

11.1-2.

Explain how to implement two stacks in one array $A[1..n]$ in such a way that neither stack overflows unless the total number of elements in both stacks together is n . The PUSH and POP operation should run in $O(1)$ time.

Answer.

As Figure 1 shows, we can implement two stacks S_1 and S_2 of total at most n elements in one

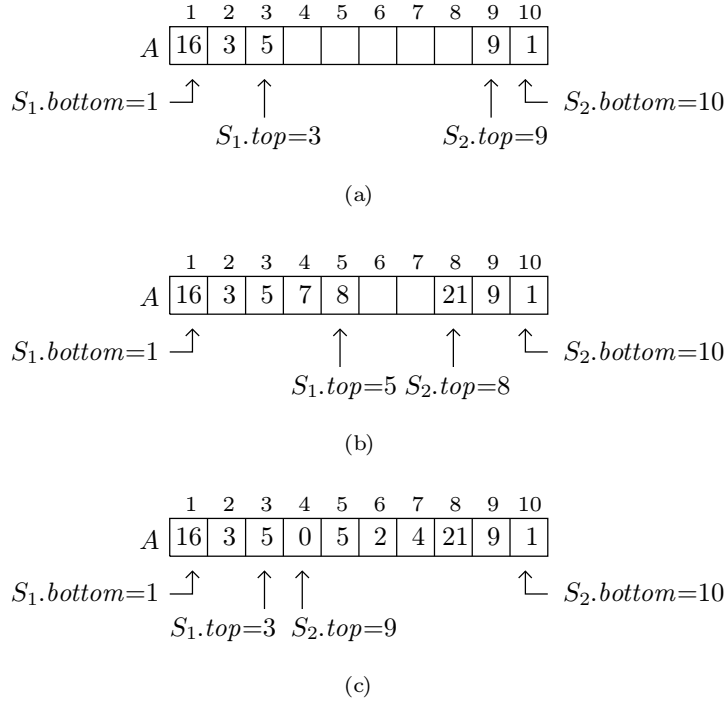



Figure 1. An array implementation of 2 stacks S_1 and S_2 . **(a)** Stack S_1 has 3 elements, with top element 5. Stack S_2 has 2 elements, with top element 9. **(b)** Stack S_1 and S_2 after the calls PUSH(S_1 , 7), PUSH(S_1 , 8) and PUSH(S_2 , 21). **(c)** Stack S_1 and S_2 after the call POP(S_1), POP(S_1), PUSH(S_2 , 4), PUSH(S_2 , 2), PUSH(S_2 , 5) and PUSH(S_2 , 0). Both stacks become full.

array $A[1..n]$. The stack S_1 consists of elements $A[1..S_1.top]$, where $A[1]$ is the element at the bottom and $A[S_1.top]$ at the top. On the contrary, the stack S_2 of elements $A[S_2.top..n]$ takes $A[n]$ as its bottom and $A[S_2.top]$ at the top.

When $S_1.top=0$, the stack S_1 contains no element and is empty. Likewise, the stack S_2 is said to be empty whenever $S_2.top=n+1$. We can test to see whether the stack is empty by the query operation STACK-EMPTY. If we attempt to pop an empty stack, then it underflows. Both stacks become full when their top elements are adjacent to each other. If an element is pushed to any one of them, then the one get augmented overflows. (In our pseudocode implementation, we don't worry about stack overflow.)

We can implement each of the stack operations with just a few lines of code:

```
STACK-EMPTY( $S$ )
1  if  $S.bottom == 1$            //  $S$  is Stack  $S_1$ 
```

*. Creative Commons  2014, Lawrence X. Amlord (颜世敏, aka 颜序).
Email address: informlarry@gmail.com

```

2      if  $S.top == 0$ 
3          return TRUE
4      else return FALSE
5  else //  $S$  is Stack  $S_2$ 
6      if  $S.top == n + 1$ 
7          return TRUE
8      else return FALSE

```

PUSH(S, x)

```

1  if  $|S_1.top - S_2.top| == 1$ 
2      error "overflow"
3  else
4      if  $S.bottom == 1$ 
5           $S.top = S.top + 1$ 
6           $A[S.top] = x$ 
7      else
8           $S.top = S.top - 1$ 
9           $A[S.top] = x$ 

```

POP(S)

```

1  if STACK-EMPTY( $S$ )
2      error "underflow"
3  else
4      if  $S.bottom == 1$ 
5           $S.top = S.top - 1$ 
6          return  $A[S.top + 1]$ 
7      else
8           $S.top = S.top + 1$ 
9          return  $A[S.top - 1]$ 

```