios

Exercício 6.11 Pega o Primeiro

PegaOPrimeiro (parâmetro por referência **L** do tipo Lista, parâmetro por referência **X** do tipo Char, parâmetro por referência **TemElemento** do tipo Boolean);

/* Caso a lista estiver vazia, TemElemento retorna o valor Falso. Caso a lista não estiver vazia, TemElemento retornará Verdadeiro, e o valor do primeiro elemento da Lista retornará no parâmetro X */

Exercício 6.12 Pega o Próximo

PegaOPróximo (parâmetro por referência **L** do tipo Lista, parâmetro por referência **X** do tipo Char, parâmetro por referência **TemElemento** do tipo Boolean);

/* Caso a lista não estiver vazia, e caso houver um **próximo elemento** em relação à última chamada de PegaOPrimeiro ou PegaOProximo, TemElemento retornará Verdadeiro, e o valor do **próximo elemento** retornará no parâmetro X. Caso a lista estiver vazia, ou caso não houver um próximo elemento em relação à última chamada, o parâmetroTemElemento retornará o valor Falso */

Exercício 6.13 Revisar os Exercícios 6.5 a 6.10, implementando uma Lista com Dois Ponteiros.

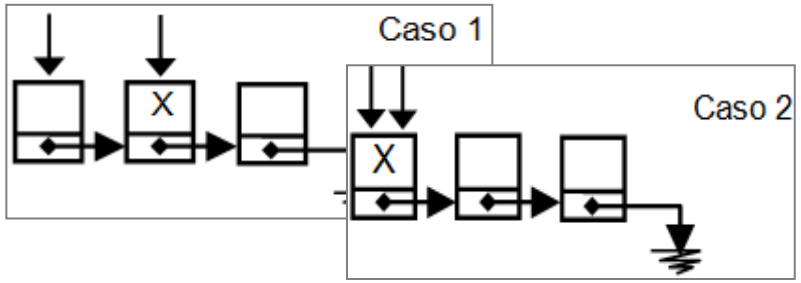
Exercício 6.11
Pega o Primeiro

PegaOPrimeiro (parâmetro por referência **L** do tipo Lista, parâmetro por referência **X** do tipo Char, parâmetro por referência **TemElemento** do tipo Boolean) {

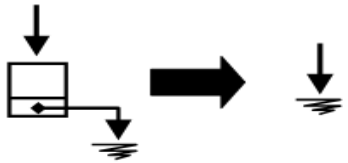
```
L.Atual = L.Primeiro;  // L.Atual passa a apontar para onde aponta L.Primeiro
Se (L.Atual != Null)    // verifica se existe um primeiro elemento.... se existir,
Então {   TemElemento = Verdadeiro; // ... TemElemento retornará Verdadeiro
          X = L.Atual→Info; }  //... e X retornará o valor do primeiro elemento
Senão     TemElemento = Falso;
} // fim PegaOPrimeiro
```

Passos Para Construir uma Boa Solução

Passo 1: Identificar Casos e Desenhar a Situação Inicial.



Passo 2: Desenho da Situação Final.



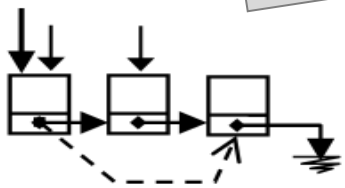
Passo 3: Algoritmo para Tratar Cada Caso, Separadamente.

Para Tratar Caso 2:
- DeleteNode(P);
- L=Null;

