```java
import components.sequence.Sequence;

/**
 * Check if a given {@code Sequence<Integer>} is a palindrome.
 *
 * @author Put your name here
 *
 */
public final class SequencePalindrome {

    /**
     * Private constructor so this utility class cannot be instantiated.
     */
    private SequencePalindrome() {
    }

    /**
     * Construct and return a sequence from a given array.
     *
     * @param args the array of integer
     * @return the sequence of integer
     * @ensures createFromArgs = [the sequence of integers in args]
     */
    private static Sequence<Integer> createFromArgs(int[] args) {
        assert args != null : "Violation of: args is not null";
        Sequence<Integer> s = new Sequence1L<Integer>();
        for (int x : args) {
            s.add(s.length(), x);
        }
        return s;
    }

    /**
     * Checks if a given {@code Sequence<Integer>} is a palindrome.
     *
     * @param s the {@code Sequence} to check
     * @return true if the given {@code Sequence} is a palindrome, false otherwise
     * @ensures isPalindrome = (s = rev(s))
     */
    private static boolean isPalindrome(Sequence<Integer> s) {
        assert s != null : "Violation of: s is not null";

        boolean result = true;

        // get length
        int len = s.length() - 1;
        int x = 0, y = 0;
        // int i = 0;

        // loop if entry(x) == entry(len-x)
        // while (i < len) {

        // if (s.entry(i) != s.entry(len-i)) {
        // result = false;
        // }
        /*
         * I have NO idea why this works for everything except the few arrays with 512
         * as the only repeated number Figured this would be a quick and dirty lab but
         * maybe I'm in need of a refresher
         */
        // i++;
        // }

        // Recursion never lets me down :)

        if (s.length() <= 1) {
            result = true;
        } else {
            x = s.remove(len);
            y = s.remove(0);

            if (x != y) {
                result = false;
            } else {
                result = isPalindrome(s);
            }

            s.add(s.length(), x);
            s.add(0, y);
        }

        // This line added just to make the program compilable.
        return result;
    }

    /**
     * Main method.
     *
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        SimpleWriter out = new SimpleWriter1L();

        final int[][] sequences = { {}, { 1 }, { 2, 2 }, { 3, 4, 3 }, { 5, 6, 7, 8, 8, 7, 6, 5 },
                { 9, 10, 11, 12, 13, 12, 11, 10, 9 }, { 1, 2 }, { 3, 4, 5 }, { 6, 7, 8, 8, 7, 9 },
                { 10, 11, 12, 12, 13, 10 }, { 14, 15, 16, 17, 15, 14 }, { 6, 7, 8, 18, 8, 7, 9 },
                { 10, 11, 12, 19, 12, 13, 10 }, { 14, 15, 16, 20, 17, 15, 14 }, { 512 }, { 512, 512 },
                { 512, 512, 512 }, { 512, 512, 512, 512 } };
        final boolean[] results = { true, true, true, true, true, true, false, false, false, false, false, false, false,
                false, true, true, true, true };

        for (int i = 0; i < sequences.length; i++) {
            Sequence<Integer> s = createFromArgs(sequences[i]);
            Sequence<Integer> sCopy = createFromArgs(sequences[i]);
            /*
             * Check returned result and parameter restores mode
             */
            boolean correctResult = (isPalindrome(s) == results[i]);
            boolean restoredParameter = s.equals(sCopy);
            if (correctResult && restoredParameter) {
                out.print("    Test passed: " + s + " is ");
                if (!results[i]) {
                    out.print("not ");
                }
                out.println("a palindrome");
            } else {
                if (!correctResult) {
                    out.print("*** Test failed: " + sCopy + " is ");
                    if (!results[i]) {
                        out.print("not ");
                    }
                    out.println("a palindrome");
                }
                if (!restoredParameter) {
                    out.println("*** Test failed: " + s + " was not restored to its original value " + sCopy);
                }
            }

            out.println();
        }

        out.close();
    }

}

```