Estruturas de Dados

Teste Laboratorial 2 – *10 de Dezembro de 2018*

Nome:_____

Número:_____

Indicações gerais.

- **Responda na folha da prova. Utilize os espaços disponibilizados e o verso da folha se necessitar de mais espaço.**
- **Leia as perguntas com atenção. As soluções pedidas têm frequentemente que obedecer a determinadas restrições. Caso essas restrições não sejam cumpridas, a resposta não é valorizada.**
- **Sempre que declarar referências para tipos genéricos, deve indicar explicitamente o tipo dos respetivos parametros formais.**

2- Pretende-se construir uma estrutura de dados que armazena informação sobre jogos de futebol. Esta estrutura deve suportar, entre outras, as seguintes operações, com a complexidade indicada:

a) **`defineJogo(String eqCasa, String eqVisitante, int assistência):`** Indica a existência de um jogo entre duas equipas, com uma determinada assistência (complexidade O(1)).

b) **`getAssistênciaTotalEmCasa(String nomeEquipa)`**: devolve o número total de pessoas que assistiram aos jogos em casa da equipa indicada (complexidade O(1)).

c) **`getEquipasVisitantes(String nomeEquipa)`** : devolve uma lista com os nomes das equipas que já jogaram (na qualidade de visitante) com a equipa indicada (complexidade O(1)).

2- Pretende-se usar um priority queue para construir uma estrutura de dados que é usada para decidir qual é a proxima organização a ser alvo de uma auditoria por uma entidade fiscalizadora. Deve ser dada prioridade às auditorias às organizações com um maior número de queixas registadas. Em caso de empate, deve ser dada prioridade aquelas que foram auditadas há mais tempo.

```
class Organização{
int numeroQueixas;
Date dataUltimaAuditoria;
// Considere que duas datas d1 e d2 podem ser
/// comparadas com d1.compareTo(d2).
// d1 é considerada menor que d2 se d1 for anterior a d2.
};
```

Estruturas de Dados

Teste Laboratorial 2 – *10 de Dezembro de 2018*

Nome:_____

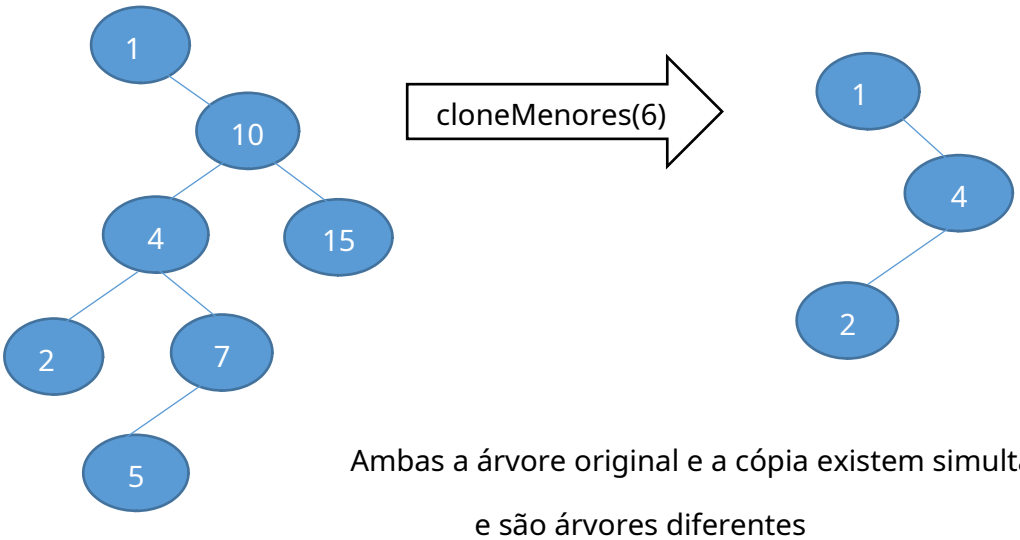
Número:_____

3- Considere a seguinte implementação de uma árvore binária de pesquisa

```
public class BinaryTree<T extends Comparable<? super T>>{

class Node{

      T value;

      Node left,right;

      Node(T t){value=t;left=right=null;}

}

// ... etc

}
```

a) Construa um método recursivo que imprime todos os valores menores do que um determinado valor recebido por parâmetro *(evite travessias desneessárias da àrvore)*

b) Construa um método recursivo (e correspondente método público) que devolve uma **cópia (clone)** da àrvore que inclui apenas todos os valores menores do que um valor X



Ambas a árvore original e a cópia existem simultaneamente

e são árvores diferentes