Universidade Federal de Ouro Preto
Instituto de Ciências Exatas e Aplicadas
Professor Matheus Guedes
CEA488 – Algoritmos e Estruturas de Dados I

2ª Lista de Exercícios (Valor: 5 pontos)

Instruções:
i) A lista de exercícios deverá ser feita individualmente.
ii) Somente serão aceitas as listas entregues até o dia da Prova 2, antes da realização da mesma.
iii) A lista poderá ser entregue via e-mail (matheusgueedes91@gmail.com) ou impressa.
v) Não é necessário interação com o usuário, quando isto não for explicitado!

Exercícios

1. Faça os quatro exercícios propostos na **Aula(Revisão de Programação).pdf**, nos slides 29, 30, 31 e 32, respectivamente.
2. Implemente a TAD Matriz, sugerida na página 95 da **Apostila de Estrutura de Dados** (apostila disponível no site), dos professores Waldemar Celes e José Lucas Rangel.
3. Considerando o uso de listas encadeadas, implemente as seguintes funcionalidades, considerando a assinatura (cabeçalho) descrita para cada função:
   A) **Lista* insereFinal (Lista *l, int elemento);** // Insere o elemento no final da lista.
   B) **Lista* inserePosicao (Lista *l, int elemento, int p);** // Insere o elemento na posição *p* da lista.
   C) **Lista* insereOrdenado (Lista *l, int elemento);** // Insere o elemento na lista de forma ordenada => ordenação decrescente.
   D) **Lista* concatena (Lista *l1, Lista *l2);** // Concatena *l2* ao final de *l1*.
   E) **Lista* copia (Lista *l1);** // Retorna uma cópia de *l1*.
   F) **Lista* intersecao (Lista *l1, Lista *l2);** // Retorna uma lista, com a interseção de *l1* e *l2*.
   G) **Lista* uniao (Lista *l1, Lista *l2);** // Retorna uma lista, com a união de *l1* e *l2*, sem elementos repetidos.
   H) **Lista* diferença (Lista *l1, Lista *l2);** // Retorna uma lista, com a diferença de *l1* e *l2* (*l1 – l2*).
   I) **Lista* insereNRepetido (Lista *l1, int elemento);** // Insere elemento no início (ou final) da lista, desde que tal elemento não esteja na lista.
   J) **int igual (Lista *l1, Lista *l2);** // Retorna 1 se as listas são iguais ou retorna 0, caso contrário.

K) **int 50PorcentoIgual (Lista *l1, Lista *l2);** // Retorna 1 se *l1* tem pelo menos 50% dos elementos de *l2* ou retorna 0, caso contrário.
L) **Lista* removePares (Lista *l);** // Remove os elementos pares da lista.
M) **Lista* removeImpares (Lista *l);** // Remove os elementos ímpares da lista.
N) **Lista* retiraInicio (Lista *l);** // Retira o elemento do início da lista.
O) **Lista* somaInsere (Lista *l);** // Soma os elementos da lista e insere a soma no final da lista. Exemplo: *l* = {2, 4, 6} => *l* = {2, 4, 6, 12}.
P) **Lista* subtraiInsere (Lista *l);** // Subtrai os elementos da lista e insere a subtração no final da lista. Exemplo: *l* = {2, 4, 6} => *l* = {2, 4, 6, -8}.
Q) **Lista* somaPares (Lista *l);** //Soma de dois a dois elementos e inserção em uma nova lista, apenas com a soma. Exemplo: *l* = {**2**, **4**, **7**, **5**, **3**, **10**} => *l* = {**6**, **12**, **13**}

