Lab 4: Stacks

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

The user enters a **maximum limit**, which denotes that the numbers entered will ***not*** exceed that limit. The numbers that the user enters will be stored **AS IF** they were inserted into two stacks, one that contains elements smaller 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 given by the user, we cannot predict how many integers smaller than or equal to 1,000 the user will enter or how many integers greater than 1,000 the user will enter (i.e., all of the elements could be in the small-value stack, or they could all be in the large-value stack, or they could be evenly divided or unevenly divided).

Here is the trick: Your implementation must be completed using **the only dynamic array already declared in the main function**; you may **NOT** use any other containers (**NO** stacks, **NO** vectors, **NO** arrays, etc.)**.** You need to figure out how an array can contain all the numbers in a way that it behaves like two stacks, one with numbers smaller or equal to 1,000 and the other with number greater than 1,000.

Functions to implement:

- **processStacks**
  - Interacts with the user and fills out the array that represents the two stacks.
- **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.
- **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.

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

Assumptions:

- The user will enter at least one number.
- The user will not exceed the capacity of the array.

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 (top): 8 7 4 2 1
Stack with large values (top): 6000 5000 3000

Press any key to continue . . .
```