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 foi travado 23 mai em 10:30.

Histórico de tentativas

	Tentativa	Tempo	Pontuação
MAIS RECENTE	Tentativa 1	73 minutos	3,6 de 10

ⓘ As respostas corretas não estão mais disponíveis.

Pontuação deste teste: **3,6** de 10
Enviado 23 mai em 10:02

Esta tentativa levou 73 minutos.

Pergunta 1

0,5 / 0,5 pts

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

0,5

Incorreta

Pergunta 2

0 / 0,4 pts

Ponteiros são variáveis que guardam o endereço de memória. Em relação aos ponteiros, avalie as afirmações a seguir.

I. Apos a sequência de instruções abaixo, podemos afirmar que *p é igual a 10

```
int *p;
int v[]={10,7,2,6,3};
p = v;
```

II. A saída na tela abaixo será -1 2 1

```
int funcao3 (int* a, int b){
   *a = *a-b;
   return *a+b;
}
int main(){
   int x = 1, y = 2;
   int z = funcao3(&x, y);
   printf("%d %d %d\n", x, y, z);
   return 0;
}
```

III. Dado o código abaixo, os números mínimo e máximo de acessos que precisam ser feitos para localizar um registro nessa estrutura contendo n elementos é 1 e n

```
typedef struct list {
   item_type item;
   struct list *next;
} list;
list *search_list(list *l, item_type x){
   return (l == NULL) ? NULL : ( (l->item == x) ? l : search_list(l->next, x));
}
```

É correto o que se afirma em

- I e III, apenas.
- II e III. apenas.
- II, apenas.
- I, apenas.
- I. II e III.

Execute os três códigos.

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;
}
```

- 10

Compiler error

20

Runtime Error

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

Incorreta

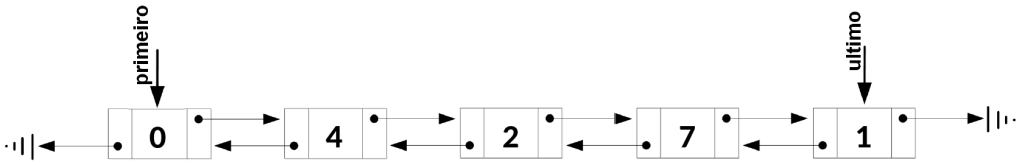
Pergunta 4

0 / 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;
Celula *j = ultimo;
Celula *k;
while (i != j){
   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: 1 7 2 4 0.

II. Na lista inicial, a execução do comando primeiro->prox->prox->prox->ant->elemento exibe na tela o número 2.

III. O algoritmo apresentado funciona corretamente quando nossa lista tem um número ímpar de células.

É correto o que se afirma em

- I e II, apenas.
- I, II e III.
- III, apenas.
- I e III, apenas.
- II, apenas.

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. CORRETA. A execução do comando primeiro->prox->prox->prox->ant->elemento exibe na tela o número 2

III. ERRADA - A condição *i != j aborta o laço quando a quantidade de células é ímpar. A j->prox != i, quando essa quantidade é par.*

Pergunta 5

0,4 / 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 "free(tmp)" 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.

- As asserções I e II são proposições falsas.
- As asserções I e II são proposições verdadeiras, mas a II não é uma justificativa correta da I.
- A asserção I é uma proposição verdadeira, e a asserção II é uma proposição falsa.
- 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 duas afirmações são verdadeiras e a segunda justifica a primeira. O comando *free(tmp)* serve para eliminar uma célula que não é mais referenciada pela nossa estrutura. Como as linguagens C/C++ não possuem coleta de lixo, o programador é o responsável por essa tarefa. A falta dessa tarefa mantém o funcionamento das demais operações da lista, contudo, isso pode gerar um vazamento de memória.

Pergunta 6

0,2 / 0,2 pts

A AVL é uma árvore de busca autobalanceada. Isso significa que pode possuir até duas raízes (SEFAZ-PI'15, adaptado).

- (●) Falso
- ( ) Verdadeiro

A afirmação é falsa conforme o conceito de árvore AVL.

Incorreta

Pergunta 7

0 / 0,2 pts

O código C++ apresentado abaixo está errado, pois acessamos o atributo de Box2 usando o símbolo “.”.

```
class Box {
   public:
      double length;   // Length of a box
      double breadth;  // Breadth of a box
      double height;   // Height of a box
};

int main() {
   Box Box1;        // Declare Box1 of type Box
   Box Box2;        // Declare Box2 of type Box
   double volume = 0.0;     // Store the volume of a box here

   // box 1 specification
   Box1->height = 5.0;
   Box1->length = 6.0;
   Box1->breadth = 7.0;

   // box 2 specification
   Box2.height = 10.0;
```

```
    Box2.length = 12.0;
    Box2.breadth = 13.0;

    // volume of box 1
    volume = Box1->height * Box1->length * Box1->breadth;
    cout << "Volume of Box1 : " << volume <<endl;

