Instituto de Informática – INF
NBC – Núcleo Básico Comum

Avaliação

Disciplina:	**Algoritmos e Estruturas de Dados – I**	
Professor:	Wanderley de Souza Alencar	**Nota:**
Aluno(a):		**Matrícula:**
1ª Avaliação Formal – 2020/2		**Data: de 07 a 09/05/2021**

INSTRUÇÕES PARA RESOLUÇÃO DESTA AVALIAÇÃO

1. Esta parte da avaliação possui 02 (duas) questões, todas com valor igual a 2,0 (dois) pontos;

2. As resoluções das questões de programação devem ser submetidas à área da disciplina no *Sharif Judge System*, cujo endereço é: `https://sharif.inf.ufg.br/wanderley`. Neste sistema, cada questão corresponde a 200 (duzentos) pontos e, portanto, atingindo-se 400 (quatrocentos) pontos você obteve pontuação máxima nesta parte da avaliação: 4,0 (quatro);

3. Os outros 6,0 (seis) pontos correspondem à *parte objetiva* da avaliação. Ela deverá ser respondida na área de disciplina na Plataforma Turing de Educação do INF/UFG: `https://turing.inf.ufg.br`;

4. As submissões das resoluções devem ser realizadas até às **23h59min do dia 09/05/2021 (domingo)**. Não é permitido o envio de resoluções por *e-mail*;

5. Se identificar equívocos numa determinada resolução, é possível reenviar nova versão até a data/hora limite da aplicação da avaliação.

Questão 01 (2,0 pontos). Escreva, utilizando a linguagem de programação **C** ou **C++**, um programa que possu funções que sejam capazes de realizar as operações requisitadas sobre uma *lista linear simplesmente encadeada* (LLSE), obrigatoriamente considerando que as declarações a seguir:

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  #define SUCESSO 1
5  #define FALHA -1
6
7  struct Nodo {
8        unsigned int chave;
9        float dado;
10       Apontador prox;
11 };
12
13 typedef
14    struct Nodo* Apontador;
15
16 Apontador prim;
```

As funções desejadas devem realizar as seguintes operações:

(a) Inserir na LLSE um *nodo*, cujo valor da chave primária seja igual a `chave` (um número natural) e um determinado `dado` (um número real), ambos fornecidos como parâmetros da função, de tal maneira que o *nodo* inserido fique exatamente no "*meio*" da lista.
Por exemplo: Se a lista original tiver previamente seis elementos, o *nodo* a ser inserido deverá se posicionar após o terceiro *nodo* existente.
Se tiver sete elementos, o *nodo* a ser inserido deverá também se posicionar após o terceiro *nodo* existente.
Na hipótese da lista original ter um único elemento, o *nodo* a ser inserido deverá se posicionar antes do existente.
Por fim, se a lista original estiver vazia, ocorre a simples inserção como primeiro (e único) elemento.

(b) Remover da LLSE o *nó* cujo valor para a chave primária seja igual ao parâmetro `chave`, fornecido para a função.

Observação: Deve-se submeter um programa completo, ou seja, com a função **main** e as funções solicitadas, com o programa principal permitindo o fornecimento de um para `chave` e `dado` para o item (a) e uma `chave` para o item (b).
Lembre-se: Não há "*casos de teste*" para esta questão e, por isso, você deverá realizar testes de funcionamento antes de realizar a submissão de sua resolução.

Questão 02 (2,0 pontos). Você está, neste momento, estagiando no *Centro de Tecnologia da Informação* (CTI) de uma rede de supermercados e, numa conversa com um(a) profissional que atua na área de controle de estoque, ouviu o seguinte relato dele:

"Uma das atividades mais *chatas* que tenho que fazer frequentemente é a de comparar duas grandes listas de produtos para dizer se elas são, ou não, iguais. Cada uma destas listas contém os códigos dos produtos armazenados em dois almoxarifados de nossa rede de supermercados.

O *chato* é que os códigos dos produtos, nas duas listas, estão todos fora de ordem e podem ainda, cada um, aparecer mais de uma vez.

Duas listas somente são consideradas *iguais* se contiverem os mesmos códigos de produtos, independentemente do número de ocorrências deles em cada uma das listas.

Diante disso, às vezes consumo todo um dia de trabalho para fazer esta conferência manualmente, já que cada lista pode ter até 2000 códigos."

Você então disse a ele(a): não se preocupe, vou elaborar um programa $\mathbb{C}$, ou em $\mathbb{C}++$ para resolver este seu problema em segundos, desde que você informe as duas listas que deseja comparar. E vou além: se elas forem diferentes, apresentarei os itens que as diferem.

O(A) profissional disse-lhe: você se tornará meu melhor amigo se fizer isto!

Você, então, já partindo para a resolução do problema, identificou que:
Entrada:
As duas primeiras linhas conterão, cada uma, uma lista de códigos de produtos, sendo que os códigos estarão sempre separados por um único espaço em branco e o último código será sempre igual a -1 (menos um), ou seja, ele é um *flag* para indicar o final daquela lista.

Saída:
Se as listas forem *iguais*, a saída deverá ter uma única linha contendo, nesta ordem, a

palavra `iguais` (grafada em letras minúsculas), um espaço em branco, e o número de códigos iguais.
Se as listas forem *diferentes*, a saída deverá ter uma única linha contendo, nesta ordem, a palavra `diferentes` (grafada em letras minúsculas), um espaço em branco e, por fim, a sequência de códigos distintos que diferenciam as listas. Os códigos devem ser apresentados em ordem crescente, e sem repetição, sempre separados por um único espaço em branco entre eles.

Exemplo 01:

ENTRADA	SAÍDA
1 2 3 2 5 6 5 8 7 16 7 20 9 13 29 -1	iguais 12
20 29 5 7 9 1 5 2 7 13 3 16 6 8 2 -1	

Exemplo 02:

ENTRADA	SAÍDA
1 4 16 7 14 2 15 9 3 10 16 -1	diferentes 1 3 4 7 14
2 16 10 10 16 15 9 2 16 15 -1	

Exemplo 03:

ENTRADA	SAÍDA
1 2 2 1 1 1 2 3 -1	iguais 3
1 2 3 -1	

Exemplo 04:

ENTRADA	SAÍDA
1265 5684 125 333 125 111 469 737 1652 16798 123 737 111 333 -1	iguais 10
5684 737 1652 123 111 469 333 1265 16798 125 -1 -1	

Observação: Cada uma das listas pode ter até uma sequência de até 2000 códigos, sendo que cada código de produto está no intervalo de 1 a 50000, inclusive extremos.

"Não é o conhecimento, mas o ato de aprender,
não a posse, mas o ato de chegar lá que concede a maior satisfação."
(Carl Friedrich Gauss [1777 – 1855])