RELAXED PREFIX (60 minutes)
===========================

Description
-----------

Verify a function isRelaxedPrefix determining if a list _pat_ (for pattern) is a relaxed prefix of another list _a_.

The relaxed prefix property holds iff _pat_ is a prefix of _a_ after removing at most one element from _pat_.

Examples
--------

pat = {1,3}   is a relaxed prefix of a = {1,3,2,3} (standard prefix)

pat = {1,2,3} is a relaxed prefix of a = {1,3,2,3} (remove 2 from pat)

pat = {1,2,4} is not a relaxed prefix of a = {1,3,2,3}.

Implementation notes
--------------------

You can implement lists as arrays, e.g., of integers. A reference implementation is given below. It may or may not contain errors.

```
public class Relaxed {

    public static boolean isRelaxedPrefix(int[] pat, int[] a) {
        int shift = 0;

        for(int i=0; i<pat.length; i++) {
            if (pat[i]!=a[i-shift])
                if (shift==0) shift=1;
                    else return false;
        }
        return true;
    }
    public static void main(String[] argv) {
        int[] pat = {1,2,3};
        int[] a1 = {1,3,2,3};
        System.out.println(isRelaxedPrefix(pat, a1));
    }

}
```

Advanced verification task (if you get bored)
---------------------------------------------

Implement and verify a function relaxedContains(pat, a) returning whether _a_ contains _pat_ in the above relaxed sense, i.e., whether _pat_ is a relaxed prefix of any suffix of _a_.