CSL 201
Assignment 1

Stack Permutations

Before presenting the problem statement of the assignment let us look at the definition of Stack.

Stack: Stack is a very important ordered data structure useful in solving wide range of problems. A stack can be thought of a pile of ojbects as shown in figure 1, in which you can put items one after another in an order say a1, a2, a3,.., an. In stack, the last added item will be at the top of the pile and the first added item at the bottom. The operation of putting an item in a stack is called **push.**

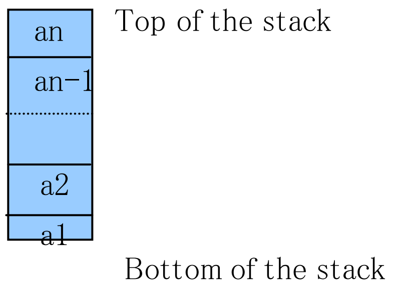

Figure 1. Stack in pictorial form.

An item can be taken out from the box using **pop** operation. **Pop** removes the topmost element of the stack and returns it. For example, if the current state of the stack is as shown in figure 1 then **pop** returns the element 'an' and as a result 'an-1' comes to the top of the stack. If you again perform a **pop** operation on this stack then it will return 'an-1', the value at the top of the stack. For this reason stacks are sometimes referred to as Last In First Out (LIFO) lists.

One way of implementing a stack is using an array where 0th index of this array represents the bottom of the stack and the index keeps on increasing as elements are added. Thus at any point, index points to the topmost element in the stack (array here). For further information on stack and how to use an array for implementing it, please refer to the following links.

http://www.cs.bu.edu/teaching/c/stack/array/
http://www.cmpe.boun.edu.tr/~akin/cmpe223/chap2.htm
http://en.wikipedia.org/wiki/Stack_(abstract_data_type)

Problem statement: Permutation of a sequence of n numbers from 1 to n is the collection of all possible arrangements of these numbers.Using a stack and its functions (push, pop) one can generate the permutations of the given numbers. But not all the permutations can be found by this method. Example, given a ordered sequence of 5 numbers say, 1,2,3,4,5 and the possible operations - Push, Pop, Print where -

1. Push(int n) - pushes a number on to the stack and removes it from the input.
2. Pop() – returns the top number on stack and prints it to the output.
3. Print() – prints the current number from the input list to the output.

we can have a permutation as - 1,3,5,4,2 which is generated by the following sequence of operations –

	Operation	Stack	Output
1.	print()	–	1
2.	push(2)	2	1
3.	print()	2	1,3
4.	push(4)	2,4	1,3
5.	print()	2,4	1,3,5
6.	pop()	2	1,3,5,4
7.	pop()	–	1,3,5,4,2

But we cannot generate 1,3,5,2,4 by this (check!). Thus 1,3,5,4,2 is a "stack permutation" but 1,3,5,2,4 is not. Now given a sequence, write a program which would say whether the given permutation is a stack permutation of the given sequence or not and if it is then what sequence of operations led to its generation. Formally,
input - a positive nonzero number n and then any permutation of 1, 2, .., n.
output - "yes" followed by the sequence of operations (push, pop and print) that can generate it from the sequence of 1, 2, .., n. [If it is a stack permutation], or
"No" [otherwise]

Note :

1. For a input "n", the sequence against which stack permutation has to be checked is in the form of 1, 2, .., n.
2. Input can be either from file or standard input. (Take that as argument - 1 for standard input and 2 for file input followed by the file name). For example,
   java StackPermutation 2 input.txt, or
   java StackPermutation 1
3. For stack implementation - do not use the data structure provided for stack in JAVA library. You are supposed to implement a stack yourself using Arrays.
4. For nth number, print() is same as push(n) followed by pop(). Prefer using single print().
5. Input needs to be validated properly by your code. Some examples of wrong inputs are;
   1. Case 1: "n" is given as 5 and the permutation is given as 1 2 4 3. This is an invalid input because a permutation of 1,2,3,4,5 must have exact 5 digits.
   2. Case 2: "n" is given as 5 and the given permutation is 1 2 6 5 4. This is an invalid input because 1,2,6,5,4 can not be a permutation of 1,2,3,4,5. Therefore first check if the permutation contains a number greater than "n" or not.
   3. Case 3: Repetition of a number in permutation sequence. For example, for "n" as 4 the permutation sequence 1,2,3,3 is not a valid input.
6. In case of invalid input print an appropriate error message and exit from the program.

Sample input 1-

5
1
3
5
4
2
output -
Yes
print()
push(2)
print()
push(4)
print()
pop()
pop()

Sample input 2-
5
1
3
5
2
4

Sample output -
No

Wrong input 1-
5
1
2
3
4

Wrong input 2-
5
1
2
6
4
5

Wrong input 3-
3

1
2
2
(Because of the repetition of a digit 2 in the permutation sequence)