## Stack using linked List

* Implement a Stack Data Structure specifically to store integer data using a Singly Linked List.
* The data members should be private.
* You need to implement the following public functions:
  1. Constructor:
* It initialises the data members as required.
  1. push(data) :
* This function should take one argument of type integer. It pushes the element into the stack and returns nothing.
  1. pop() :
* It pops the element from the top of the stack and in turn, returns the element being popped or deleted. In case the stack is empty, it returns -1.
  1. top :
* It returns the element being kept at the top of the stack. In case the stack is empty, it returns -1.
  1. size() :
* It returns the size of the stack at any given instance of time.
  1. isEmpty() :
* It returns a boolean value indicating whether the stack is empty or not.
* Operations Performed on the Stack:
* Query-1(Denoted by an integer 1): Pushes an integer data to the stack.
* Query-2(Denoted by an integer 2): Pops the data kept at the top of the stack and returns it to the caller.
* Query-3(Denoted by an integer 3): Fetches and returns the data being kept at the top of the stack but doesn't remove it, unlike the pop function.
* Query-4(Denoted by an integer 4): Returns the current size of the stack.
* Query-5(Denoted by an integer 5): Returns a boolean value denoting whether the stack is empty or not.

**Input Format**

* The first line contains an integer 'q' which denotes the number of queries to be run against each operation in the stack.
* Then the test cases follow.
* Every 'q' lines represent an operation that needs to be performed.
* For the push operation, the input line will contain two integers separated by a single space, representing the type of the operation in integer and the integer data being pushed into the stack.
* For the rest of the operations on the stack, the input line will contain only one integer value, representing the query being performed on the stack.

**Output Format**

* For Query-1, you do not need to return anything.
* For Query-2, prints the data being popped from the stack.
* For Query-3, prints the data kept on the top of the stack.
* For Query-4, prints the current size of the stack.
* For Query-5, prints 'true' or 'false'(without quotes).
* Output for every query will be printed in a separate line.

**Sample Input 0**

6

1 13

1 47

4

5

2

3

**Sample Output 0**

2

false

47

13

//SOURCE CODE

import java.io.\*;

import java.util.\*;

public class Solution{

public static void main(String[] args){

Scanner sc=new Scanner (System.in);

Stackusingll s=new Stackusingll();

int n=sc.nextInt();

for (int i=0;i<n;i++){

int query=sc.nextInt();

if(query==1){

int val=sc.nextInt();

s.push(val);

}

else if(query==2){

System.out.println(s.pop());

}

else if(query==3){

System.out.println(s.peak());

}

else if(query==4){

System.out.println(s.len());

}

else if(query==5){

System.out.println(s.isEmpty());

}

}

}

}

class Stackusingll{

int len=0;

Node top;

Stackusingll(){

top=null;

}

class Node{

int data;

Node next;

Node(int n){

data=n;

next=null;

}

}

public void push(int val){

Node n=new Node(val);

if(top==null){

top=n;

len++;

}

else{

n.next=top;

top=n;

len++;

}

}

public int pop(){

if(len>0){

Node temp=top;

top=top.next;

len--;

return temp.data;

}

else

return -1;

}

public int peak(){

if (len>0)

return top.data;

else

return -1;

}

public int len(){

return len;

}

public boolean isEmpty(){

return top==null;

}

}