Prova II

Entrega 18 nov em 15:20 **Pontos** 10,4 **Perguntas** 10
Disponível 18 nov em 13:25 - 18 nov em 15:20 aproximadamente 2 horas
Limite de tempo 120 Minutos

Instruções

Nossa segunda prova de AEDs 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 teórica será neste momento. A parte prática foi realizada ontem.

A prova teórica terá 8 questões fechadas no valor de 0,55 pontos e, em seguida, duas abertas no valor de 3 pontos cada. No total, distribuímos 0,4 pontos a mais. Após terminar uma questão, o aluno não terá a opção de retornar à mesma. No caso das questões abertas, o aluno poderá enviar um arquivo contendo sua resposta. Essa resposta pode ser uma imagem (foto de boa resolução) ou um código fonte. No horário da aula, você pode acessar a Prova II através do menu "Testes / Prova II".

Após a prova, para cada questão aberta, o aluno deve fazer um vídeo de no MÁXIMO 2 minutos explicando sua resposta. Vídeos com mais de 2 minutos serão penalizados. Cada vídeo deverá ser postado no YouTube e seu link submetido no Canvas até às 23:59 no dia de hoje. Vídeos alterados após essa data/horário serão zerados. Questões sem o vídeo serão penalizadas. Você pode acessar a submissão dos vídeos através do menu "Testes / Prova II - submissão de vídeos".

Histórico de tentativas

	Tentativa	Tempo	Pontuação
MAIS RECENTE	Tentativa 1	90 minutos	1,65 de 10,4 *

* Algumas perguntas ainda não avaliadas

ⓘ As respostas corretas estarão disponíveis de 18 nov em 18:00 a 24 nov em 18:00.

Pontuação deste teste: **1,65** de 10,4 *
Enviado 18 nov em 14:56
Esta tentativa levou 90 minutos.

Pergunta 1	**0,55 / 0,55 pts**

Os conceitos de endereço e ponteiro são trabalhados de forma explícita na linguagem C. Em relação aos ponteiros, avalie as afirmações a seguir.

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

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

II. A saída na tela abaixo quando o usuário digita 15 como entrada de dados será 10 15

```
int a = 5, *b, c = 10;
b = &a;
scanf("%d", b);
printf("%d %d", a, c);
```

iii. Referente a alocação dinâmica de memória em C, as funções calloc e realloc são usadas para liberar arays.

É correto o que se afirma em

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

Execute os dois códigos. No caso da terceira afirmação, em C, quem libera espaço é a função free.

Incorreta

Pergunta 2

0 / 0,55 pts

Um ponteiro ou apontador é uma variável que armazena um endereço de memória. Em relação aos ponteiros, avalie as afirmações a seguir.

I. Referente a alocação dinâmica de memória em C, a função clear é usada para limpar o conteúdo de um ponteiro.

II. Apos a sequência de instruções abaixo, se executarmos a instrução "*p = 7;", x passará a ter o valor 7.

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

III. A saída na tela abaixo será 24

```
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;
}
```

É correto o que se afirma em

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

Execute os dois códigos. No caso da primeira afirmação, em C, não existe a função clear e, sim, a free.

Incorreta

Pergunta 3

0 / 0,55 pts

Uma lista encadeada é uma representação de uma sequência de registros na memória *Random Access Memory* (RAM) do computador. Cada elemento

da sequência é armazenado em uma célula da lista sendo o primeiro elemento na primeira célula, o segundo na segunda, e assim por diante. Considere o código abaixo contendo um registro do tipo NoLista e uma função para inserir no início da lista (UFS'14 - adaptada).

```
struct NoLista {
   int elemento;
   struct NoLista *proximo;
};

struct NoLista * inserirInicio(struct NoLista *inicio, int num, int *erro){
   struct NoLista *novo;
   *erro = 0;
   novo = (struct NoLista*) malloc(sizeof(struct NoLista));
   if (novo == NULL){
      *erro = 1;
      return inicio;
   } else {
      novo->elemento=num;
      _____/* (1) */
      return novo;
   }
}
```

Para que a função, que insere um novo elemento no início da lista e retorne o início da lista, funcione corretamente, a linha em branco, marcada com o comentário (1), deve ser preenchida com

- inicio->proximo = novo;
- novo->proximo = inicio;
- **novo = inicio;**
- inicio = novo;
- novo->proximo = inicio->proximo;

O comando /* (1) */ será "*novo->proximo = inicio;*". Nesse caso, temos que o ponteiro novo aponta para a nova célula que contém o novo item e seu "próximo" aponta para o início atual. Quando a função for chamada, o novo valor de início deve ser atualizado com o endereço da célula recém criada que é retornado pela função em questão.