    // volume of box 2
    volume = Box2.height * Box2.length * Box2.breadth;
    cout << "Volume of Box2 : " << volume <<endl;
    return 0;
}
```

- Falso
- Verdadeiro

O código C++ apresentado, realmente, está errado. Contudo, o erro acontece porque acessamos os atributos de Box1 usando o símbolo “->”. Box1 é um objeto e está sendo usado como se fosse um ponteiro.

Pergunta 8

1 / 2,5 pts

Suponha uma árvore binária (não necessariamente de pesquisa) de caracteres e faça um método eficiente que retorna true se o caminho entre a raiz e alguma folha tiver dois ou mais dígitos. Em seguida, mostre a complexidade do seu método.

q8.java (https://pucminas.instructure.com/files/6108664/download)

```
boolean busca(){
  return busca(raiz, 0);
}
boolean condicao(char c){
  return (c >= '0' && c <= '9');
}
boolean busca(No i, int n){
  boolean resp = false;
  if(i == null){
    resp = (n >= 2);
  } else {
    n += (condicao(i.elemento)) ? 1 : 0;
    resp = busca(i.esq, n);
    if(resp == false){
      resp = busca(i.dir, n);
    }
  }
  return resp;
} //O método faz O(n) visitas onde n é o número de nós da árvore
```

A aluna fez o caminhar, contudo, não contabilizou corretamente. A Análise de complexidade não foi adequada.

Pergunta 9

1 / 2,5 pts

Seja nossa classe **Pilha Flexível**, faça um método **R.E.C.U.R.S.I.V.O** que recebe três objetos do tipo Pilha e mostra na tela a soma do primeiro elemento das três pilhas, a do segundo, do terceiro e, assim, sucessivamente. Lembre-se que as pilhas podem ter tamanhos distintos.

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

Sua Resposta:

```
public int soma (int i, int j, int k) {

   Pilha tmp = new Pilha (i);
   tmp.prox = topo;

   Pilha tmp2 = new Pilha (j);
   tmp2.prox = topo;

   Pilha tmp3 = new Pilha (k);
   tmp3.prox = topo;

   return (tmp.prox + tmp2.prox + tmp3.prox);

}
```

Nesta questão o aluno deve *desempilhar* e exibir um elemento de cada pilha, intercalando. Assim, caso uma das pilha fique vazia primeiro, a exibição deve continuar até que a outra estrutura fique vazia. Atenção as consistências que devem ser feitas para cada estrutura, verificando se a mesma esta ou não vazia.

Enunciado não atendido.

Pergunta 10

0,5 / 2,5 pts

Assuma que você tem uma **Fila Flexível** com as seguintes operações: *void enfileirar(int i), int desenfileirar(), boolean vazia()*. Como você pode usar essas operações para simular uma **Pilha Flexível**. Em particular as operações *void empilhar(int i), int desempilhar()*? Apresente as implementações para essas duas operações. Dica: use duas filas, uma principal e outra temporária.

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

Sua Resposta:

```
class Fila {

   private Celula primeiro, ultimo;

   public Fila () {
      topo = new Celula();
      ultimo = primeiro;
   }

   public void enfileirar (int i) {
      Celula tmp = new Celula(i);
      tmp.prox = topo;
      topo = tmp;
      tmp = null;
   }

   public int desenfileirar () throws Exception {
      if (vazia() == true) {
         throw new Exception ("Erro!");
      }

      Celula tmp = primeiro;
      primeiro = primeiro.prox;

      int elemento = primeiro.elemento;
      tmp.prox = null;
      tmp = null;

      return elemento;
   }

   public boolean vazia () {
      if (topo == ultimo) { return true; }

      return false;

   }
}

class SimulaPilha {

   Fila tmp = new Fila();
```

```
    Fila principal = new Fila();

    tmp.primeiro = principal.ultimo;
    tmp.ultimo = principal.primeiro;

    int x;

    principal.enfileirar(x);


}

/***** explicação *****

A Fila (estrutura utilizada no exercício)  possui os métodos
de inserção no FIM e de remoção no FIM; e a Pilha (estrutura
pedida para ser simulada)  possui os  métodos de inserção no
FIM e de remoção no INICIO. Com isso, para simular uma Pilha,
deve-se remover no INICIO, assim, temos que fazer duas filas,
sendo que uma percorre a  fila na direção correta e a outra a
percorre na direção oposta. Dessa forma, quando removermos no
FIM da primeira lista,  a remoção, na verdade,  acontecerá no
INICIO da segunda fila,  atendendo ao critério de  remoção de
uma Pilha.

*****/
```

Nesta questão espera-se que o aluno implemente uma pilha com o auxilio de duas filas para garantir a política **LIFO**. Assim, para **empilhar**, basta que ele chame o enfileirar da fila principal. Para **desempilhar**, o conteúdo da fila principal deve ser movido para temporária até que fique apenas um elemento na fila. Ele então será removido e o conteúdo movido novamente para a fila principal.

Enunciado não atendido.

Pontuação do teste: **3,6** de 10