Pontifícia Universidade Católica de Minas Gerais
Instituto de Ciências Exatas e Informática (ICEI)
Curso de Ciência da Computação
Disciplina: Algoritmos e Estruturas de Dados II

Prova II :

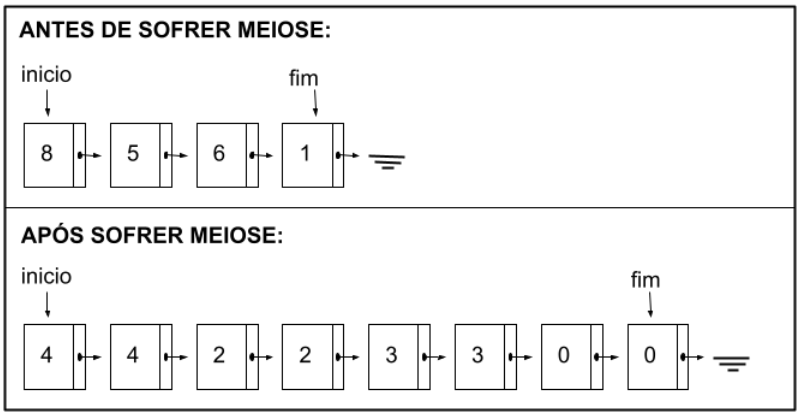
Aluno: _____

1. Dada uma lista simplesmente encadeada, implemente a função ***void meiose()***. Esta função deve percorrer toda a lista, dividindo cada célula em duas, com cada nova célula contendo a metade do valor da célula original. Abaixo, é apresentada a estrutura do código e uma ilustração que mostra a lista antes e após a operação. Apresente a complexidade do código em função de theta.

```
typedef struct Celula {
int elemento;
struct Celula* prox;
} Celula;

Celula* inicio;
Celula* fim;

void Meiose(){
   //implemente a função aqui
   }
```



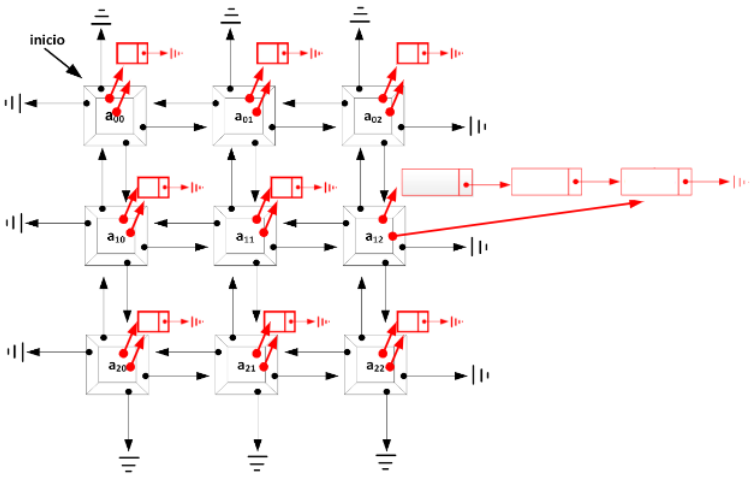
2. Considere a implementação de uma árvore binária de busca (BST) em que os nós armazenam números inteiros. Na BST vista em sala de aula, não eram aceitos elementos repetidos. Para este exercício, você deve modificar a BST para que, ao inserir um número já existente, em vez de ignorar a inserção, a árvore armazene a quantidade de vezes que cada número foi inserido. Você deve implementar o método ***void inserir(int x)***. Você pode modificar os atributos da árvore. Apresente a complexidade do código em função de theta.

```
class No {
  public int elemento;
  public No esq, dir;
}
```

```
public class Arvore {
    private No raiz;
    public void inserir(int x) {
        // Implemente o método aqui
    }
}
```

3. Considere a estrutura de classes abaixo, em Java, que representa uma matriz encadeada, onde cada célula da matriz (CelulaMatriz) contém uma lista de células (Celula). A classe Matriz possui um método diagUnificada que precisa ser implementado. Este método deve retornar um ponteiro para uma célula dupla (CelulaDupla) que será o início de uma lista duplamente encadeada contendo todas as células da diagonal principal da matriz (uma única lista duplamente encadeada resultado da concatenação de todas as listas da diagonal principal). Considere que as listas da Matriz possuem Célula Cabeça, que devem ser desconsideradas na lista resultante.

```
class Matriz{
    CelulaMatriz inicio;
    int linha, coluna;
    CelulaDupla diagUnificada(){
    //implemente seu método aqui
    }
}
class CelulaMatriz{
    CelulaMatriz esq, dir, inf, sup;
    Celula inicio, fim;
}
class Celula{
    int elemento;
    Celula prox;
}
class CelulaDupla
    int elemento;
    CelulaDupla prox, ant;
}
```



4. Provar ou refutar cada uma das afirmações a seguir, apresentando uma explicação detalhada do motivo pelo qual a afirmação é verdadeira ou falsa. Mesmo que você acredite que a afirmação esteja correta, é necessário justificar com clareza seu raciocínio.

   (a) Não é possível desenhar uma árvore binária de busca tendo apenas o caminhamento em **Pós-Ordem** a seguir:

   $$\text{Pós-ordem: } [4, 5, 6, 8, 7, 12, 18, 15, 10, 37, 42, 40, 55, 50, 20]$$

   (b) Os algoritmos Merge Sort e Heap Sort sempre tem complexidade $\Theta(n \log n)$, independentemente do caso (melhor, médio ou pior).

   (c) Dado o código abaixo, os valores de c, &c e *c correspondem a endereços de memória.

```
int a = 5;
int *b = &a;
int **c = &b;
```

   (d) A inserção e remoção de um elemento no início de uma lista encadeada tem complexidade $\Theta(1)$ apenas quando ela tem célula cabeça.

   (e) **Código de Inserção em Lista Duplamente Encadeada**

```
public void insertAtEnd(DoublyListNode head, int value) {
    DoublyListNode newNode = new DoublyListNode(value);
    DoublyListNode temp = head;
    while (temp.next != null) {
        temp = temp.next;
    }
    temp.next = newNode;
    newNode.prev = temp;
}
```

   O código acima insere um novo elemento no final de uma lista duplamente encadeada com complexidade $\Theta(n)$.