Max Stack

Input file:	`standard input`
Output file:	`standard output`
Time limit:	0.5 seconds
Memory limit:	256 megabytes

Ayush is upset with the performance in the last DSA labs. He really wants the performance to improve this time. Hence, to simplify the questions, he has given specific instructions as to how the question needs to be implemented.

You need to design a stack that performs the following operations in **constant time** -

- **push(x)** - pushes the integer x at the top of the stack
- **pop()** - removes the element from the top of the stack and returns it. If the stack is empty it returns -1.
- **top()** - returns the element at the top of the stack. If the stack is empty it returns -1.
- **getMax()** - returns the maximum element in the stack. If the stack is empty it returns -1.

Input

The first line of input has a single integer **Q** indicating the number of queries. The following Q queries will have the given format -

- 1 x - Push integer x into the top of the stack (do not return anything)
- 2 - Pop the element from the top of the stack and print it.
- 3 - Print the top element from the stack
- 4 - Print the maximum element in the stack

$1 \leq \mathbf{Q} \leq 4 * 10^4$, $-2^{31} \leq \mathbf{x} \leq 2^{31} - 1$

Output

For each instance of the queries 2, 3, and 4 print the answer in a new line. Print -1 in case the stack is empty(in queries of type 2,3 and 4).

Example

standard input	standard output
9 1 33 1 0 1 100 1 -10 3 4 2 2 4	-10 100 -10 100 33

Note

In the above test case -

- 33 is added to the stack
- 0 is added to the stack
- 100 is added to the stack.
- -10 is added to the stack.
- -10 is at the top of the stack and is printed.
- 100 is the maximum element in the stack and is printed.
- -10 is at the top of the stack and is printed and popped out.
- 100 is at the top of the stack and is printed and popped out.
- 33 is the maximum number in the stack and and is printed.