Universidade Federal da Paraíba Centro de Informática Departamento de Informática	Estrutura de Dados Semestre: 2016.2 Professor: Tiago Maritan

Lista de Exercícios

Não precisa entregar. É apenas para auxiliar nos estudos

1) Compare as listas implementadas com representação sequencial com as listas implementadas com representação dinâmica e identifique quais são as vantagens e desvantagens de cada uma das abordagens.

2) Considere a implementação das estruturas de dados LISTA SEQUENCIAL e LISTA SIMPLESMENTE ENCADEADA vistas em sala de aula. Acrescente nelas as seguintes operações/funções:

a) Uma função para exibir a lista. Esta função deve exibir o tamanho da lista e todos os seus elementos;
b) Uma função que divida uma lista em duas outras listas, uma com elementos de pares e a outra com elementos de chaves ímpares;
c) Uma função que concatena duas listas, gerando uma terceira lista;
d) Uma função que modifique um elemento de uma lista, sendo dado a sua posição e o novo valor;
e) Uma função que remova um elemento da lista dado o seu valor;

3) Implemente a estrutura de dados LISTA DUPLAMENTE ENCADEADA em um módulo (biblioteca) chamada LISTA_DUPENC (usando o inteiro como tipo base), contendo as operações básicas de lista encadeada vistas em sala de aula.

4) Reescreva a função de inserção de elementos em listas duplamente encadeadas, considerando o seguinte:

- Caso o elemento seja inserido numa posição até o meio da lista, essa inserção deve ser feita percorrendo os elementos do início da lista até a posição desejada;
- Caso o elemento seja inserido após o meio da lista, essa inserção deve ser feita percorrendo os elementos do fim da lista até a posição desejada;

5) Faça um programa que teste todas as funcionalidades/operações sobre listas implementadas nas questões 2, 3 e 4.

6) Implemente a estrutura LISTA SIMPLESMENTE ENCADEADA, , contendo todas as operações básicas de lista encadeada vistas em sala de aula, numa linguagem de programação orientada a objetos (ex: C++, Java) de sua preferência.

7) Considerando os módulos PILHA SEQUENCIAL E PILHA ENCADEADA, criados em sala de aula, acrescente neles as seguintes operações/funcionalidades:

f) Uma função para exibir a pilha. Esta função deve exibir o tamanho da pilha e todos os seus elementos;
g) Uma função para inverter uma pilha.
h) Uma função que elimina todas as ocorrências de um determinado elemento em uma pilha, retornando o número de elementos eliminados.
i) Uma função que recebe duas pilhas e verifica se elas são iguais.
j) Uma função para remover todos os elementos de uma pilha, ou seja, destruir a pilha.

8) Crie um módulo main que teste todas as funcionalidades/operações sobre pilhas implementadas na questão 1.

9) Implemente um programa que analise uma expressão matemática e identifique se os elementos separadores de abertura '[', '{' e '(' são encerrados de forma correta com os elementos separadores de encerramento ')', '}' e ']'.

Por simplicidade, o programa não verifica a ordem de emprego desses elementos de abertura. Ou seja, expressões tais como **2 * (3 – [4+ { 2 + 3 }])** e **2 * { 3 – (4 + [2 + 3]) }** são consideradas válidas.

Por outro lado, expressões tais como **2 * (3 – [4+5)]** são consideradas inválidas, pois o último elemento aberto '[', posicionado antes do número 4, está sendo encerrado com o ')', posicionado após o número 5.

Dica: Utilize uma estrutura de dados pilha pra isso. Por exemplo, toda vez que um '(' for encontrado na expressão, você pode empilhar esse elemento em uma pilha. Por outro lado, quando um ')' for encontrado, você pode desempilhar.

10) Escreva um programa que use uma pilha para determinar se uma string é um palíndromo (isto é, tem a mesma leitura no sentido normal e no sentido inverso). O programa deve ignorar espaços em branco, pontuação e caracteres especiais.

11) Considerando as estruturas de dados FILA SEQUENCIAL e FILA ENCADEADA, criadas em sala de aula, acrescente nele as seguintes funcionalidades:

- Uma função para exibir a fila. Esta função deve exibir o tamanho da fila e os seus elementos;
- Uma função para verificar se um determinado elemento está na fila. Em caso positivo, a função deve retornar a posição do elemento na fila. Considere a posição da frente como posição 1;
- Uma função que receba duas filas e uma fila vazia. Esta terceira fila deve ser preenchida com os valores das duas filas de forma intercalada. O preenchimento só deve acontecer se as duas filas forem do mesmo tamanho.
- Uma função que inverte a ordem dos elementos da fila. A função deve receber uma fila preenchida e uma fila vazia, e deve preencher a fila vazia com os elementos da fila preenchida em ordem invertida.

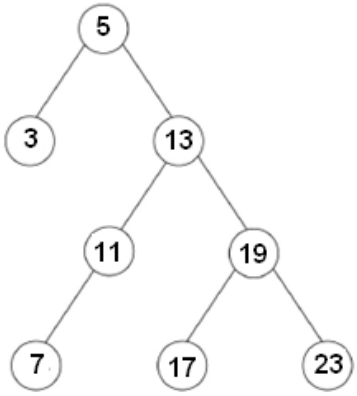
12) Implemente a estrutura de dados ÁRVORE BINÁRIA em uma biblioteca chamada ARV_BIN (usando o inteiro como tipo base), contendo as operações básicas de árvores binárias (vistas em sala de aula).