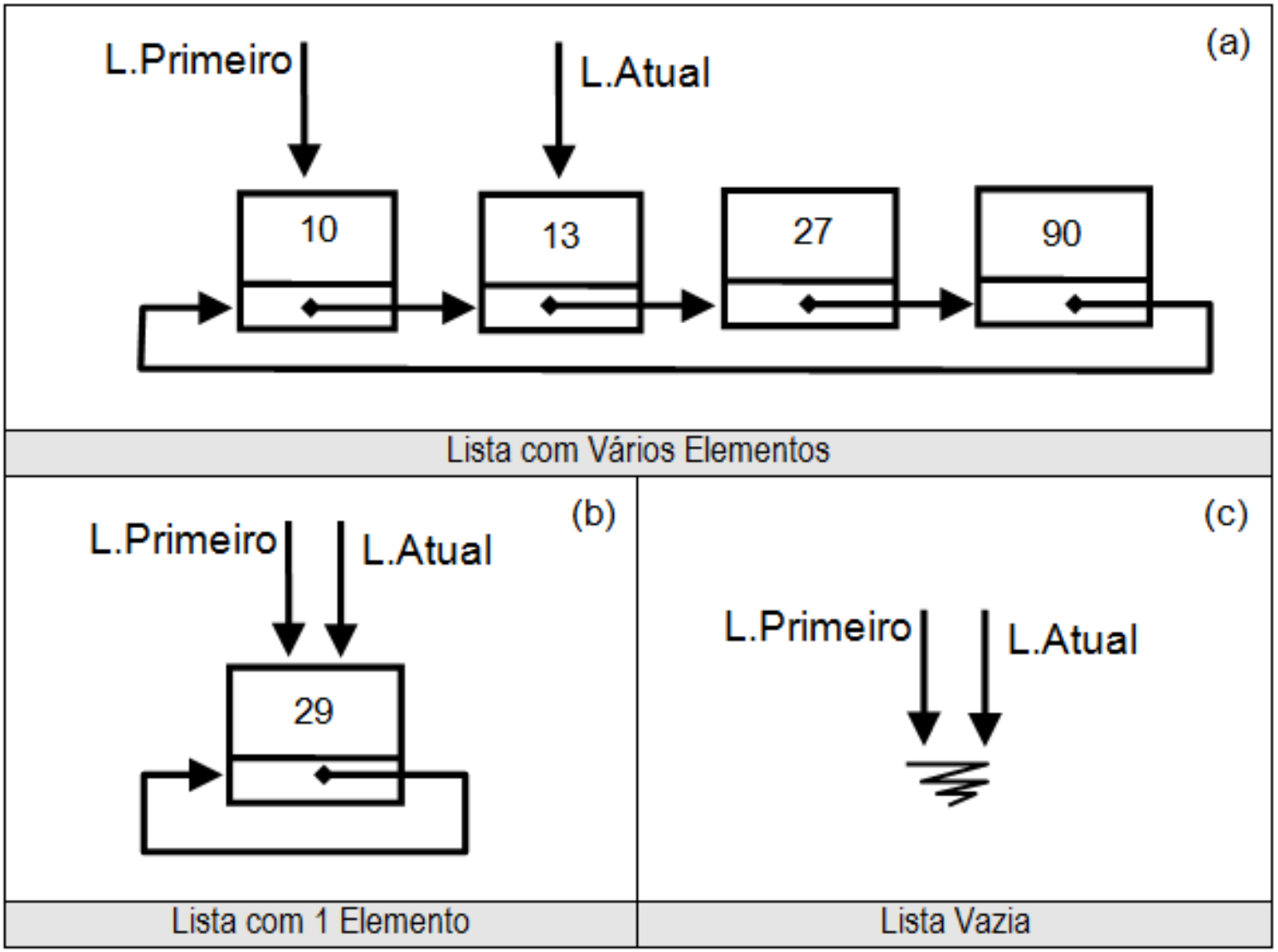
Passo 4: Faça um Algoritmo Geral do Modo Mais Simples.

```
Se Caso 1
Então [tratar Caso 1]
Senão Se Caso 2
        Então [tratar Caso 2]
        Senão Se Caso 3...
```

