

# Prova II - Teoria

**Entrega** 23 mai em 10:30**Pontos** 10**Perguntas** 10**Disponível** 23 mai em 8:30 - 23 mai em 10:30 2 horas**Limite de tempo** 120 Minutos

## Instruções

Nossa segunda prova de Algoritmos e Estruturas de Dados II tem duas partes: teórica e prática. Cada uma vale 10 pontos. A prova teórica será realizada no Canvas e a prática, no Verde.

A prova teórica tem 10 questões. A primeira é uma autoavaliação da sua preparação para esta prova (incluindo a qualidade da cola autorizada para esta prova) e vale 0,5 ponto. Em seguida, temos 4 questões fechadas, 2 de verdadeiro ou falso e 3 abertas. Cada fechada vale 0,4 pontos; cada verdadeira ou falso, 0,2 pontos; e cada aberta, 2,5 pontos. No caso das questões abertas, o aluno poderá enviar um arquivo contendo sua resposta ou entregá-la em na folha em branco disponibilizada pelo professor.

Abaixo, seguem as regras para a prova.

- 1) O código de acesso à prova será fornecido pelo professor no início da prova.
- 2) Após o envio de uma questão não é permitido que o aluno volte na mesma.
- 3) A prova é individual e é permitida a consulta à cola que contém o nome do aluno.
- 4) A interpretação faz parte da prova.
- 5) Se existir algum erro, após a divulgação do gabarito, peça a anulação da questão.
- 6) Os alunos da manhã 1/manhã farão a prova nos lab 1.
- 7) Os alunos da turma 2/manhã farão a prova nos lab 2.
- 8) Os alunos da tarde farão a prova no lab 11.

Desejamos uma excelente prova para todos.

Este teste não está mais disponível, pois o curso foi concluído.

## Histórico de tentativas

	Tentativa	Tempo	Pontuação
MAIS RECENTE	<a href="#">Tentativa 1</a>	21 minutos	1,7 de 10

⚠ As respostas corretas não estão mais disponíveis.

Pontuação deste teste: **1,7** de 10

Enviado 23 mai em 9:33

Esta tentativa levou 21 minutos.

### Pergunta 1

0,5 / 0,5 pts

Autoavaliação sobre sua preparação para esta prova (mínimo 0 e máximo 0,5).

0,25

### Pergunta 2

0,4 / 0,4 pts

Um ponteiro é uma variável capaz de armazenar um endereço de memória ou o endereço de outra variável. Em relação aos ponteiros, avalie as afirmações a seguir.

I. Após a sequência de instruções abaixo, p é igual a 5.

```
int *p;  
int x = 5;  
p = &x;
```

II. A saída na tela abaixo será 13

```
int soma(int *a, int *b){  
    *a = *a + *b;  
    return *a;  
}  
int main(){  
    int x = 5, y = 3;  
    x = 5; y = 3;  
    y = soma(&x, &y);  
    printf("soma: %d\n", (x + y));  
    return 0;  
}
```

III. Uma forma de alocarmos um vetor de números reais com dez posições na linguagem C é fazendo `double* vet = (double*) malloc(10 * sizeof(double) + 1);`

É correto o que se afirma em

☐ I, apenas.

☐ I e III, apenas.

☐ II e III. apenas.

☒ II, apenas.

☐ I, II e III.

Execute os dois códigos. No caso da última afirmação, a alocação deve ser feita `double* vet = (double*) malloc(10 * sizeof(double));`

Incorreta

### Pergunta 3

0 / 0,4 pts

Mostre a saída na tela para o código abaixo:

```
void fun(int *p)
{
    int q = 10;
    p = &q;
}


int main()
{
    int r = 20;
    int *p = &r;
    fun(p);
    printf("%d", *p);
    return 0;
}
```

☐ Runtime Error

☒ Compiler error

☐ 10

☐ 20

Basta executar o código. Esta questão foi obtida no [www.geeksforgeeks.org](http://www.geeksforgeeks.org)  (<http://www.geeksforgeeks.org>).

