Usando somente as funções do TAD Lista, resolva as questões abaixo:

1) Verificar se duas listas são iguais. Duas listas são iguais se ambas as estruturas têm o mesmo número de elementos, e estes são iguais um a um. Em particular, duas listas vazias são iguais.

int iguaisListas(Lista lst1, Lista lst2){...}

2) Verificar se duas listas são semelhantes. Duas listas são semelhantes se têm os mesmos elementos mesmo em ordem diferente. Se existe um elemento repetido na lista lst1, o mesmo deve aparecer repetido na lista lst2.

int semelhantesListas(Lista lst1, Lista lst2){...}

3) Verificar se a lista lst2 é uma sublista de lst1. Neste caso, todos os elementos de lst2 estão em lst1, consecutivos e na mesma ordem. Em particular, uma lista vazia é uma sublista de qualquer lista, e uma lista é sublista de si mesma.

int subLista(Lista lst1, Lista lst2){...}

4) Verificar se uma lista lst2 está contida numa lista lst1. Para tal, todos os elementos de lst2 devem estar em lst1, mesmo em ordem diferente.

int contidaLista(Lista lst1, Lista lst2){...}

5) Verificar se uma lista lst está ordenada: todos elementos respeitam a relação de ordem <=.

int ordenadaLista(Lista lst1){...}

6) Adiciona o elemento elem no final de lst:

void adicLista(Lista lst, TipoL elem){...}

7) Substitue o conteúdo atual do iterador pelo elemento elem:

void substitueLista(Lista lst, TipoL elem){...}

8) Indica se o elemento elem aparece na lista:

int estaNaLista(Lista lst, TipoL elem){...}

9) Exibe todos os elementos da lista, utilizando a operação posLista para avançar:

void exibeLista(Lista lst){...}

10) Coloca o iterador na posição anterior à atual:

void antLista(Lista lst){...}

11) Retorna a posição do iterador na lista:

Int posIteradorLista(Lista lst){...}

12) Deixar na lista somente uma ocorrência de cada um dos elementos:

void simplificarLista(Lista lst){...}

13) Retorna o número total de elementos diferentes em lst:

int numDiferentesLista(Lista lst){...}

14) Computa o número de vezes que o elemento elem aparece na lista:

int numOcorrenciasLista(Lista lst, TipoL elem){...}

15) Retorna o elemento que aparece mais vezes na lista não vazia lst:

TipoL maxOcorrenciaLista(Lista lst, TipoL elem){...}

16) Retorna a posição da última ocorrência do elemento elem. Se não ocorre, retorna zero:

int ultOcorrenciaLista(Lista lst, TipoL elem){...}

17) Elimina da lista lst todos os elementos compreendidos entre a posição p1 e p2, inclusive:

Void eliminarLista(Lista lst, int p1, int p2){...}

18) Ordena em ordem crescente a lista lst:

void ordenarLista(Lista lst){...}

19) Elimina da lista lst1 todos os elementos que aparecem na lista lst2:

void diferencaLista(Lista lst1, Lista lst2){...}