**LISTA DE EXERCÍCIOS PREPARATÓRIOS PARA A PROVA 2**

**QUESTÕES TEÓRICAS 1. Definição e Vantagens**

**a) Explique o que é uma lista encadeada e compare-a com um vetor (array) em termos de inserção/remoção e alocação de memória.**

**Uma lista encadeada é uma estrutura de dados composta por uma sequência de elementos chamados nós (nodes), onde cada nó contém um dado e um ponteiro para o próximo nó da lista.**

**Comparação com vetores (arrays):  
Em listas encadeadas, inserir ou remover elementos em qualquer posição (especialmente no início) é mais eficiente, pois envolve apenas a manipulação de ponteiros.**

**Em vetores, inserir ou remover no meio ou início exige o deslocamento dos elementos seguintes, o que torna essas operações mais custosas em tempo.**

**Alocação de Memória:  
Listas encadeadas usam alocação dinâmica (heap), ou seja, só ocupam memória conforme necessário.**

**Vetores usam alocação estática ou semidireta, exigindo definir o tamanho previamente (ou realocação em vetores dinâmicos).**

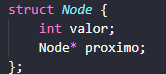
**b) Por que listas encadeadas são consideradas estruturas dinâmicas?**

**Seus elementos são alocados dinamicamente na heap, em tempo de execução.**

**Não têm tamanho fixo, podendo crescer ou diminuir conforme a necessidade, sem desperdiçar memória.**

**2. Nós e Ponteiros**

**a) Descreva a estrutura de um nó (Node) em uma lista encadeada simples.**

****

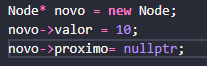
**b) Qual a importância do ponteiro nullptr em listas encadeadas?**

**O ponteiro nullptr em C++ é fundamental para indicar o fim da lista onde o último nó aponta para nullptr. Um ponteiro não está apontando para nenhum nó válido, evitando acessos inválidos à memória.**

**3. Operações Básicas**

**a) Descreva o passo a passo para inserir um nó no início da lista.**

1. **Criar um novo nó (novo).**
2. **Atribuir o valor desejado a novo->dado.**
3. **Fazer novo->proximo apontar para o atual início da lista (inicio).**
4. **Atualizar o ponteiro da lista para que inicio = novo.**

****

**b) Quais verificações são necessárias antes de remover um nó de uma lista?**

**Antes de remover um nó, é importante verificar se a lista está vazia, ou seja, se o ponteiro de início é nullptr. Ter acesso ao nó anterior ao que será removido (no caso de remoção no meio ou fim). Garantir que o ponteiro de ligação da lista seja corretamente ajustado. Liberar a memória alocada com delete para evitar vazamentos.**

**4. Memória a) Por que os nós de uma lista encadeada são alocados na heap?**

**Os nós são alocados na heap porque permite alocação dinâmica em tempo de execução, evita a limitação de tamanho fixo, permite criar estruturas que mudam de tamanho durante a execução do programa.**

**b) O que acontece se não liberarmos a memória dos nós removidos (delete)?**

**Se não liberarmos a memória com delete ao remover nós ocorrerá um vazamento de memória e a memória usada pelos nós removidos nunca será liberada, mesmo sem uso.Com o tempo, isso pode consumir toda a memória disponível, prejudicando o desempenho ou travando o programa.**