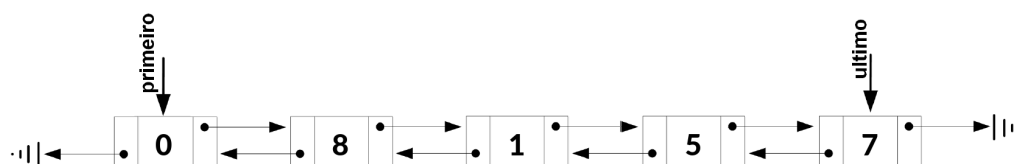
## Pergunta 4

0,4 / 0,4 pts

Os conjuntos são tão fundamentais para a ciência da computação quanto o são para a matemática. Enquanto os conjuntos matemáticos são invariáveis, os conjuntos manipulados por algoritmos podem crescer, encolher ou sofrer outras mudanças ao longo do tempo. Chamamos tais conjuntos de conjuntos dinâmicos.

CORMEN T. H., et al., **Algoritmos: teoria e prática**, 3ed, Elsevier, 2012

A figura apresentada abaixo contém uma lista duplamente encadeada, um tipo de conjunto dinâmico.



O algoritmo abaixo é manipula os elementos de uma lista duplamente encadeada.

```

Celula *i = primeiro->prox;
Celula *j = ultimo;
Celula *k;
while (j->prox != i){
    int tmp = i->elemento;
    i->elemento = j->elemento;
    j->elemento = tmp;
}
  
```

```
i = i->prox;  
for (k = primeiro; k->prox != j; k = k->prox); j = k;  
}
```

Considerando a figura e o código acima e seus conhecimentos sobre listas duplamente encadeadas, avalie as afirmações a seguir:

I. A execução do algoritmo na lista faz com que o conteúdo da lista (da primeira para a última célula) seja: 0 7 5 1 8 .

II. Sabendo que a primeira célula é o nó cabeça, o algoritmo inverte as células úteis quando temos um número ímpar dessas células.

III. Na lista inicial, a execução do comando primeiro->prox->prox->ant->elemento exibe na tela o número 8.

É correto o que se afirma em

☐ I e II, apenas.

☒ I e III, apenas.

☐ III, apenas.

☐ II, apenas.

☐ I, II e III.

I. CORRETA. O algoritmo em questão inverte a ordem dos elementos. A ordem do primeiro elemento é mantida dado que ele é o nó cabeça.

II. ERRADA - A condição  $i \neq j$  aborta o laço quando a quantidade de células é ímpar. A  $j \rightarrow \text{prox} \neq i$ , quando essa quantidade é par.

III. CORRETA. A execução do comando  $\text{primeiro} \rightarrow \text{prox} \rightarrow \text{prox} \rightarrow \text{ant} \rightarrow \text{elemento}$  exibe na tela o número 8.

Incorreta

### Pergunta 5

0 / 0,4 pts

Uma lista encadeada é uma estrutura de dados flexível composta por células sendo que cada célula tem um elemento e aponta para a próxima célula da lista. A última célula aponta para *null*. A lista encadeada tem dois ponteiros: primeiro e último. Eles apontam para a primeira e última célula da lista, respectivamente. A lista é dita duplamente encadeada quando cada célula tem um ponteiro anterior que aponta para a célula anterior. O ponteiro anterior da primeira célula aponta para *null*. A respeito das listas duplamente encadeadas, avalie as asserções a seguir.

I. Na função abaixo em C para remover a última célula, o comando "*tmp = NULL*" pode ser eliminado mantendo o funcionamento de nossa lista para outras operações. Contudo, isso pode gerar um vazamento de memória porque o espaço de memória relativo à célula removida" só será liberado no final da execução do programa.

```
int removerInicio() {  
    if (primeiro == ultimo) errx(1, "Erro!");
```

```
    CelulaDupla *tmp = primeiro;  
    primeiro = primeiro->prox;  
    int elemento = primeiro->elemento;  
    primeiro->ant = NULL;  
    tmp->prox = NULL
```

```
free(tmp);  
tmp = NULL;  
return elemento;  
}
```

### PORQUE

II. As linguagem C e C++ não possuem coleta automática de lixo como na linguagem JAVA. Essa coleta do JAVA é realiza pela Máquina Virtual Java que reivindica a memória ocupada por objetos que não são mais acessíveis.

A respeito dessas asserções, assinale a opção correta.

☐

A asserção I é uma proposição falsa, e a asserção II é uma proposição verdadeira.

☒

As asserções I e II são proposições verdadeiras, e a II é uma justificativa correta da I.

☐

As asserções I e II são proposições verdadeiras, mas a II não é uma justificativa correta da I.

☐

As asserções I e II são proposições falsas.

☐

A asserção I é uma proposição verdadeira, e a asserção II é uma proposição falsa.