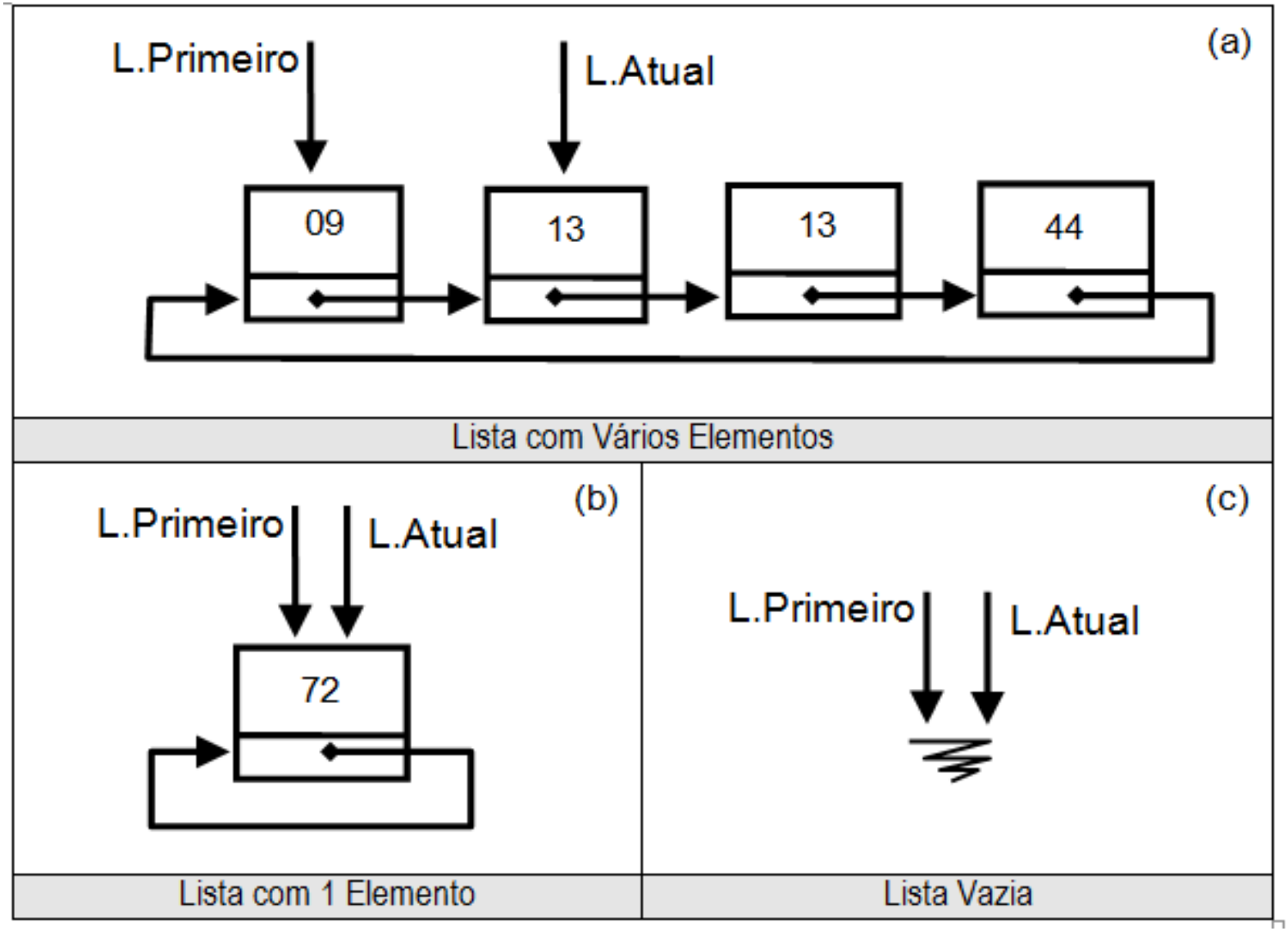
Passo 5: Testar Cada Caso, Alterando o Desenho Passo a Passo.



