IMPLEMENTING A STACK WITH AN ARRAY (50/50 points)

In this exercise, we will encode imperative stacks of `int`s using the `type stack = int array` as defined in the prelude.

We will use the first cell (cell `0`) to store the number of items in the stack. Then the cells of the array starting from `1` will be used to store the elements in the stack. The bottom element of the stack will be stored at position `1`.

The stack will thus have a maximum capacity, being the length of the array minus one.

- An empty stack of capacity `4` would match the following pattern:
  `[| 0 ; _ ; _ ; _ ; _|]`
- A stack of capacity `4` containing `1` at the bottom, then `2` and then `3` at the top would match the following pattern:
  `[| 3 ; 1 ; 2 ; 3 ; _|]`

1. Define a function `create : int -> stack` that creates a new stack of the given maximum capacity.
2. Define a function `push : stack -> int -> unit` that adds an element as the top of the stack.
   The function must fail with the exception `Full` given in the prelude if nothing can be added to the stack.
3. Define a function `append : stack -> int array -> unit` that adds an array of integers as the top of the stack. The first element of the array must be at the top of the stack, the others in order, up to the last element, which will be the lowest in the stack. In other words, the last element of the array should be pushed first, etc.
   The function must fail with the exception `Full` given in the prelude if some elements could not fit in the stack. But in this case, it should still fill the stack with as many elements as possible.
4. Define a function `pop : stack -> int` that takes an element as the top of the stack, removes it from the stack, and return it.
   The function must fail with the exception `Empty` given in the prelude if nothing can be taken from the stack.

THE GIVEN PRELUDE

```
type stack = int array
exception Full
exception Empty
```

YOUR OCAML ENVIRONMENT

Rechercher un cours

```ocaml
  let long = Array.length buf in
  let pos = buf.(0) in
  if pos + 1 < long then buf.(pos + 1) <- elt else raise Full;
  buf.(0) <- pos + 1
;;

let append buf arr =
  let long = Array.length arr in
  for i = long - 1 downto 0 do
    push buf arr.(i)
  done
;;

let pop buf =
  let pos = buf.(0) in
  if pos = 0 then raise Empty else
    let result = buf.(pos) in
    buf.(0) <- pos - 1;
    buf.(pos) <- 0;
    result
;;