Incorreta

Pergunta 4

0 / 0,55 pts

A fila é uma estrutura de dados onde o primeiro elemento que entra é o primeiro a sair. Essa estrutura é usada quando desejamos que a ordem de remoção dos elementos seja igual aquela em que eles foram inseridos em nossa estrutura. Considere o código abaixo pertencente a uma Fila contendo nó cabeça e os ponteiros primeiro e último.

```
public Fila metodo(){
   Fila resp = new Fila();
      Celula i = this.primeiro.prox;
      Celula j = i.prox;

      while (j != null) {
         resp.inserir(j.elemento);
         i.prox = j.prox;
         j.prox = null;
         if (i.prox != null) {
            i = i.prox;
            j = i.prox;
         } else
            j = null;
      }
      this.ultimo = i;
   return resp;
}
```

Considerando o código acima, avalie as afirmações a seguir.

I. A execução do método mantém todos os elementos da fila inicial.

II. O método apresentado funciona corretamente quando a fila está vazia.

III. O objeto *resp* terá os elementos contidos nas posições pares de nossa fila, desconsiderando o nó cabeça.

É correto o que se afirma em

- III, apenas.
- II, apenas.
- I, II e III.
- I e II, apenas. (selected)
- I e III, apenas.

I. ERRADA - A execução do método faz com que a fila tenha somente os elementos que estavam nas posições ímpares da fila inicial, desconsiderando o nó cabeça.

II. ERRADA - Quando a fila estiver vazio, teremos um erro de execução no comando Celula j = i.prox.

III. CORRETA O objeto *resp* terá os elementos contidos nas posições pares de nossa fila, desconsiderando o nó cabeça.

Incorreta

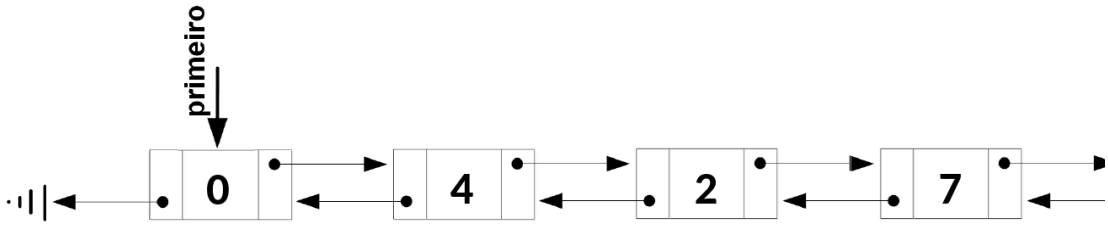
Pergunta 5

0 / 0,55 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

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

I e II, apenas.

I, II e III.

III, 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.*

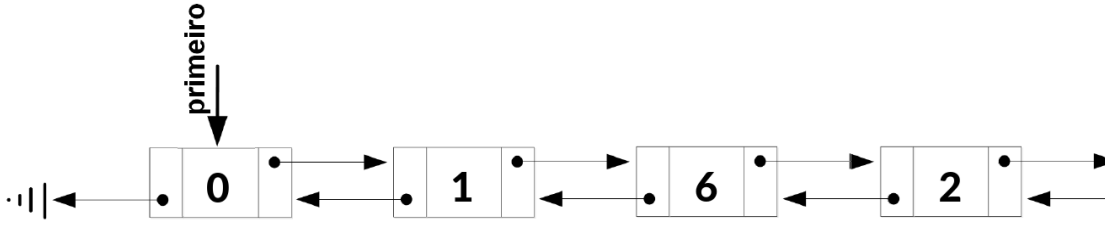
Pergunta 6

0,55 / 0,55 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 0 2 6 1.*

II. Na lista inicial, a execução do comando primeiro->prox->prox->prox->ant->elemento exibe na tela o número 6.

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

- II, apenas.
- I e III, apenas.
- I, II e III.
- I e II, apenas. (selected)
- III, 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 6

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.*dessas células.

