Exercise 3: Stacks

You may work on this exercise with another student. If you do, write **both names in the header**, but turn in **two copies** of the project, one with your name on the folder and the other with the other student's name on the folder.

Folder name: A200_E3_YourLastName_YourFirstName.

Two **stacks** of **positive integers** are needed, one containing elements with **values less than or equal to 1,000** and the other containing elements with values **larger than 1,000**.

The user enters a **maximum limit**, which denotes that the numbers entered will *not* exceed that limit. For example, the user enters 4, 5, 6, 1020, 1030, 7, 1040, and these numbers will be stored as if they were inserted in two stacks, one that contains elements less or equal to 1,000, and one that contains elements larger than 1,000. Although the total number of elements in the small-value stack and the large-value stack combined are never going to exceed the limit, we cannot predict how many integers the user will enter and how many will be placed in each stack. (All of the elements could be in the small-value stack, or they could be evenly divided, or both stacks could be empty, and so on.)

Describe how you can represent the two stacks in **one single array** by drawing the array and showing how it would look (give an example with four numbers smaller than 1,000 and three numbers larger than 1,000 in an array of capacity 10). Make sure you identify the top for both stacks.

The problem asks only to enter numbers into two stacks and then print all the numbers. Since you are manipulating a stack, it is assumed that printing the numbers will empty the stack.

Below you can find an output example (items **in red** are user's input):

```
Enter max number of integers: 10
Enter integers (-1 to quit): 1 2 3000 4 5000 6000 7 8 -1

Stack with small values: 8 7 4 2 1
Stack with large values: 6000 5000 3000

Press any key to continue . . .
```

Complete the given project by implementing the stacks as **one single array** and creating the following functions:

- Function **processStacks**
    - Interacts with the user and fills out the array.

- Function **printSmallValues**
    - Prints values smaller than or equal to 1,000 with running time O(n), where n is the number of elements smaller than or equal to 1,000.
    - Since we are simulating a stack, the print out should be in a LIFO order.
- Function **printLargeValues**
    - Prints values larger than 1,000 with running time O(m) , where m is the number of elements larger than 1,000.
    - Since we are simulating a stack, the print out should be in a LIFO order.