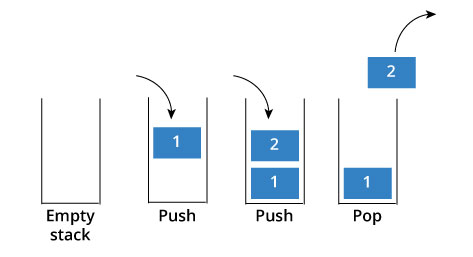
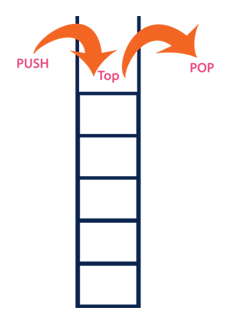
**STACK data structure**

**Stack** is a data structure in which the element removed from the set by the DELETE operation is prespecified. In a stack, the element deleted from the set is the one most recently inserted, i.e. the element that is pushed at the end is popped out first. The stack implements a last-in, first-out, or **LIFO**, mechanism.

The **INSERT**operation on a stack is often called **PUSH**, and the **DELETE** operation is often called **POP**.

* **Push** - This adds a data value to the top of the stack.
* **Pop** - This removes the data value on top of the stack

As shown in Figure, we can implement a stack of at most *n* elements with an array *S* [1..*n*]. The array has an attribute *top*[*S*] that indexes the most recently inserted element. The stack consists of elements *S*[1..*top*[*S*]], where *S*[1] is the element at the bottom of the stack and *S*[*top*[*S*]] is the element at the top.

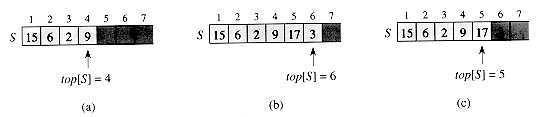


Figure. An array implementation of a stack S. Stack elements appear only in the lightly shaded positions. (a) Stack S has 4 elements. The top element is 9. (b) Stack S after the calls PUSH(S, 17) and PUSH(S, 3). (c) Stack S after the call POP(S) has returned the element 3, which is the one most recently pushed. Although element 3 still appears in the array, it is no longer in the stack; the top is element 17.

When *top* [*S*] = 0, the stack contains no elements and is ***empty***. The stack can be tested for emptiness by the query operation **STACK-EMPTY**. If an empty stack is popped, we say the stack ***underflows*,** which is normally an error. If *top*[*S*] exceeds *n*, the stack ***overflows***.

**STACK-EMPTY(*S*)**

1 **if** *top* [*S*] = 0

2 **then return** TRUE

3 **else return** FALSE

**PUSH(*S, x*)**

1 *top*[*S*] http://staff.ustc.edu.cn/~csli/graduate/algorithms/images/arrlt12.gif *top* [*S*] + 1

2 *S* [*top*[*S*]] http://staff.ustc.edu.cn/~csli/graduate/algorithms/images/arrlt12.gif *x*

**POP(*S*)**

1 **if** **STACK-EMPTY(*S*)**

2 **then error** "underflow"

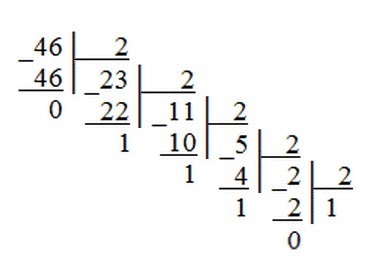
3 **else** *top* [*S*] http://staff.ustc.edu.cn/~csli/graduate/algorithms/images/arrlt12.gif *top* [*S*] - 1

4 **return** *S* [*top* [*S*] + 1]

**Applications of stacks**

**- Stacks can be used for Conversion from one form of expression to another.**

**(for example, from decimal to binary) (46)10 -> (101110)2**



**- Call and return process (**when a function is called, for example functions F1 and F2)

void main ()

**{---- |Local variables |**

**F1 -> --- for F2 |Parameters |**

**F2 |Return address** (Program Counter "Register" value)|

**--- |Local variables |**

**---- for F1 |Parameters |**

**} |Return address** (Program Counter "Register" value)|