Programming - Unit 2
Homework 1

submit by 14th March 2021

We call *empty* a stack s such that $pop(s)$ yields an exception. We write $rest(s)$ the stack resulting after popping one element out of a nonempty stack s. Let $\leq_S$ be the *partial* order on stacks of integers defined as follows: $s1 \leq_S s2$ if and only if either $s1$ is empty or, if it isn't, $s2$ is also not empty and moreover:

- $pop(s1) \leq pop(s2)$, and
- $rest(s1) \leq_S rest(s2)$.

Check that $\leq_S$ is a partial order (reflexive, antisymmetric and transitive).

Write a `StackUtil` Java class providing, among possible others, two methods:

- `public static boolean leqS (ListStack s1, ListStack s2)`
- `public static void sortS (ListStack[] A)`

where `leqS(s1, s2)` evaluates to `true` if `s1` $\leq_S$ `s2`, to `false` otherwise, while `sortS(A)` sorts the array `A` in such a way that, for all $i < j <$ `A.length`, `leqS(A[j], A[i]) == false`. Note that `leqS(A[j], A[i]) == false` does not imply `A[i]` $\leq_S$ `A[j]`.

The following main should print what is specified in the comments:

```
public class testSort {

   public static void main(String[] args) {

      ListStack s1 = new ListStack();
      ListStack s2 = new ListStack();

      s1.push(5);
      s2.push(6);

      s1.push(2);
      s2.push(2);

      System.out.println(StackUtil.leqS(s1, s2)); // prints true
      s1.push(1);
      System.out.println(StackUtil.leqS(s1, s2)); // prints false

      ListStack s3 = new ListStack();
      s3.push(0);
```

```
        ListStack[] A = new ListStack[3];
        ListStack [0] = s1;
        ListStack [1] = s2;
        ListStack [2] = s3;

        StackUtil.sortS(A);
        System.out.println(A[0].pop()); // prints 0
    }
}
```