Pergunta 7

0,55 / 0,55 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. No método abaixo em JAVA para remover a primeira célula, o comando "*tmp.prox = null*" pode ser eliminado sem qualquer prejuízo.

```
public int removerInicio() throws Exception {
   if (primeiro == ultimo) throw new Exception("Erro!”);

   CelulaDupla tmp = primeiro;
   primeiro = primeiro.prox;
   int elemento = primeiro.elemento;
   primeiro.ant = null;
   tmp.prox = null;
   tmp = null;
   return elemento;
}
```

PORQUE

II. A Máquina Virtual Java realiza a coleta lixo automática, reivindicando a memória ocupada por objetos que não são mais acessíveis.

A respeito dessas asserções, assinale a opção correta.

- A asserção I é uma proposição verdadeira, e a asserção II é uma proposição falsa.
- As asserções I e II são proposições falsas.
- As asserções I e II são proposições verdadeiras, mas a I não é uma justificativa correta da I.
- 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 *tmp.prox. = null* apenas desconecta a última célula da penúltimo. Isso é desnecessário porque a última célula não será mais referenciada e, como, explicado na segunda alternativa, a mesma será coletada pela JVM.

Incorreta

Pergunta 8

0 / 0,55 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(ultimo->prox)*" 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 removerFim() {
   if (primeiro == ultimo) errx(1, "Erro!”);

   int elemento = ultimo->elemento;
   ultimo = ultimo->ant;
   ultimo->prox->ant = NULL;
   free(ultimo->prox);
   ultimo->prox = 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, mas a I não é uma justificativa correta da I.
- A asserção I é uma proposição verdadeira, e a asserção II é uma proposição falsa.

As asserções I e II são proposições falsas.

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(ultimo->prox)* 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 9

Não avaliado ainda / 3 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:

```
public void inserirPilha(int x){
    enfileirar(x);
} // end inserirPilha()

public int removerPilha(){
    Fila tmp = new Fila();
    int tamanhoFila = tamanhoFila();
    int ultimoFila = -1;
```

```
    for(int i = 0; i < tamanhoFila-1; i++){
        int elementoTmp = desenfileirar();
        tmp.inserirPilha(elementoTmp);
    }

    ultimoFila = desenfileirar();

    this.inicio = tmp.inicio;
    this.fim = tmp.fim;

    return ultimoFila;
} // end removerPilha()
```

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.

Pergunta 10

Não avaliado ainda / 3 pts

Considere a classe **Fila Flexível** de inteiros vista na sala de aula. Crie o método *void separar(Fila f1, Fila f2)* que recebe dois objetos do tipo **Fila f1** e **f2** e copia todos os números ímpares da Fila corrente para f1 e os pares para f2. Seu método não pode utilizar os métodos de inserir e remover existentes na fila.

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

Sua Resposta:

```
public void separar(Fila f1, Fila f2) throws Exception{
    if(inicio == fim){
        throw new Exception("ERRO");
    }
```

```
    Celula tmpOrigin = this.inicio.prox;

    while(tmpOrigin != this.fim.prox){

        if(tmpOrigin.elemento % 2 == 0){
            f1.fim.prox = new Celula(tmpOrigin.elemento);
            f1.fim = f1.fim.prox;
        } else{
            f2.fim.prox = new Celula(tmpOrigin.elemento);
            f2.fim = f2.fim.prox;
        }

        tmpOrigin = tmpOrigin.prox;
    }
} // end separar()
```

Esta questão exige que o aluno desenfileire um elemento da fila principal e teste se o resto da divisão do mesmo por 2 é igual a 0. Assim, ele enfileira em **f1** se for **0** e em **f2** se for **1,** apenas com o ajuste dos valores de ultimo de cada estrutura.

Pontuação do teste: **1,65** de 10,4