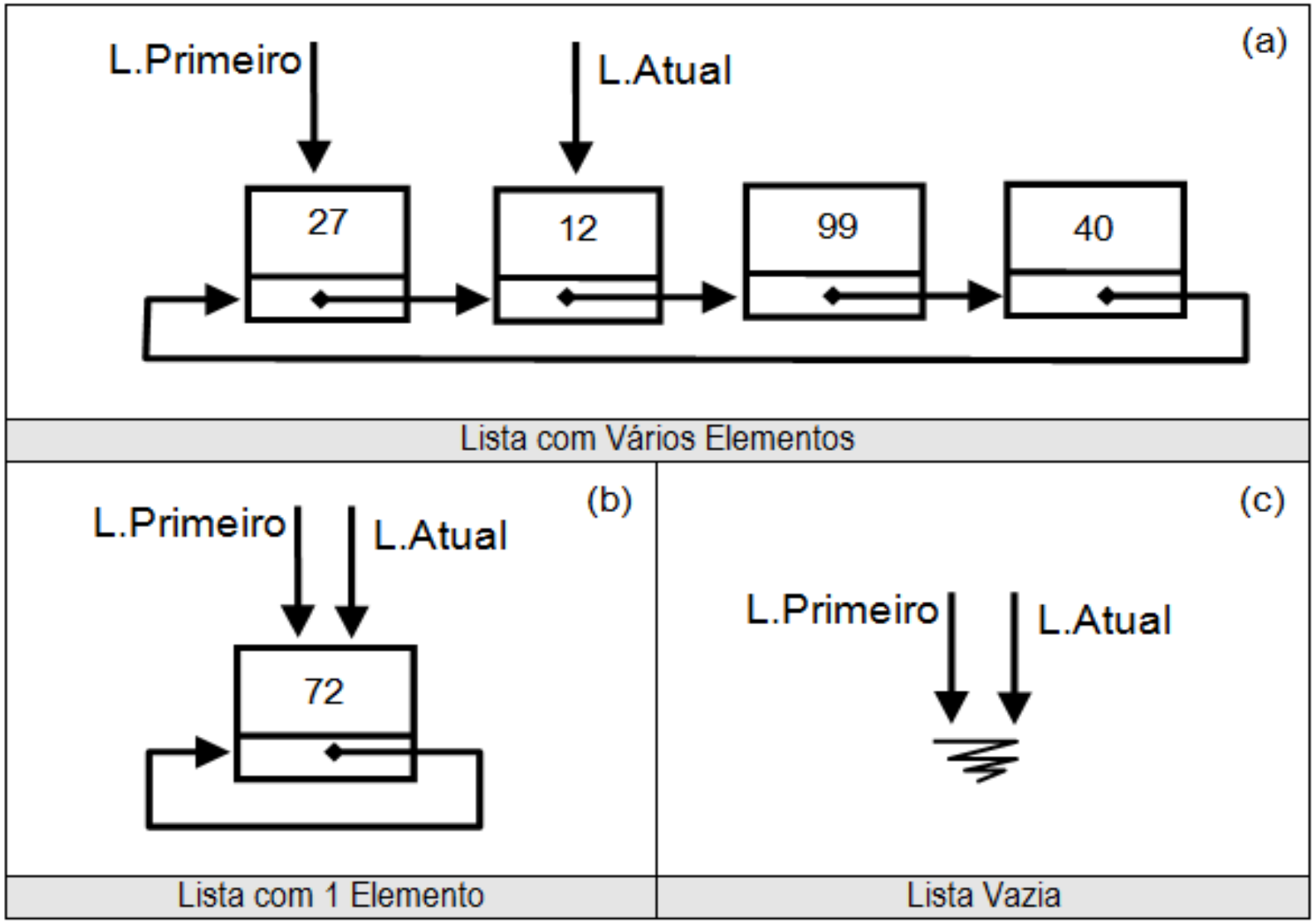
Exercício 6.14 Lista Cadastral Sem Elementos Repetidos Implementada como uma Lista Encadeada Ordenada Circular

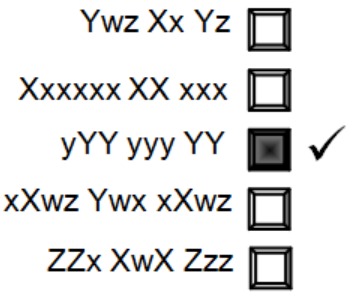


Exercício 6.16 Lista Cadastral Com Elementos Repetidos Implementada como uma Lista Encadeada Ordenada Circular



Exercício 6.18 Lista Cadastral Sem Elementos Repetidos, Implementada como uma Lista Encadeada Circular Não Ordenada





Exercícios

Exercício 6.15 Operação Destrói

Exercício 6.17 Operação RetiraTodosDeValorX de uma Lista Cadastral Com Elementos Repetidos

RetiraTodosComValorX (parâmetro por referência **L** do tipo Lista, parâmetro **X** do tipo Char, parâmetro por referência **Ok** do tipo Boolean);