```

Switch >>

Typechecl

Reset Templ

Full-screen |

Check & Sa

```
Exercise complete (click for details)                                    50 pts
v Exercise 1: create                                          Completed, 10 pts
  Found create with compatible type.
  Computing create 9
  Correct value [|0; 0; 0; 0; 0; 0; 0; 0; 0; 0|]                           1 pt
  Computing create 8
  Correct value [|0; 0; 0; 0; 0; 0; 0; 0; 0|]                              1 pt
  Computing create 4
  Correct value [|0; 0; 0; 0; 0|]                                          1 pt
  Computing create 4
  Correct value [|0; 0; 0; 0; 0|]                                          1 pt
  Computing create 5
  Correct value [|0; 0; 0; 0; 0; 0|]                                       1 pt
  Computing create 2
  Correct value [|0; 0; 0|]                                                1 pt
  Computing create 8
  Correct value [|0; 0; 0; 0; 0; 0; 0; 0; 0|]                              1 pt
  Computing create 5
  Correct value [|0; 0; 0; 0; 0; 0|]                                       1 pt
  Computing create 6
  Correct value [|0; 0; 0; 0; 0; 0; 0|]                                    1 pt
  Computing create 8
  Correct value [|0; 0; 0; 0; 0; 0; 0; 0; 0|]                              1 pt
v Exercise 2: push                                            Completed, 10 pts
  Found push with compatible type.
  Computing push [|2; -4; 4|] 9
  Correct exception Full                                                   1 pt
  Computing push [|3; -1; -1; -3; 0; 0; 0|] 0
  Correct value [|4; -1; -1; -3; 0; 0; 0|]                                 1 pt
  Computing push [|0|] 1
  Correct exception Full                                                   1 pt
  Computing push [|2; -3; 4; 0|] 2
  Correct value [|3; -3; 4; 2|]                                            1 pt
  Computing push [|1; 3; 0; 0|] 6
  Correct value [|2; 3; 6; 0|]                                             1 pt
  Computing push [|4; 2; 1; -1; -2|] 6
  Correct exception Full                                                   1 pt
  Computing push [|2; 0; -5; 0; 0; 0|] 8
  Correct value [|3; 0; -5; 8; 0; 0|]                                      1 pt
  Computing push [|1; -3|] 1
  Correct exception Full                                                   1 pt
  Computing push [|3; -5; -5; -1|] 5
  Correct exception Full                                                   1 pt
```

```
v Exercise 3: append                                              Completed, 10 pts
Found append with compatible type.
Computing append [|1; 1|] [||]
Correct value [|1; 1|]                                                         1 pt
Computing append [|4; 3; -3; 4; -4; 0; 0|] [|6; 2; 8; 5|]
Correct exception Full                                                         1 pt
Computing append [|4; 2; 0; 0; 2; 0; 0; 0; 0; 0|] [|8; 1; 7; 5; 0|]
Correct value [|9; 2; 0; 0; 2; 0; 5; 7; 1; 8|]                                 1 pt
Computing append [|3; -3; -4; -5; 0; 0; 0|] [|2; 3; 5|]
Correct value [|6; -3; -4; -5; 5; 3; 2|]                                       1 pt
Computing append [|2; -3; -1; 0|] [|3; 7; 5; 2|]
Correct exception Full                                                         1 pt
Computing append [|0; 0; 0; 0; 0; 0|] [|2; 9; 1|]
Correct value [|3; 1; 9; 2; 0; 0|]                                             1 pt
Computing append [|0|] [|4; 9|]
Correct exception Full                                                         1 pt
Computing append [|3; 0; -3; 1; 0; 0; 0; 0; 0|] [|4; 5; 7|]
Correct value [|6; 0; -3; 1; 7; 5; 4; 0; 0|]                                   1 pt
Computing append [|3; -1; 4; -5; 0|] [|3; 6; 1|]
Correct exception Full                                                         1 pt
Computing append [|4; 4; 1; 1; 2; 0; 0; 0; 0; 0|] [|7; 2; 6; 3; 4|]
Correct value [|9; 4; 1; 1; 2; 4; 3; 6; 2; 7|]                                 1 pt
v Exercise 4: pop                                                 Completed, 20 pts
Found pop with compatible type.
Computing pop [|3; -1; 3; -3; 0; 0; 0; 0|]
Correct value -3                                                               1 pt
Correct value [|2; -1; 3; 0; 0; 0; 0; 0|]                                      1 pt
Computing pop [|0; 0|]
Correct exception Empty                                                        1 pt
Correct exception Empty                                                        1 pt
Computing pop [|0; 0; 0; 0; 0|]
Correct exception Empty                                                        1 pt
Correct exception Empty                                                        1 pt
Computing pop [|1; -3|]
Correct value -3                                                               1 pt
Correct value [|0; 0|]                                                         1 pt
Computing pop [|0; 0; 0; 0; 0|]
Correct exception Empty                                                        1 pt
Correct exception Empty                                                        1 pt
Computing pop [|3; 3; -5; -5; 0; 0; 0; 0|]
Correct value -5                                                               1 pt
Correct value [|2; 3; -5; 0; 0; 0; 0; 0|]                                      1 pt
Computing pop [|2; -3; 0; 0|]
Correct value 0                                                                1 pt
Correct value [|1; -3; 0; 0|]                                                  1 pt
Computing pop [|0; 0; 0; 0|]
Correct exception Empty                                                        1 pt
Correct exception Empty                                                        1 pt
Computing pop [|3; -5; -4; 2; 0|]
Correct value 2                                                                1 pt
Correct value [|2; -5; -4; 0; 0|]                                              1 pt
Computing pop [|1; -1; 0; 0|]
Correct value -1                                                               1 pt
Correct value [|0; 0; 0; 0|]                                                   1 pt
```