BCC202 – Estruturas de Dados I (2020-02)

Departamento de Computação - Universidade Federal de Ouro Preto - MG
Professora: **Prof. Pedro Silva**

Prova 02 - `10 pontos (Peso 1,5)` (29/07/2021 às 10:10) Nota:

Nome: _____ Matrícula: _____

Leia com atenção as instruções abaixo antes de iniciar a solução da prova:

- A interpretação das questões faz parte da avaliação, por isso faça as observações que achar necessário, por escrito;

- Esta prova é **individual**, **sem consulta** e tem duração de 1 hora e 40 minutos (das 10:10 às 11:50).

- O Moodle será fechado às 12:20 (30 min extras).

- As soluções para as questões devem ser especificadas em papel e posteriormente fotografadas para submissão via Moodle.

- As folhas de respostas devem ser organizadas de maneira razoavelmente clara e coerente num único arquivo em formado *pdf* a ser submetido no Moodle. Se sua matrícula é 15.1.1234. O formato entregue deve ser *PrimeiroNome*_1511234.*pdf*.

- No caso de soluções idênticas, as pessoas envolvidas terão suas notas zeradas enquanto a situação não for devidamente esclarecidas.

- Respostas misteriosas não receberão crédito total. Uma resposta correta sem explicação, ou desenvolvimento não receberá crédito. Uma resposta incorreta apoiada por explicações substancialmente corretas pode receber crédito parcial.

- **Boa Prova!**

Utilize o valor de M como sendo a composição dos quatro últimos números da sua matrícula. Por exemplo, se sua matrícula tem o final 4381, você terá os números $M[0] = 4$, $M[1] = 3$, $M[2] = 8$ e $M[3] = 1$. Outro exemplo, se sua matrícula tem o final 1001, você terá os números $M[0] = 1$, $M[1] = 0$, $M[2] = 0$ e $M[3] = 1$.

Para as próximas questões, considere como sendo o vetor A, o vetor presente na tabela abaixo. Cada matrícula tem o seu respectivo vetor A. Preste atenção qual é o seu. **Utilizar um vetor que não é correspondente a sua matrícula implica em erro da questão.**

Matrícula	Vetor Turma 21							
20.1.4023	4	0	2	3	9	1	7	5
20.1.4019	4	0	1	9	3	7	5	2
20.1.4112	4	1	2	9	0	3	7	5
15.1.5838	5	8	3	9	0	1	7	2
20.1.4110	4	1	0	9	3	7	5	2
20.1.4139	4	1	3	9	0	7	5	2
20.1.4004	4	0	9	1	3	7	5	2
20.1.4015	4	0	1	5	9	3	7	2
20.1.4030	4	0	3	9	1	7	5	2
20.1.4007	4	0	7	9	1	3	5	2
19.2.4066	4	0	6	9	1	3	7	5
19.1.4106	4	1	0	6	9	3	7	5
19.1.4033	4	0	3	9	1	7	5	2
18.2.4215	4	2	1	5	9	0	3	7
20.1.4057	4	0	5	7	9	1	3	2
19.2.4074	4	0	7	9	1	3	5	2
20.1.4029	4	0	2	9	1	3	7	5

Matrícula	Vetor Turma 22							
20.1.4012	4	0	1	2	9	3	7	5
20.1.4111	4	1	9	0	3	7	5	2
18.2.4105	4	1	0	5	9	3	7	2
20.1.4008	4	0	8	9	1	3	7	5
19.2.4212	4	2	1	9	0	3	7	5
20.1.4114	4	1	9	0	3	7	5	2
20.1.4109	4	1	0	9	3	7	5	2
14.2.4442	4	2	9	0	1	3	7	5
20.1.4011	4	0	1	9	3	7	5	2
17.1.4001	4	0	1	9	3	7	5	2
19.2.4073	4	0	7	3	9	1	5	2
19.2.4165	4	1	6	5	9	0	3	7
19.1.4170	4	1	7	0	9	3	5	2
20.1.4009	4	0	9	1	3	7	5	2
20.1.4108	4	1	0	8	9	3	7	5
20.1.4033	4	0	3	9	1	7	5	2
19.1.4108	4	1	0	8	9	3	7	5

Matrícula	Vetor Turma 21							
20.2.4019	4	0	1	9	3	7	5	2
19.2.4208	4	2	0	8	9	1	3	7
19.2.4004	4	0	9	1	3	7	5	2
20.1.4036	4	0	3	6	9	1	7	5
20.1.4003	4	0	3	9	1	7	5	2
20.1.4018	4	0	1	8	9	3	7	5
19.1.4110	4	1	0	9	3	7	5	2
20.2.4189	4	1	8	9	0	3	7	5
20.1.4027	4	0	2	7	9	1	3	5
20.1.4002	4	0	2	9	1	3	7	5
18.2.4076	4	0	7	6	9	1	3	5
15.2.5819	5	8	1	9	0	3	7	2
19.2.4003	4	0	3	9	1	7	5	2
16.1.4323	4	3	2	9	0	1	7	5

Matrícula	Vetor Turma 22							
20.1.4022	4	0	2	9	1	3	7	5
20.1.4113	4	1	3	9	0	7	5	2
20.1.4005	4	0	5	9	1	3	7	2
20.1.4016	4	0	1	6	9	3	7	5
20.1.4038	4	0	3	8	9	1	7	5
19.2.4099	4	0	9	1	3	7	5	2
19.2.4070	4	0	7	9	1	3	5	2
18.1.4152	4	1	5	2	9	0	3	7
18.1.4171	4	1	7	9	0	3	5	2

