Lista de Exercícios 10 - Lista Simplesmente Encadeada

1. Construa uma função que recebe como parâmetros uma Lista L de valores inteiros e um valor **X**. A função deve retirar os primeiros **X** valores da lista L, inserindo-os no fim de L. Use as funções de inserção e remoção separadas.

   Exemplo:

   L: [3,5,8,9,12,11,7,10]

   x: 4

   L após a função: [12,11,7,10,3,5,8,9]

2. Um grupo de pesquisa em segurança da informação está estudando a frequência de uso de senhas semelhantes pelos usuários de um determinado sistema. Assim, foi criada uma lista contendo as senhas dos usuários. Uma das análises a ser realizada é a verificação de senhas não seguras. Uma senha pode ser considerada "não segura" se ela possuir uma quantidade muito pequena de caracteres ou for utilizada por uma quantidade muito grande de usuários. Para classificar as senhas não seguras, você deve criar uma lista simplesmente encadeada contendo todas as senhas com menos de 4 caracteres ou cuja frequência de utilização é maior que 5 (ou seja, senhas que aparecem mais que cinco vezes na lista). Crie a função naoSeguras que recebe uma lista de senhas e retorna outra lista contendo todas as senhas não seguras existentes.

3. Faça uma função que verifica se uma lista encadeada está ordenada:

   ```
   int verificaOrdem(No** lista);
   ```

   Retorno da função:
   0 – Ordem crescente
   1 – Ordem decrescente;
   2 – Não está ordenada.

   Observação: É proibido o uso de listas auxiliares, percorrer os nós da lista.

4. Construa uma função para ordenar uma lista simplesmente encadeada.
   ```
   void ordenarL(No** lista);
   ```

   Observação:

   - É proibido o uso de listas auxiliares, percorrer os nós da lista.
   - A lista não pode estar vazia.

   Dicas:

   - Percorrer os nós da lista, verificando o dado do nó atual e do próximo (caso ele exista), trocando os valores dos dados dos nós caso necessário.
   - Uma lista está completamente ordenada quando não há mais trocas de dados entre os nós, ou seja, ao percorrer uma lista, se nenhuma troca foi realizada, a lista está ordenada.

5. Implemente uma função e inserir de forma ordenada em uma lista encadeada.
   `void insereOrdenado(No** lista, int valor);`

   Observação:

   - É proibido o uso de listas auxiliares, percorrer os nós da lista e ajustar os ponteiros;
   - Considere que a lista está inicialmente vazia.

   Dicas:

   - Use como base a função removeValor, ajustando os ponteiros dos nós e levando em consideração se o elemento será inserido no início da lista, no meio ou no final.

6. Crie uma função parar inserir em uma lista encadeada com base em uma posição/índice.

```
bool inserePosicao(No** lista,  int pos, int valor);
L1: 2 5 3 9 9 8 7 6 2 1

inserePosicao(L1, 0, 100); //returna true
L1: 100 2 5 3 9 9 8 7 6 2 1

inserePosicao(L1, 3, 200); //returna true
L1: 100 2 5 200 3 9 9 8 7 6 2 1

inserePosicao(L1, 11, 300); //returna true
L1: 100 2 5 200 3 9 9 8 7 6 2 300 1

inserePosicao(L1, 13, 1000); //returna false, pois pos é inválido
L1: 100 2 5 200 3 9 9 8 7 6 2 300 1

inserePosicao(L1, -1, 1000); //returna false, pois pos é inválido
L1: 100 2 5 200 3 9 9 8 7 6 2 300 1
```

7. Dados duas listas L1 e L2, crie uma função para computar a lista L3 que é a intersecção de L1 e L2.
   Exemplo:

```
L1: 2 5 3 9 9 8 7 6 2 1
L2: 20 30 100 6 1 5 9
L3: 5 9 6 1
```