Lista de Exercícios 8 - Lista Simplesmente Encadeada

1. Crie um menu para executar as operações sobre uma lista que armazena valores do tipo string.
2. Crie as seguintes funções:

```
//retorna o total de elementos da lista
int totalL(No **lista)

//verifica se duas listas são iguais
bool igualL(No **lista1, No **lista2)

//Insere um valor no final da lista
void insereFinalL(No **lista, int valor)

//retorna o valor do último elemento da lista SEM removê-lo
DadoNoLista leFinalL(No **lista)

//retorna o valor do último elemento da lista e REMOVE o elemento
DadoNoLista removeFinalL(No **lista)

//retorna o valor do dado armazenado em um nó da lista, com base na posição passada por parâmetro
//se posição=0, a função deve retornar o valor do primeiro nó da lista
//se posição=1, a função deve retornar o valor do segundo nó da lista
DadoNoLista lePosicao(No **lista, int posição)

//União de duas listas
void uniao(No **lista1, No **lista2, No **listaSaida)
```

3. Considere uma lista encadeada L1 representando uma sequência de caracteres. Construa uma função para imprimir a sequência de caracteres da lista L1 na ordem inversa.

   Exemplo: para a lista L1={A,E,I,O,U}, a função deve imprimir "UOIEA".

   Dicas: faça uso de uma lista auxiliar e as funções *removeFinalL* e *insereFinalL* do exercício anterior.

4. Faça uma função que verifica se uma Lista L1 está ordenada (em ordem crescente ou decrescente).

5. Construa uma função que recebe como parâmetros uma Lista L1 de valores inteiros e um valor **X**. A função deve retirar os primeiros **X** valores da lista L1, inserindo-os no fim de L1. Use as funções de inserção e remoção separadas.

   Exemplo:

   L1: [3,5,8,9,12,11,7,10]

   x: 4

   L1 após a função: [12,11,7,10,3,5,8,9]

6. 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.

7. Implemente duas funções para a ordenação de uma lista simplesmente encadeada:
   - Primeira função: percorrer os nós da lista, verificando o dado do nó atual e do próximo, trocando os valores dos dados dos nós caso necessário.
   - Segunda função: retirar os elementos de uma lista L e inserir de forma ordenada em uma lista auxiliar LAUX. Em seguida, passe os elementos da lista LAUX para L.

8. 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
```

9. 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
```