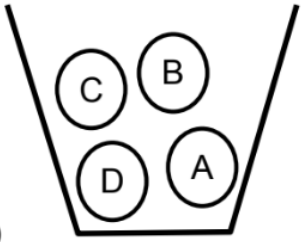
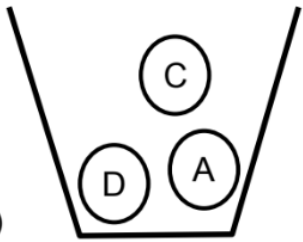
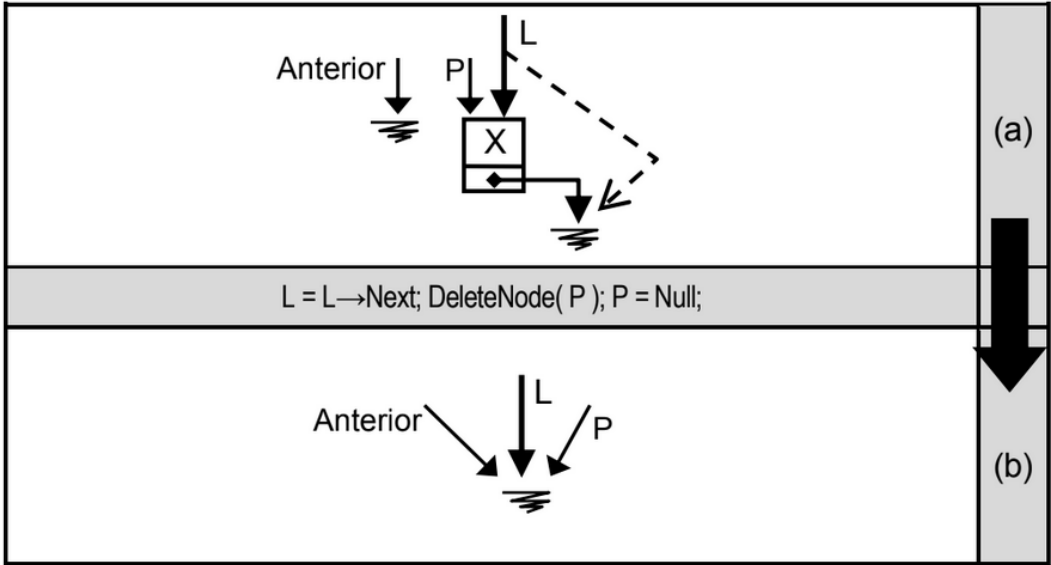
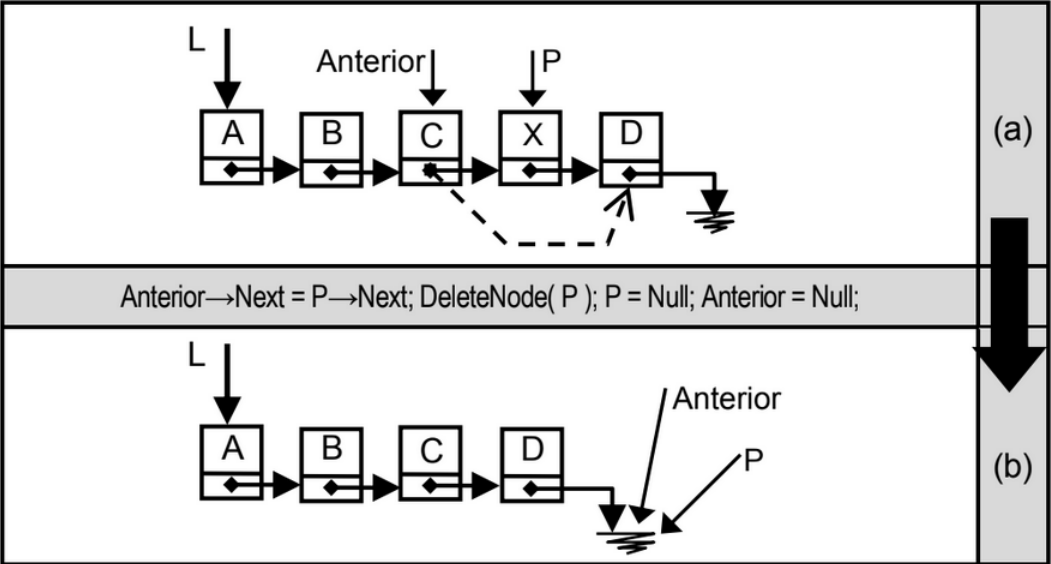


Diagramas da Lista Cadastral

Lista L	Operação	Resultado
<div>(a)</div> 	Insere(L, 'A', Ok)	Não insere, pois a Lista L já contém o valor 'A'.
	EstáNaLista(L, 'F')	Resultado Falso, pois o valor 'F' não está na Lista L
	EstáNaLista(L, 'B')	Resultado Verdadeiro, pois a Lista L contém elemento de valor 'B'.
	Retira(L, 'F', Ok)	Não retira, pois a Lista L não contém elemento de valor 'F'.
<div>(b)</div> 	Retira(L, 'B', Ok)	Retira o elemento de valor 'B' da Lista L (situação do Quadro 6.5a), que ficará agora apenas com os elementos A, C e D (situação do Quadro 6.5b).



