1-)

```
ALGORITHM FX
Input: bst BSTree
Output: ?
BEGIN
    IF exists(right(bst)) THEN
        RETURN FX(right(bst))
    ELSE
        RETURN root(bst)
END
```

Considere o pseudocódigo do algoritmo seguinte, qual o propósito destes

Resposta correta: **Determina o nó de maior valor**

2-)

Considere o Código em JAVA da implementação da classe privada treeNode. Qual o conjunto de instruções que completa corretamente o código.

```
private class TreeNode<E> {

    private E element;
    private TreeNode<E> left;
    private TreeNode<E> right;

    public TreeNode(E element, TreeNode<E> left, TreeNode<E> right) {
        this.element = element;
        this.left = left;
        this.right = right;
    }

    public TreeNode(E element) {
        A - completar
    }

    public boolean isExternalNode(){
        B - completar
    }

}
```

Resposta correta: **A - this(element,null,null); B- return(left==null && right==null);**

3-)

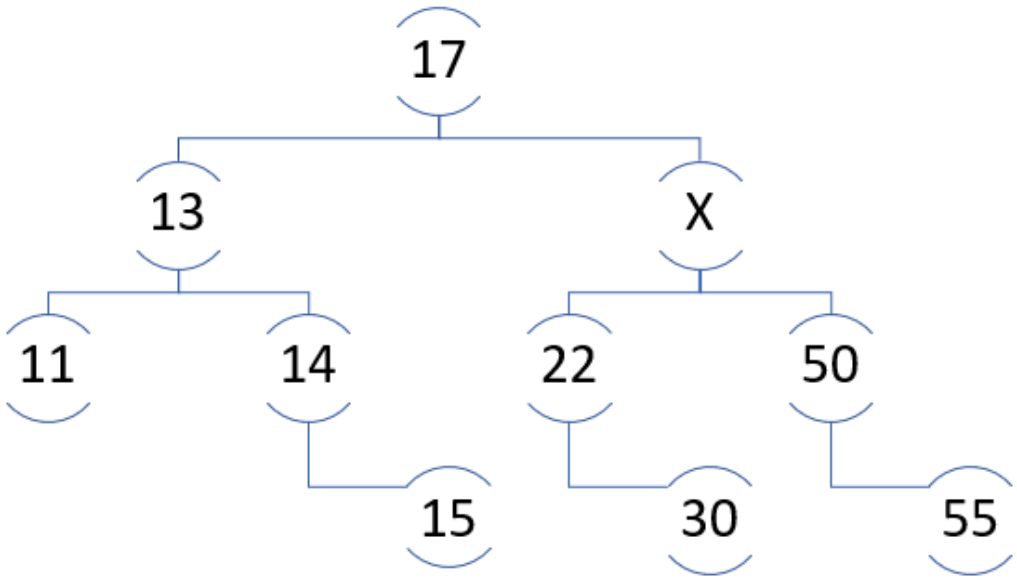
Considere o método X, que é um método da classe TreeLinked (lecionada na asulas TP).

```
public boolean X(Position<E> position, int x) throws InvalidPositionException {
    TreeNode aux = checkPosition(position);
    return aux.children.size()==x;
}
```

Resposta correta: **Verificar se o grau do nó que está na posição *pos* é igual a *x***

4-)

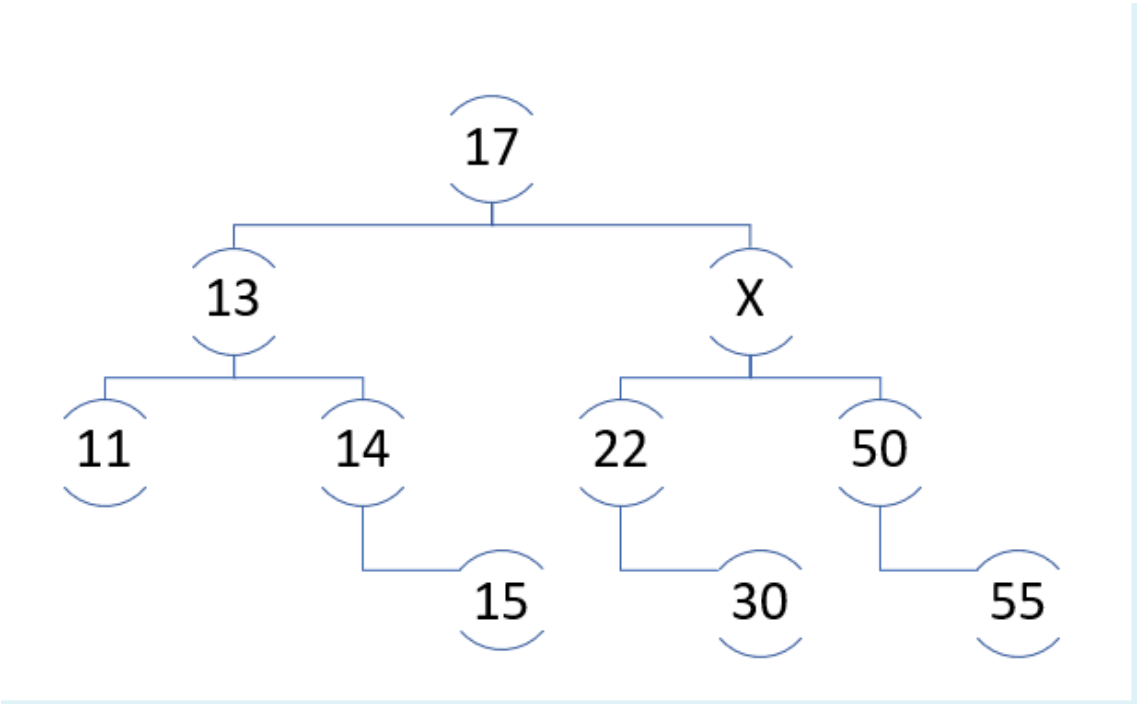
Se percorrermos a arvore de pesquisa usando o algoritmo pre-order obtemos a seguinte sequencia:



Resposta correta: **17,13,11,14,15,X,22,30,50,55**

5-)

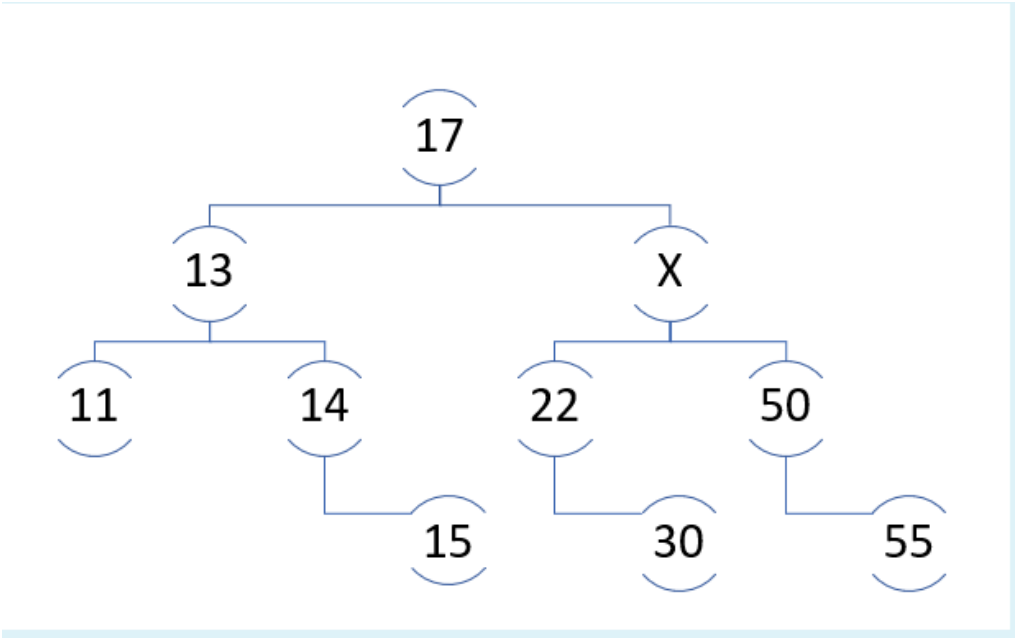
Se percorrermos a arvore de pesquisa usando o algoritmo pos-order obtemos a seguinte sequencia:



Resposta correta: **11,15,14,13,30,22,55,50,X,17**

6-)

Para ser uma arvore de pesquisa binaria, o X pode ter o seguinte valor:



Resposta correta: **31**

7-)

O método isBrothers é um método da classe TreeLinked (lecionada nas aulas TP) que verifica se as duas posições dadas, conrespondem a nós "irmãos" dentro da árvore .

```
public boolean isBrothers(Position<E> position1, Position<E> position2)
) throws InvalidPositionException {
   [B]
   [A]
}
```

Resposta correta:
[A] boolean result= parent(position1) == parent(position2); [B] return result;

8-)

Considere o código em JAVA da implementação do método recursivo inOrder. Qual o conjunto de instruções que completa corretamente o método inOrder

```
@Override
public Iterable<E> inOrder() {
    ArrayList<E> list = new ArrayList<>();
    inOrder(this.root, list);
    return list;
}

private void inOrder(TreeNode<E> treeRoot, ArrayList<E> elements) {
    if( treeRoot == null) return;
    [ a completar ]
}
```

Resposta correta: **inOrder(tree.left,elements); elements.add(treeRoot.element); inOrder(tree.right,elements);**

9-)

Relativamente à implementação do TAD MAP . Selecione a afirmação correta

Resposta correta:

Os métodos put e get tem uma complexidade algoritmica menor na implementação usando uma árvore binaria de pesquisa (BST) do que usando um ArrayList, mas o método clear tem uma complexidade idêntica.

10-)

Considere que utilizou como estrutura de dados para implementar o TAD Map uma árvore binária de pesquisa.

Se pretendermos que o método keys nos retorne uma coleção **ordenada,** deve-se utilizar a seguinte estratégia algoritmica.

Resposta correta:

Travessia da árvore no modo in-order.

11-)

Considere o Código em JAVA da implementação da classe privada treeNode. Qual o conjunto de instruções que completa corretamente o código.

```
private class TreeNode<E> {

    private E element;
    private TreeNode<E> left;
    private TreeNode<E> right;

    public TreeNode(E element, TreeNode<E> left, TreeNode<E> right) {
        this.element = element;
        this.left = left;
        this.right = right;
    }

    public TreeNode(E element) {
        A - completar
    }

    public boolean isInternalNode(){
        B - completar
    }

}
```

Resposta correta: **A - this(element,null,null); B- return ((left!=null || right!=null) && this!=root);**