4. Considerando o uso de pilhas, implemente as seguintes funcionalidades. **Importante: A estrutura pilha fornece acesso apenas ao primeiro elemento da pilha, ou seja, o elemento do topo**.
   A) Conversão de um número decimal em binário.
   B) Seja *A* uma sequência formada por *n* números inteiros. Implemente uma função que empilhe na pilha *p1* os números pares e na pilha *p2* os números ímpares.
   C) Cópia de pilhas: Cópia da pilha p1 para a pilha p2.
   D) Intercalação de pilhas: Dada duas pilhas de elementos inteiros, codificar uma função que crie a pilha *p3* intercalando os elementos da pilha *p1* e *p2*.
   E) Pilhas iguais: Escreva uma função que determine se duas pilhas são iguais.
   F) Desempilhar 2 elementos da pilha: Escreva uma função que desempilhe dois elementos da pilha.
   G) Desempilhar *n* elementos da pilha: Escreva uma função que desempilhe *n* elementos da pilha.
   H) Tamanho da pilha: Escreva uma função que retorne a quantidade de elementos da pilha.
   I) Concatenação de pilhas: Escreva uma função que concatene a pilha *p2* na pilha *p1*.
   J) União de pilhas: Dada duas pilhas *p1* e *p2*, crie uma pilha *p3* com a união dos elementos de *p1* e *p2*, sem repetição.

K) Inserção de elementos não repetidos na pilha: Escreva uma função que permita inserir na pilha apenas elementos que já não estejam na mesma.
L) Diferença entre duas pilhas: Retorna uma pilha, com a diferença da pilha *p1* e da pilha *p2* (*p1 – p2*).

5. Considerando o uso de filas, implemente as seguintes funcionalidades. **Importante: A estrutura fila permite inserir elementos apenas no fim e remover elementos apenas do início da fila**.
   A) Considere as compras realizadas por clientes em um supermercado qualquer. Considere o uso de 3 filas, onde a fila *f1* contém apenas os clientes que estão comprando mais do que 10 produtos e não necessitam de atendimento especial. A fila *f2* representa os clientes que estão comprando até 10 produtos. Por sua vez, a fila *f3* representa os clientes que necessitam de atendimento especial, como idosos e gestantes. Implemente funções que realizem a inserção em cada uma destas filas, de acordo com o tipo de cliente que está efetuando a compra.
   B) Imprima a quantidade total de clientes do supermercado (|*f1*| + |*f2*| + |*f3*|.
   C) Faça uma função que receba três filas, duas já preenchidas em ordem crescente e preencha a última com os valores das duas primeiras em ordem crescente.
   D) Escreva um algoritmo que forneça o maior, o menor e a média aritmética dos elementos de uma fila.
   E) Escreva um programa que tenha uma fila cujos elementos possuem um campo inteiro representando sua prioridade. Insira *n* elementos com prioridades diversas na fila e depois divida a fila em duas, uma com elementos cuja prioridade é menor ou igual ao valor *p* fornecido pelo usuário e outra com os elementos restantes.
6. Considerando o uso de árvores, implemente as seguintes funcionalidades:
   A) Imprimir uma dada árvore *a1* através das ordens **pré-ordem**, **ordem simétrica** e **pós-ordem**.
   B) Criar uma função que retorne o tamanho de uma árvore *a1*.
   C) Criar uma função que conte o número de nós de uma árvore binária.
   D) Criar uma função que conte o número de folhas de uma árvore binária.
   E) Criar uma função para excluir todas as folhas de uma árvore binária, deixando a raiz e os nós internos no respectivo lugar.
   F) Criar uma função que determine se uma árvore binária é cheia ou

não.

G) Escrever uma função que determine se duas árvores binárias são iguais.
H) Escrever uma função que insira elementos em uma árvore seguindo a seguinte propriedade: A sub-árvore à esquerda contém valores sempre menores ou iguais ao valor do nó raiz. A sub-árvore à direita contém valores sempre maiores ou iguais ao valor do nó raiz.
I) Escrever uma função que conte o número de ocorrências de um determinado valor em uma dada árvore.
J) Escrever uma função que não permita a inserção de elementos repetidos em uma árvore. Considere que a árvore está ordenada, seguindo a propriedade supracitada.
K) Criar uma função para retornar o maior elemento de uma árvore.
L) Criar uma função para retornar o menor elemento de uma árvore.
M) Implementar uma função que faça a cópia de uma árvore.