Acrescente a esta biblioteca as seguintes funcionalidades:

- Uma função para retornar o filho esquerdo de um nó;
- Uma função para retornar o filho direito de um nó;
- Uma função para exibir os descendentes de um nó;
- Uma função que receba um ponteiro para o nó raiz de uma árvore binária e retorna o número de nós dessa árvore;

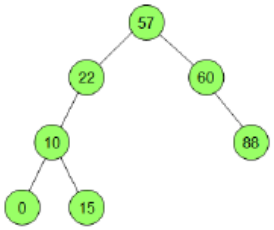
13) Implemente a estrutura de dados ÁRVORE BINÁRIA DE PESQUISA em uma biblioteca chamada ARV_BIN_P (usando o inteiro como tipo base), contendo as operações básicas de árvore binária de pesquisa (vistas em sala de aula).

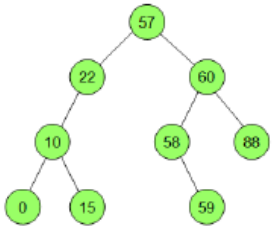
14) Considere a árvore binária abaixo.

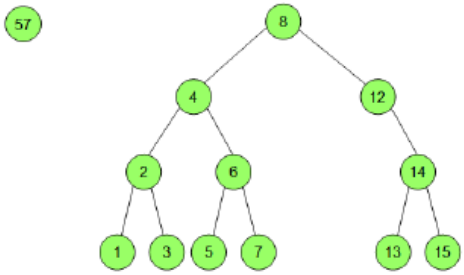


Qual é a sequencia de nós exibida quando aplicado o caminhamento em árvore pré-ordem, in-ordem e pós-ordem sob esta árvore binária?

15) Para cada um das árvores binária abaixo, indique se ela é ou não uma árvore AVL. Justifique.





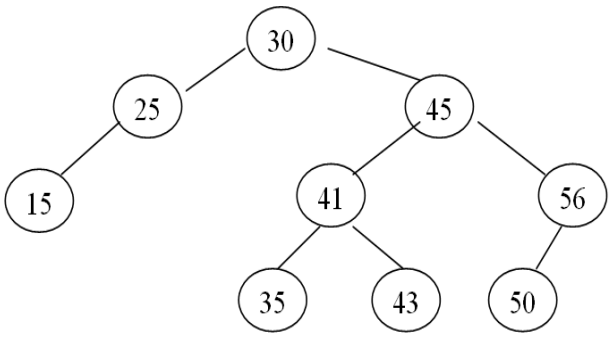


16) Partindo de uma árvore AVL vazia, realize a inserção da seguinte sequência de chaves:

99, 44, 71, 80, 74, 63, 59, 120, 98, 150.

Redesenhe a árvore a cada inserção. Indique para cada rotação feita, o nome da rotação e o nó desregulado.

17) Considere a árvore AVL a seguir:



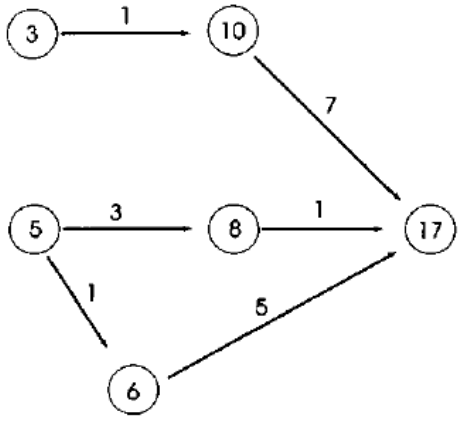
Realize, na árvore acima, as inserções das chaves 49, 60, 65. Mostre todas as rotações e o formato da árvore após cada operação, caso seja necessário.

18) Partindo de uma árvore AVL vazia, realize a inserção da seguinte sequência de chaves, redesenhando a árvore a cada inserção. Indique para cada rotação feita, o nome da rotação e o nó desregulado.

a) 41, 38, 31, 12, 19, 8, 27, 49
b) 10, 20, 30, 40, 35
c) 50, 30, 20, 70, 40, 35, 37, 38, 10, 32, 45, 42, 25, 47, 36.
d) 100, 80, 60, 40, 20, 70, 30, 50, 35, 45, 55, 75, 65, 73, 77

19) Escreva um algoritmo (pode ser em pseudolinguagem) que verifica se uma dada árvore binária é do tipo AVL. Suponha já existente uma função p.altura() que retorna a altura de uma árvore binária referenciada por **p**.

20) Considere o grafo abaixo:



a) Represente o grafo abaixo usando cada uma das formas de implementações de grafos abaixo (vistas em sala de aula):
   - Lista de Adjacência
   - Matriz de Adjacência
   - Estrutura Aresta
b) A sequencia de vértices { 10, 17, 6, 5, 8, 17} forma um clico? Explique sua resposta.
c) Defina dois possíveis caminhos entre o vértice 6 e o vértice 10.
d) Considerando o valor de cada aresta a distância entre os vértices. Qual o menor caminho possível para a questão da letra c?

e) Descreva a sequencia de visita dos vértice do grafo acima usando Busca em largura e Busca em profundidade.

Bons estudos!!!