Hafidzuddin Rizqy Amirullah | 5025221089

1. isBST()

```
public static boolean isBST(MyTree t) {
       // Write your codes in here
       //...
       // Write your codes in here
       return isBST(t, -9999, 9999);
   }
   // Helper function for isBST
   // Get a boolean value to know whether 't' is BST (Binary Search Tree)
   // whose values are within the range between lowerBound and upperBound
   private static boolean isBST(MyTree t, int lowerBound, int upperBound) {
       // Write your codes in here
       //...
       // Write your codes in here
       if (t.getEmpty()) {
             return true;
       }
       if (t.getValue() < lowerBound || t.getValue() > upperBound) {
             return false;
       }
       return (isBST(t.getLeft(), lowerBound, t.getValue() - 1) &&
isBST(t.getRight(), t.getValue() + 1, upperBound));
}
```

Using 1, 9, 2, 4, 0, 2, 9, 3, 0, 2, 4, 5, 6

```
Question 1: Binary Search Tree (BST) -> isBST()
The tree is not BST
```

Using 1, 4, 2, 3

```
1
   |
   |- 4
   |   |
   |   |- [nil]
   |   |
   |   |- 2
   |       |
   |       |- 3
   |       |
   |       |- [nil]
   |
   |- [nil]


Question 1: Binary Search Tree (BST) -> isBST()
The tree is BST
```

2. printDescending()

```
public static void printDescending(MyTree t) {
	if(!t.getLeft().getEmpty()) {
		printDescending(t.getLeft());
	}
	System.out.println(t.getValue());
	if(!t.getRight().getEmpty()) {
		printDescending(t.getRight());
	}

}
```

Using 1, 9, 2, 4, 0, 2, 9, 3, 0, 2, 4, 5, 6

```
Question 2: printDescending()
0
0
1
2
2
2
3
4
4
5
6
9
9
```

Using 1, 4, 2, 3

```
Question 2: printDescending()
1
2
3
4
```

3. max()

```
public static int max(MyTree t) {
	if(t.getRight().getEmpty()) {
		return t.getValue();
	}

	return max(t.getRight());
}
```

Using 1, 9, 2, 4, 0, 2, 9, 3, 0, 2, 4, 5, 6

```
Question 3: max()
Max value of the tree: 9
```

Using 1, 4, 2, 3

```
Question 3: max()
Max value of the tree: 4
```

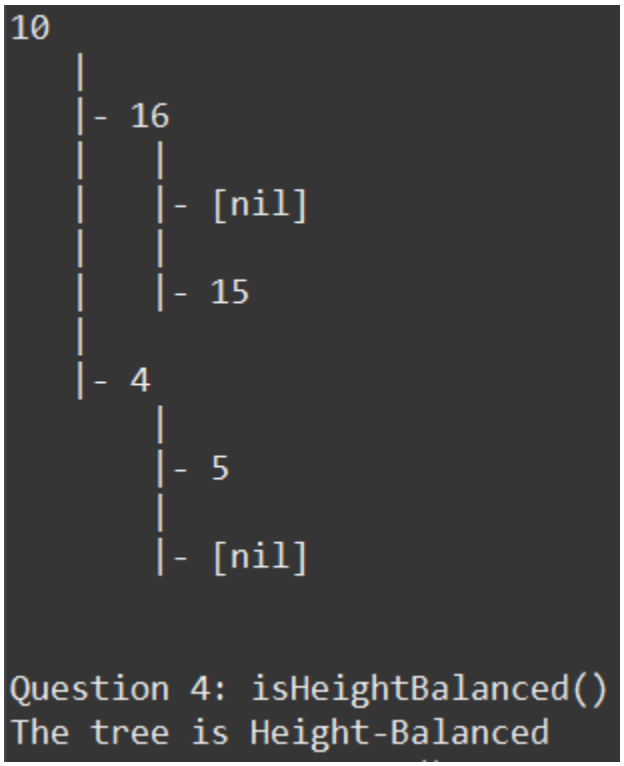
4. isHeightBalanced()

```
public static boolean isHeightBalanced(MyTree t) {
		// Write your codes in here
	//...
	// Write your codes in here
		if (t.getEmpty()) {
			return true;
		}

		if (Math.abs(getHeight(t.getLeft()) - getHeight(t.getRight())) > 1) {
			return false;
		}

		return isHeightBalanced(t.getLeft()) &&
isHeightBalanced(t.getRight());
}
```

```
private static int getHeight(MyTree t) {
		if(t.getEmpty()) {
			return 0;
		}
		return (1 + Math.max(getHeight(t.getLeft()),
getHeight(t.getRight())));
}
```