A primeira afirmação é falsa. Realmente, o comando `tmp = NULL` pode ser eliminado mantendo o funcionamento de nossa lista para outras operações. Isso, porque a variável `tmp` é uma variável local cujo escopo de vida limita-se a função em questão. A remoção do comando citado não causa qualquer vazamento de memória.

A segunda afirmação é verdadeira dado que as linguagem C e C++ não possuem coleta automática de lixo e o Java possui.

### Pergunta 6

0,2 / 0,2 pts

A AVL é uma árvore de busca autobalanceada. Isso significa que cada nó da árvore possui até três descendentes (SEFAZ-PI'15, adaptado).

☒ Falso☐ Verdadeiro

A afirmação é falsa conforme o conceito de árvore AVL.

### Pergunta 7

0,2 / 0,2 pts

O comando Java `No no = new No()` cria um objeto do tipo `No` e armazena seu endereço na variável `no`. A mesma coisa acontece em C++ no código `No no = new No()`.

☐ Verdadeiro☒ Falso



O comando C++ apresentado não compila, pois "No no" cria um objeto do tipo No e, não, um ponteiro. Logo, a variável no não armazena um endereço de memória como no comando Java apresentado.

Não respondida

**Pergunta 8****0 / 2,5 pts**

Suponha uma árvore binária (não necessariamente de pesquisa) de caracteres e faça um método eficiente que retorna true se a diferença entre as alturas das folhas for maior que dois. Em seguida, mostre a complexidade do seu método.

```
class MaxMin{
    int max, min;
    MaxMin(){
        max = min = -1;
    }
}

boolean busca(){
    return busca(raiz, 0, new MaxMin());
}

boolean busca(No i, int n, MaxMin mm){
    boolean resp = false;
    if(i == null){
        if(mm.max == mm.min && mm.min == -1){
            mm.max = mm.min = n;
        } else if(n > mm.max){
            mm.max = n;
        } else if(n < mm.min){
            mm.min = n;
        }
        resp = (mm.max - mm.min >= 2);
    } else {
        resp = busca(i.esq, n+1, mm);
        if(resp == false){
            resp = busca(i.dir, n+1, mm);
        }
    }
    return resp;
} //O método faz O(n) visitas onde n é o número de nós da
árvore
```

Não respondida

## Pergunta 9

0 / 2,5 pts

Estenda a definição de sua classe **Fila Flexível** para incluir um método chamado *extrairdafila(int i)*, que pesquisa e retorna um inteiro *i* de uma **Fila**. Os demais elementos devem permanecer na Fila e ser

reorganizados quando necessário. Estime e apresente o tempo de execução de para o melhor e pior caso.

**Nesta questão você deve apresentar o código fonte da solução e explicá-lo via vídeo.**

Sua Resposta:

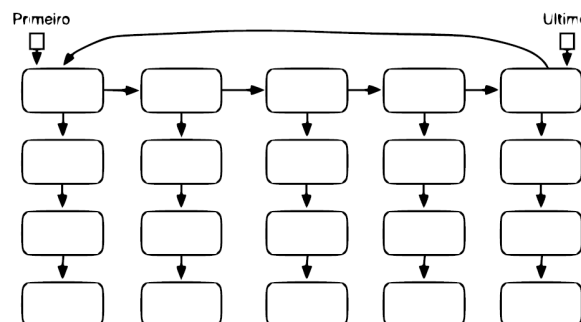
Nesta questão o aluno deve *buscar o elemento na estrutura e remove-lo com atenção aos ajustes de ponteiros que devem ser feitos.*

lo respondida

Pergunta 10

0 / 2,5 pts

Conforme mostra a Figura, consideramos um Colar a junção de uma **Fila Circular Flexível** e uma **Pilha Flexível**. Nesta estrutura, a célula Último deve referenciar a Célula Primeiro, tornando a **Fila Circular**. E cada célula da **Fila** irá conter um ponteiro para o Topo de uma **Pilha** com  $n$  elementos. Pensando nisso, você deve implementar o construtor de sua classe Colar com o seguinte cabeçalho *Colar(int m, int n)*, onde  $m$  é a quantidade de elementos na **Fila** e  $n$  a quantidade de elementos em cada **Pilha**. Além disso, faça uma análise de complexidade de seu método.



**Nesta questão você deve apresentar o código fonte da solução e explicá-lo via vídeo.**

Sua Resposta:

Pontuação do teste: **1,7** de 10