Lista Encadeada - Exercícios

Estrutura de Dados e Armazenamento

1) Criar a classe ListaLigadaOrdenada, herdeira de ListaLigada, representando uma lista encadeada que tem seus dados em ordem crescente, a partir do nó apontado pelo head.

a) Reescrever o método insereNode, para inserir o novo nó de forma a manter a lista em ordem crescente.
Dica: para implementar esse método, é bom ter 2 variáveis do tipo Node para percorrer a lista: ant (endereço do nó anterior) e atual (endereço do nó atual). E também uma variável boolean inseriu, que indica se já inseriu o nó ou não.

b) Reescrever o método buscaNode, para buscar o elemento na lista ordenada (qdo encontrar um valor maior do que o procurado, já pode parar a busca)

c) Reescrever o método removeNode

d) Testar a classe ListaLigadaOrdenada

2) Criar na classe ListaLigada as versões recursivas do método exibe, buscaNode, removeNode, tamanho. Nesse caso, pode criar um método público que recebe o mesmo argumento que o método não recursivo, e um método privado que recebe mais um argumento que é endereço do primeiro nó da lista (head.next). Teste os métodos recursivos!

3) Na classe ListaLigada, acrescentar os métodos e testá-los:

a) inserirAposPrimeiroImpar(int valor) – void

Insere o valor após o primeiro valor ímpar que encontrar na lista, quando percorre a partir do head. Se não houver nenhum valor ímpar na lista, insere no final de todos.

Por exemplo:

Suponha que a lista seja assim:

head → 2

se chamar o método inserirAposPrimeiroImpar(7), a lista ficaria assim:

head → 2 → 7

Suponha que a lista seja assim:

head → 2 → 7 → 10 → 35 → 9

se chamar o método inserirAposPrimeiroImpar(100), a lista ficaria assim:

head → 2 → 7 → 100 → 10 → 35 → 9

b) getElemento(int indice) - int

Retorna o info do nó que está no índice passado no argumento, considerando-se que o 1º nó a partir do head tem o índice zero e assim sucessivamente.

Por exemplo:

Suponha que a lista seja assim:

head → 2 → 7 → 10 → 35 → 9

A chamada do método getElemento(0) retorna 2

A chamado do método getElemento(3) retorna 35

c) removeOcorrencias(int valor) - boolean.
Remove todas as ocorrências do valor na lista. Se encontrar ocorrências, remove todas e retorna true. Se não encontrar ocorrências, retorna false.

Por exemplo:

Suponha que a lista seja assim:

head → 2 → 7 → 2 → 35 → 2

A chamada do método removeOcorrencias(70) retorna false.

A chamada do método removeOcorrencias(2) retorna true e a lista ficará assim:

head → 7 → 35

d) arrayToList(int[ ] vetor) - void.

Se a lista não estiver vazia, exibe mensagem "Operação inválida" e não faz nada.

Se a lista estiver vazia, copie o vetor para a lista, de forma que fique na mesma ordem do vetor, a partir do head.

Por exemplo:

A chamada arrayToList(vetor), sendo vetor = { 30, 50, 6, 90, 110}, produzirá a lista:

head → 30 → 50 → 6 → 90 → 110