5. insertHB()

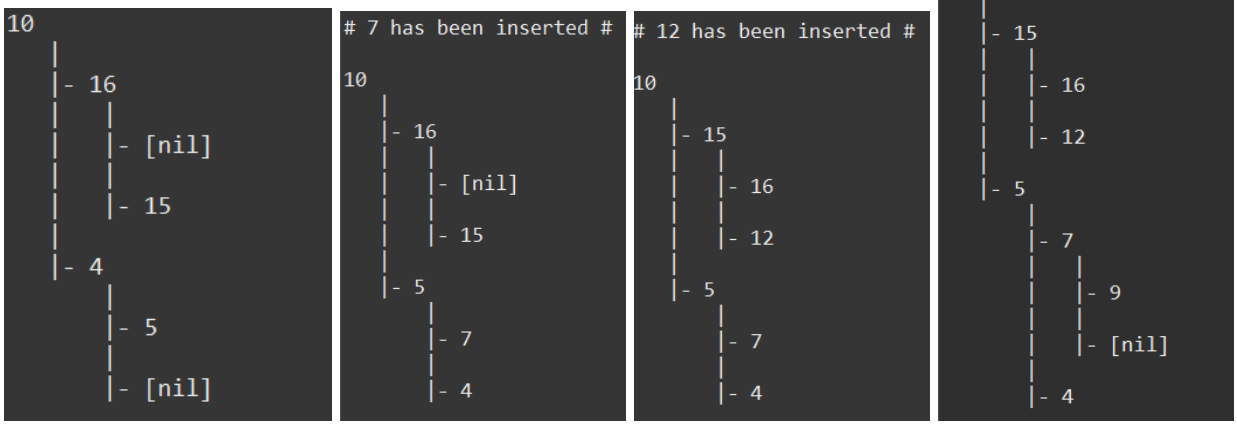
```
public static MyTree insertHB(int n, MyTree t) {
		if (t.getEmpty()) {
			return new MyTree(n, new MyTree(), new MyTree());
		}
		else if(n < t.getValue()) {
			MyTree _tempLeft = insertHB(n, t.getLeft());
			t = new MyTree(t.getValue(), _tempLeft, t.getRight());
		}
		else if(n > t.getValue()) {
			MyTree _tempRight = insertHB(n, t.getRight());
			t = new MyTree(t.getValue(), t.getLeft(), _tempRight);
		}

		int _bFactor = getBalanceFactor(t);

		if(_bFactor > 1 && n < t.getLeft().getValue()) {
			return rebalanceForRight(t);
		}
		if(_bFactor > 1 && n > t.getLeft().getValue()) {
			t = new MyTree(t.getValue(), rebalanceForLeft(t.getLeft()),
t.getRight());
			return rebalanceForRight(t);
		}
		if(_bFactor < -1 && n > t.getRight().getValue()){
			return rebalanceForLeft(t);
		}
		if(_bFactor < -1 && n < t.getLeft().getValue()) {
			t = new MyTree(t.getValue(), rebalanceForRight(t.getLeft()),
t.getRight());
			return rebalanceForRight(t);
		}

		return t;
}
```

```
private static int getBalanceFactor(MyTree t) {
	if(t.getEmpty()) {
		return 0;
	}
	return getHeight(t.getLeft()) - getHeight(t.getRight());
}
```



