DCC013 - Estrutura de Dados / DCC107 - Laboratório de Programação II

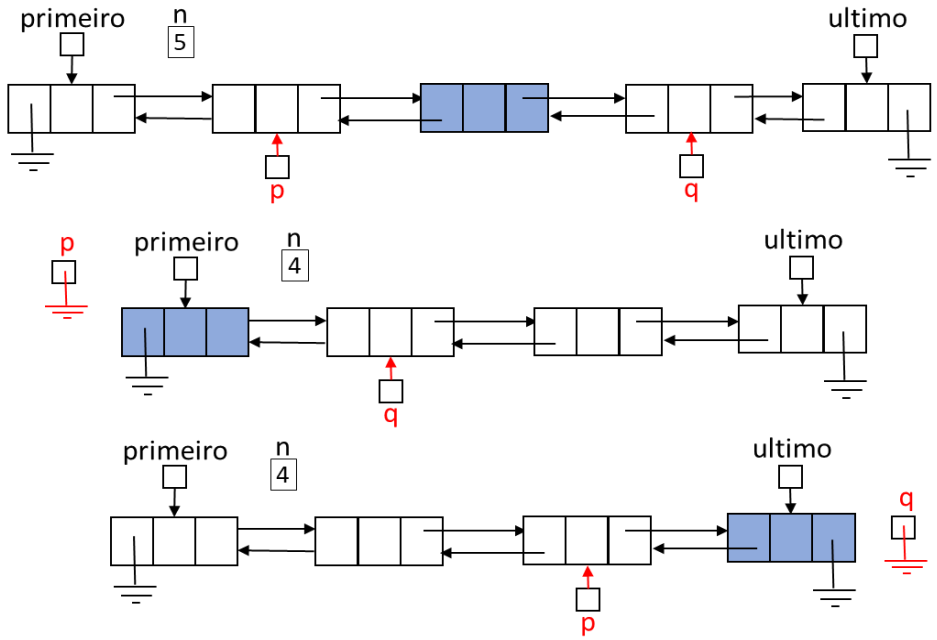
Lista duplamente encadeada

Você pode utilizar qualquer ambiente de programação para desenvolver sua atividade. Ao final, **copie e cole o seu código-fonte com a resposta aqui mesmo neste documento**, dentro dos espaços indicados para isso e **preservando a identação do código**. **Depois que terminar sua avaliação, não se esqueça de entregar sua atividade!** Fique atento ao relógio, pois as atividades entregues com atraso não serão aceitas.

Para resolver esta atividade, **clique aqui para baixar** o projeto que contém a implementação do TAD `ListaDupla` de **números inteiros** (usando uma **lista duplamente encadeada - LDE**). Na sua solução para a questão abaixo, você pode utilizar/chamar qualquer uma das operações que estejam disponíveis no projeto. Outras operações que você eventualmente deseja utilizar devem ser copiadas para sua resposta neste documento.

A) Implementar a operação `void ListaDupla::removeEntre(NoDuplo* p, NoDuplo* q)` para, dados dois ponteiros `p` e `q` para nós da LDE, remover:
i) o nó entre os nós `p` e `q`, se eles têm um único nó entre eles.
ii) o primeiro nó da LDE, se `p` for nulo e `q` apontar para o segundo nó da lista;
iii) o último nó da LDE, se `p` apontar para o penúltimo e `q` for nulo.
Mesmo quando ambos ponteiros `p` e `q` não forem nulos, `p` pode apontar para qualquer nó que antecede ao nó apontado por `q` e não é garantido que exista apenas um nó entre eles. Não será removido nenhum nó se ambos forem nulos ou se não houver um único nó entre eles. Considere que a lista tem 3 ou mais nós.

Exemplos: O nó “azul” deve ser removido nos casos das 3 figuras a seguir.



B) usando a operação do item (A), implementar a operação **`void`** `ListaDupla::remove1Ocorrencia(`**`int`** `val)` para remover a primeira ocorrência de um nó da LDE cujo valor é `val`.

Inserir seu código aqui:

```
void ListaDupla::removeK(int k)
{
    if(k < 0 || k >= n)
        cout << "Indice invalido" << endl;
    else if(k == 0)
        removeInicio();
    else if(k == n-1)
        removeFinal();
    else
    {
        NoDuplo *p = primeiro;
        for(int i = 0; i < k; i++)
            p = p->getProx();
        NoDuplo *ant = p->getAnt();
        NoDuplo *prox = p->getProx();
        ant->setProx(prox);
        prox->setAnt(ant);
        delete p;
        n--;
    }
}

void ListaDupla::removeEntre(NoDuplo *p, NoDuplo *q)
{
    NoDuplo *aux = primeiro;
    int indice = 0;

     if(p == NULL && q == NULL) //Se ambos forem NULL, não retiramos
nenhum nó
    {
        cout << "Nenhum nó foi encontrado" << endl;
        return;
    }

    if(p == NULL && q == primeiro->getProx()) // Se p for NULL e q for
o segundo, retiramos o primeiro
    {
        removeInicio();
```

```
        return;
    }

    if(p == ultimo->getAnt() && q == NULL) // Se p for o penultimo e q
for NULL, retiramos o ultimo
    {
        removeFinal();
        return;
    }

    if(p->getProx() == q->getAnt()) // Se houver apenas um nó entre p e
q
    {
        while(aux != p)
        {
            aux = aux->getProx();
            indice++;
        }
        removeK(indice + 1); // Remove o próximo do indice p
    }

}

void ListaDupla::remove1Ocorrencia(int val)
{
    NoDuplo *p = primeiro;
    NoDuplo *ant = NULL;
    NoDuplo *prox = NULL;

    while(p != NULL)
    {
        if(p->getInfo() == val)
        {
            ant = p->getAnt();
            prox = p->getProx();
               break; // Para a execução do while para pegar sempre a
primeira ocorerencia
        }
        p = p->getProx();
    }

    removeEntre(ant, prox); // Chamando a operação anterior
}
```