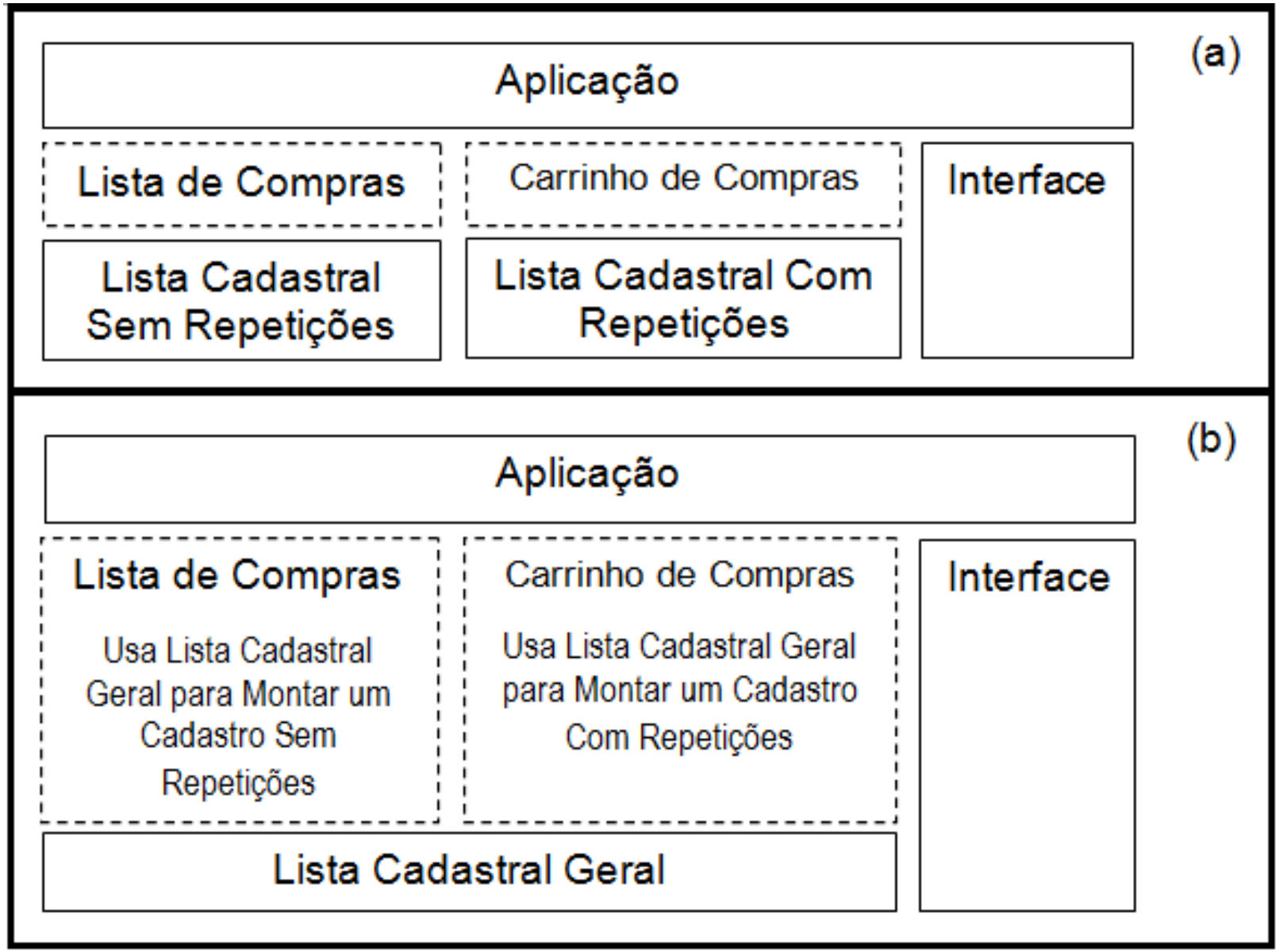
/* Retira todos os elementos de valor X que forem encontrados na Lista L. Se algum elemento de valor X for encontrado e removido, Ok retorna Verdadeiro; falso caso contrário */

Exercício 6.19 Implementar e Testar uma Lista Cadastral em uma Linguagem de Programação

Avanço de Projeto: Qual Tipo de Lista Usar?

- Ywz Xx Yz ☐
- Xxxxxx XX xxx ☐
- yYY yyy YY ☑ ✓
- xXwz Ywx xXwz ☐
- ZZx XwX Zzz ☐

Arquitetura



Exercício 6.21 Calcular o Número de Itens da Lista de Compras que Foram Efetivamente Comprados Pelo Jogador

Inteiro **ItensDaListaQueForamComprados** (parâmetros por referência **ListaDeCompras**, **CarrinhoDeCompras** do tipo Lista);

/* Recebe a Lista de Compras e o Carrinho de Compras e conta quantos itens da Lista de Compras aparecem pelo menos uma vez no Carrinho de Compras. Produz resultado do tipo inteiro */