6. rebalanceForLeft()

```
private static MyTree rebalanceForLeft(MyTree t) {
	MyTree _newParent = t.getRight();
	MyTree _tempTree = _newParent.getLeft();
	t = new MyTree(t.getValue(), t.getLeft(), _tempTree);

	return new MyTree(_newParent.getValue(), t, _newParent.getRight());
}
```

7. rebalanceForRight()

```
private static MyTree rebalanceForRight(MyTree t) {
	MyTree _newParent = t.getLeft();
	MyTree _tempTree = _newParent.getRight();
	t = new MyTree(t.getValue(), _tempTree, t.getRight());

	return new MyTree(_newParent.getValue(), _newParent.getLeft(), t);
}
```

8. deleteHB()

```
public static MyTree deleteHB(MyTree t, int x) {
	if (t.getEmpty()) {
		return t;
	}
	if (x > t.getValue()) {
		t = new MyTree(t.getValue(), t.getLeft(), deleteHB(t.getRight(), x));
	}
	else if (x < t.getValue()) {
		t = new MyTree(t.getValue(), deleteHB(t.getLeft(), x), t.getRight());
	}
	else {
		MyTree _temp = null;
		if (t.getLeft().getEmpty() || t.getRight().getEmpty()) {
			if (t.getLeft().getEmpty()) {
				_temp = t.getRight();
			}
			else if (t.getRight().getEmpty()) {
				_temp = t.getLeft();
			}

			if(_temp.getEmpty()) {
				_temp = t;
				t = new MyTree();
			}
			else {
				t = _temp;
			}
		}
		else {
			_temp = getLeftMost(t.getRight());
			t = new MyTree(_temp.getValue(), _temp.getLeft(),
deleteHB(t.getRight(), _temp.getValue()));
		}
	}

	if (t.getEmpty()) {
		return t;
	}

	int _balanceFactor = getBalanceFactor(t);

	if(_balanceFactor > 1 && getBalanceFactor(t.getLeft()) >= 0) {
		return rebalanceForRight(t);
	}
	if(_balanceFactor > 1 && getBalanceFactor(t.getLeft()) < 0) {
		t = new MyTree(t.getValue(), rebalanceForLeft(t.getLeft()),
```

```
t.getRight());
                return rebalanceForRight(t);
        }
        if(_balanceFactor < -1 && getBalanceFactor(t.getRight()) <= 0) {
                return rebalanceForLeft(t);
                }
        if(_balanceFactor < -1 && getBalanceFactor(t.getRight()) > 0) {
              t = new MyTree(t.getValue(), rebalanceForRight(t.getLeft()),
t.getRight()); //right left
                return rebalanceForLeft(t);
         }

        return t;
}
```

```
public static MyTree getLeftMost(MyTree t) {
        MyTree _currentNode = t;
        while (!_currentNode.getLeft().getEmpty()) {
                _currentNode = new MyTree(_currentNode.getLeft().getValue(),
_currentNode.getLeft().getLeft(), _currentNode.getLeft().getRight());
        }

        return _currentNode;
}
```

```
# 9 has been inserted #

10
   |
   |- 15
   |    |
   |    |- 16
   |    |
   |    |- 12
   |
   |- 5
        |
        |- 7
        |    |
        |    |- 9
        |    |
        |    |- [nil]
        |
        |- 4
```

```
# 7 has been deleted #

10
    |
    |- 15
    |    |
    |    |- 16
    |    |
    |    |- 12
    |
    |- 5
         |
         |- 9
         |
         |- 4
```

```
# 12 has been deleted #

10
    |
    |- 15
    |    |
    |    |- 16
    |    |
    |    |- [nil]
    |
    |- 5
         |
         |- 9
         |
         |- 4
```

```
# 9 has been deleted #

10
   |
   |- 15
   |   |
   |   |- 16
   |   |
   |   |- [nil]
   |
   |- 5
       |
       |- [nil]
       |
       |- 4
```

```
# 10 has been deleted #

15
   |
   |- 16
   |
   |- [nil]
```

```
# 15 has been deleted #

16
```