/\*\*

\* Returns the {@code String} prefix representation of the given

\* {@code BinaryTree<T>}.

\*

\* **@param** <T>

\* the type of the {@code BinaryTree} node labels

\* **@param** t

\* the {@code BinaryTree} to convert to a {@code String}

\* **@return** the prefix representation of {@code t}

\* **@ensures** treeToString = [the String prefix representation of t]

\*/

public static <T> String treeToString(BinaryTree<T> t) {

String result = "()";

BinaryTree<T> left = null;

BinaryTree<T> right = null;

T root = t.root();

if (!root.equals(null)) {

t.disassemble(left, right);

result = t.root().toString() + "(" + *treeToString*(left) + *treeToString*(right) + ")";

}

t.assemble(root, left, right);

return result;

}

/\*\*

\* Returns a copy of the the given {@code BinaryTree}.

\*

\* **@param** t

\* the {@code BinaryTree} to copy

\* **@return** a copy of the given {@code BinaryTree}

\* **@ensures** copy = t

\*/

public static BinaryTree<Integer> copy(BinaryTree<Integer> t) {

BinaryTree<Integer> left = null;

BinaryTree<Integer> right = null;

BinaryTree<Integer> copy = null;

Integer root = t.root();

if (!root.equals(null)) {

t.disassemble(left, right);

copy.assemble(root, *copy*(left), *copy*(right));

}

t.assemble(root, left, right);

return copy;

}