Quiz Section for Program Design (I)

Exercise #9

Tino is a CCU student passionate about programming design. Recently, he learned a new concept—Pointers. He wants to use pointers to implement a simple version of a **stack** by function, printing the operations during the *push* and *pop* actions, and finally printing all the elements in the stack starting from *top*.
He has decided to seek your help. Please assist Tino in completing his stack implementation!

A **stack** is a data structure that works like a stack of plates: you add (*push*) and remove (*pop*) items from the *top*. It follows the LIFO rule, meaning Last In, First Out.

Examples (assuming the right side of the stack is the top.):

- ❖ Stack [1, 2] → Push 3 → Stack [1, 2, 3].
- ❖ Stack [1, 2, 3] → Pop → Stack [1, 2].
- ❖ Stack [1, 2, 3] → Top is 3.

- **Input Format**

  The first line contains two integers `S` and `N`. `S` determines the max size of the stack. The following `N` lines contain an integer `M`. `M` determines the stack to operate *pop* (`M`=0) or *push* (`M`=1) action. If `M` is 1 (push action), then the following integer `V` determines the number that is pushed to the stack.

- **Output Format**
  - ○ Print the action and the number that pushed or popped from the stack.
  - ○ Print the error message (see example output), if the stack is full/empty when operating push/pop respectively.
  - ○ Finally, print all the remaining elements in the stack starting from the top.
- **Technical Specifications**
  - ○ **You should not use any global variable**, which means you have to declare all the variables in the function (or main function).
  - ○ $1 \leq S, N \leq 2000$
  - ○ $M \in \{0, 1\}$
  - ○ $-2^{31} \leq V \leq 2^{31} - 1$

Notice: You need to complete this assignment based on the example code.

```c
#include <stdio.h>

int is_full(int *index, const int size){
    //determine whether the stack is full or not.
}
int is_empty(int *index){
    //determine whether the stack is empty or not.
}
void push(int *stack,int *index, const int size, const int num){
    //if the stack is not full, push the element to the stack.
}
void pop(int *stack, int *index){
    //if the stack is not empty, pop the top of element from the stack.
}

int main(){
    int S,N;
    scanf("%d %d", &S, &N);
    int stack[S], index = 0;
    //...
}
```

Input	Output

4 4
1 2
1 3
0
1 5
	push 2
	push 3
	pop 3
	push 5
	The all stack elements are:
	5 2

2 4
0
1 6
1 7
1 9
	No element left!
	push 6
	push 7
	The stack is already full!
	The all stack elements are:
	7 6