Código do arquivo lista_cadastral.h separado em duas imagens

Imagem 1

```
include <assert.h>
#include <stdlib.h>

// Declaração do Nó
typedef struct Node {
    int info;
    struct Node *proximo;
} Node;

// Definição da Lista Cadastral
typedef struct Lista {
    Node *inicio, *fim;
    int tamanho;
} Lista;

// Função que retorna uma List avazia
Lista *cria() {
    Lista *l;

    l = (Lista *) malloc(sizeof(Lista));
    assert(l != NULL);

    l->inicio = NULL;
    l->fim = NULL;
    l->tamanho = 0;

    return l;
}

// Função que libera a memória alocada pela Lista atual
void libera(Lista *l) {
    if (l != NULL) {
        Node *p = l->inicio;

        while (p != NULL) {
            l->inicio = p->proximo;
            free(p);
            p = l->inicio;
        }

        free(l);
    }
}

// Função que retorna se o elemento recebido está na Lista ou não
int esta_na_lista(Lista *l, int x) {
    assert(l != NULL);

    Node *p = l->inicio;

    while (p != NULL) {
        if (p->info == x) {
            return 1;
        }

        p = p->proximo;
    }

    return 0;
}
```

Imagem 2

```
// Função que insere um elemento na Lista
void insere(Lista *l, int x) {
    assert(l != NULL);

    // Impede que tenha elementos repetidos
    if (esta_na_lista(l, x)) {
        return;
    }

    Node *p = (Node *) malloc(sizeof(Node));

    p->info = x;
    p->proximo = NULL;

    if (l->inicio == NULL) {
        l->inicio = p;
        l->fim = p;
    } else {
        l->fim->proximo = p;
    }

    l->fim = p;

    l->tamanho++;
}

// Função que retorna se o elemento recebido foi retirado com sucesso ou não
int retira(Lista *l, int x) {
    assert(l != NULL);

    Node *anterior = NULL;
    Node *p = l->inicio;

    while (p != NULL) {
        if (p->info == x) {
            if (anterior == NULL) {
                l->inicio = l->inicio->proximo;
                free(p);
            } else if (p == l->fim) {
                l->fim = anterior;
                l->fim->proximo = NULL;
                free(p);
            } else {
                anterior->proximo = p->proximo;
                free(p);
            }

            l->tamanho--;

            return 1;
        } else {
            anterior = p;
            p = p->proximo;
        }
    }

    return 0;
}
```

Código do arquivo usa_lista_cadastral.c

```
#include "lista_cadastral.h"

#include <stdio.h>

// Função que imprime os elementos da Lista
void imprime(Lista *l) {
    assert(l != NULL);

    Node *p = l->inicio;

    printf("\nImprimindo a Lista:\n");

    while (p != NULL) {
        printf("%d ", p->info);
        p = p->proximo;
    }

    printf("\n");
}

// Função que retorna o tamanho atual da Lista
int tamanho_lista(Lista *l) {
    assert(l != NULL);

    int tamanho = 0;
    Node *p = l->inicio;

    while (p != NULL) {
        p = p->proximo;
        tamanho++;
    }

    return tamanho;
}

// Função main que usa as funções da Lista
int main() {
    Lista *l = cria();

    printf("Adicionando os elementos 10, 15, 82, 98, 3 à Lista\n");
    insere(l, 10);
    insere(l, 15);
    insere(l, 82);
    insere(l, 98);
    insere(l, 3);
    imprime(l);
    printf("Tamanho atual da lista: %d\n", tamanho_lista(l));

    printf("\nRemovendo os elementos 82 e 98 da Lista\n");
    retira(l, 82);
    retira(l, 98);
    imprime(l);
    printf("Tamanho atual da lista: %d\n", tamanho_lista(l));

    libera(l);
}
```

Imagem da execução do programa

```
ranzani in AED1/Frequências/F5 on ʘ main [!?]
→ gcc usa_lista_cadastral.c -o usa_lista_cadastral
ranzani in AED1/Frequências/F5 on ʘ main [!?]
→ ./usa_lista_cadastral
Adicionando os elementos 10, 15, 82, 98, 3 à Lista

Imprimindo a Lista:
10 15 82 98 3
Tamanho atual da lista: 5

Removendo os elementos 82 e 98 da Lista

Imprimindo a Lista:
10 15 3
Tamanho atual da lista: 3
```