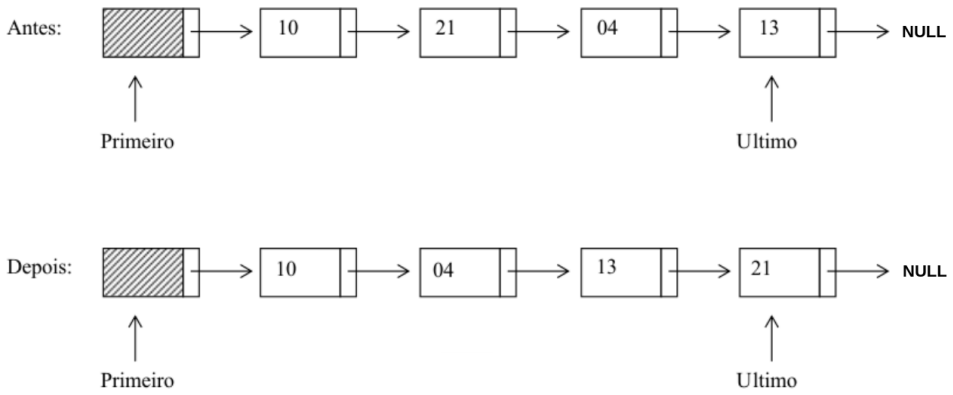
Questao 1 (2.0 pontos)

Dada uma lista encadeada com cabeça e com um número qualquer de elementos apresentada abaixo.

```
typedef struct Celula {
    struct Celula *prox;
    int item;
} TCelula;

typedef struct {
    TCelula *cabeca;
} TLista;
```

Escreva uma função $voidMoveMaior(TLista * pLista)$ que encontra o **maior** elemento da lista e o move para o **fim** da lista, como exemplificado na figura abaixo. (Obs. Não vale trocar apenas os campos item ou usar uma lista / fila / pilha auxiliar! Você deverá fazer a manipulação dos apontadores para trocar as células de posição). Qual é a ordem de complexidade da sua função. Explique.



Questao 2 (3.0 pontos)

Apresente o passo a passo da ordenação do vetor A utilizando um algoritmo que faz uso do paradigma de divisão e conquista e outro quadrático. Explique detalhadamente as mudanças feitas em cada etapa. Evite que o pior caso ocorra.

Os algoritmo de ordenação escolhidos são estáveis? Por que você escolheu cada um dos algoritmos?

Questao 3 (2.0 pontos)

Existe uma estrutura de dados chamada sacola, suportando duas operações:

- `1 x`: coloque um elemento `x` na sacola.
- `2 x`: remova algum elemento da sacola. Sendo que o valor `x` é o valor retornado pela operação.

Dada uma sequencia de operações, separadas por ';', você precisa definir qual é a estrutura de dados representada pela sacola: é uma **pilha**, uma **fila**, **outra coisa** que você não consegue identificar, ou algo **impossível** de acontecer? Inicialmente a sacola está **vazia**.

(a) `1 7 ; 2 7 ; 1 2 ; 1 9 ; 2 2 ; 1 1 ; 2 9`

(b) `1 6 ; 1 8 ; 2 8 ; 1 4 ; 1 3 ; 1 2 ; 2 6 ; 2 2 ; 2 3`

(c) `1 1 ; 2 2 ; 1 3 ; 1 4 ; 2 1 ; 2 2 ; 2 3 ; 2 4`

(d) `1 8 ; 1 7 ; 2 7 ; 1 2 ; 2 2 ; 1 4 ; 1 1 ; 2 1 ; 2 4`

(e) `2 5 ; 2 2 ; 2 8 ; 1 8 ; 1 2 ; 1 5`

Questao 4 (3.0 pontos)

```
typedef struct Celula {
    struct Celula *prox;
    int item;
} TCelula;

typedef struct {
    TCelula *cabeca;
    int tam;
} TLista;

int TLista_Retira(TLista *pLista, TItem *pX) {
    if (TLista_EhVazia(pLista))
        return 0;
    TCelula *pAux;
    pAux = pLista->cabeca->prox;
    *pX = pAux->item;
    pLista->cabeca->prox = pAux->prox;
    free(pAux);
    return 1;
}
```

Sabe-se que é possível implementar uma fila utilizando uma lista. Dada essa afirmação e considerando a função responsável por remover um item de uma lista *TLista_Retira(TLista *pLista, TItem *pX)* apresentada acima, é possível implementar a operação de desempilhar de uma TAD TFila (assinatura abaixo) utilizando a função *TLista_Retira* sem alterações? Considere que a inserção da TAD TFila é feita no início e que ela possui uma TAD TLista internamente.

```
typedef struct {
    TLista lista;
} TFila;

int TFila_Desenfileira(TFila *pFila, TItem &pX);
```

a) Se sim, a implementação é otimizada (custo constante)? Como pode ser otimizada? Descreva as alterações e faça a implementação tanto da otimização (se houver), quanto da função *TFila_Desenfileira*. (Se necessário, pode-se alterar tanto o *struct TLista* quanto a função *TLista_Retira*).

b) Se não, o que deve ser alterado para obtermos uma implementação otimizada (custo constante)? Descreva as alterações e faça a implementação tanto da alteração necessária, quanto da função *TFila_Desenfileira*.. (Se necessário, pode-se alterar tanto o *struct TLista* quanto a função *TLista_Retira*). Como fica a implementação da função *TFila_Desenfileira*?