```java
import components.binarytree.BinaryTree;

/**
 * {@code Queue} represented as a {@code Sequence} of entries, with
 * implementations of primary methods.
 *
 * @param <T>
 *            type of {@code Queue} entries
 * @correspondence this = $this.entries
 */
public class HelloWorld {

    /**
     * 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 rep = "";
        if (t.root().equals("")) {
            rep += "()";
        } else {
            rep += t.root() + "(";
            BinaryTree<T> left = t.newInstance();
            BinaryTree<T> right = t.newInstance();
            T temp = t.disassemble(left, right);
            String leftString = treeToString(left);
            String rightString = treeToString(right);
            rep += leftString + rightString;
            rep += ")";
            t.assemble(temp, left, right);
        }

        return rep;
    }

    /**
     * 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> tree = t.newInstance();

        if (t.size() != 0) {
            BinaryTree<Integer> left = t.newInstance();
            BinaryTree<Integer> right = t.newInstance();
            Integer root = t.disassemble(left, right);
            BinaryTree<Integer> copyLeft = copy(left);
            BinaryTree<Integer> copyRight = copy(right);
            tree.assemble(root, copyLeft, copyRight);
            t.assemble(root, left, right);
        }
        return tree;
    }

}
```