Exercício 6.22 Calcular o Número de Itens Comprados Incorretamente

Inteiro **ItensCompradosIncorretamente** (parâmetros por referência **ListaDeCompras**, **CarrinhoDeCompras** do tipo Lista);

/* Recebe a Lista de Compras e o Carrinho de Compras e conta quantos itens do Carrinho de Compras não fazem parte da Lista de Compras. Retorna resultado do tipo inteiro */

Exercício 6.23 Calcular o Número de Itens Comprados em Excesso

Inteiro **ItensCompradosEmExcesso** (parâmetros por referência **ListaDeCompras**, **CarrinhoDeCompras** do tipo Lista);

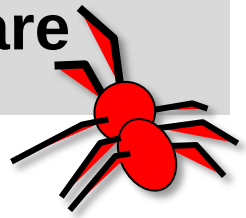
/* Recebe a Lista de Compras e o Carrinho de Compras e conta quantos itens foram comprados em Excesso, ou seja, número de itens da Lista de Compras que aparecem mais que uma vez no Carrinho de Compras. Produz resultado do tipo inteiro */

Avanço de Projeto

Exercício 6.24 Identificar Outras Aplicações de Listas Cadastrais

Exercício 6.25 Defina as Regras, Escolha um Nome e Inicie o Desenvolvimento do Seu *Jogo*

Exercício 6.20 Propor uma Arquitetura de Software



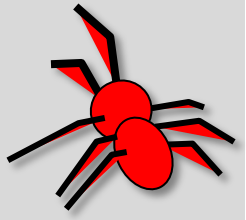
(Exercício 6.30) Técnica de Implementação

Qual combinação de técnicas parece ser mais adequada às características do jogo referente ao Desafio 3 que você desenvolverá: Alocação Sequencial e Estática ou Alocação Encadeada e Dinâmica?

Exercícios de Fixação

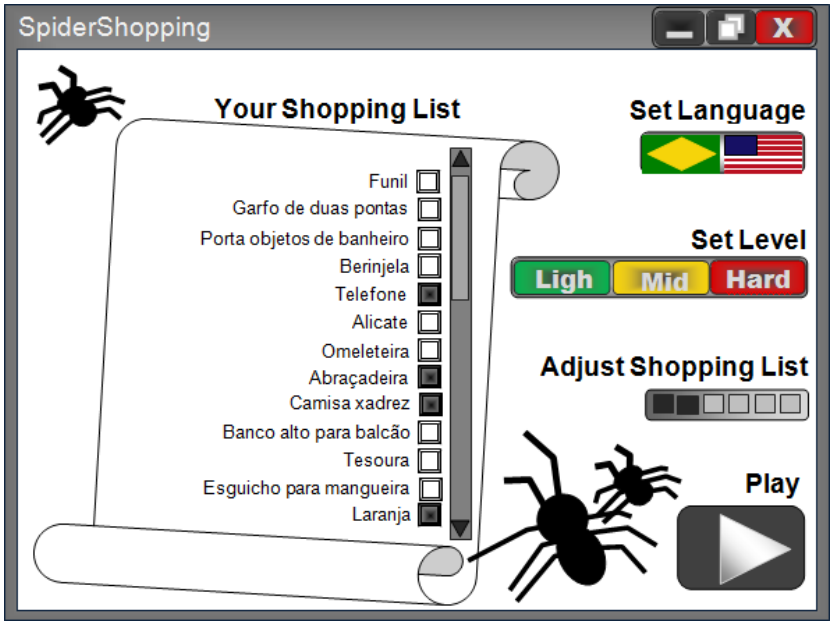
Exercício 6.26 Qual a diferença entre uma Lista Cadastral e uma Fila?

Exercício 6.27 Uma Lista Cadastral pode ser implementada como uma Lista Encadeada, mas pode ser implementada também com outra técnica. Cite um exemplo de como isso pode acontecer. Explique a diferença entre os termos Lista Cadastral e Lista Encadeada.



Exercício 6.29 Como seria a implementação de uma Lista Cadastral com Alocação Sequencial e Estática de Memória? Faça um diagrama e explique como seria o funcionamento.

Comece a Desenvolver Seu Jogo Agora!



Faça um jogo com a sua cara! Distribua para os seus amigos!

Estruturas de Dados com Jogos

Aprender a programar pode ser divertido!