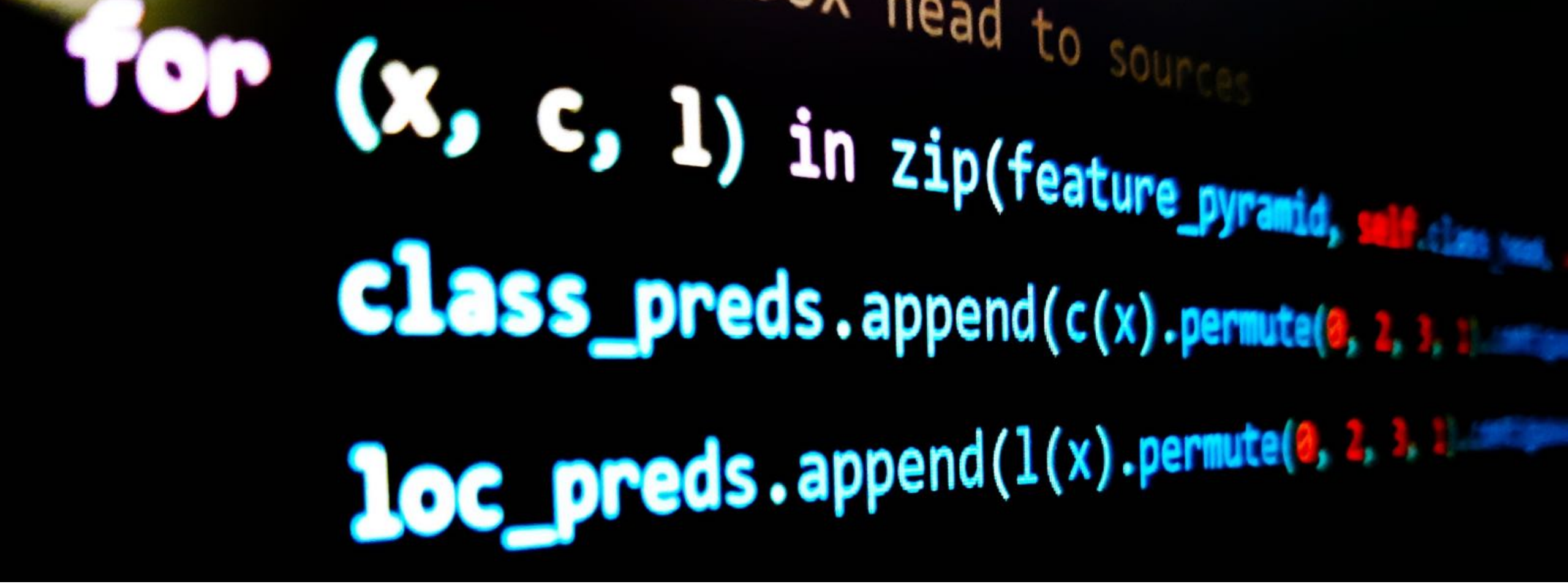


STACKS

CCDSALG T2 AY 2020-2021

STACKS

- An ordered list in which all insertions and deletions are made at one end called TOP.
- Only the top element is accessible
- Implements a last-in, first-out (LIFO) policy.

TOP of the stack (of books) ➡

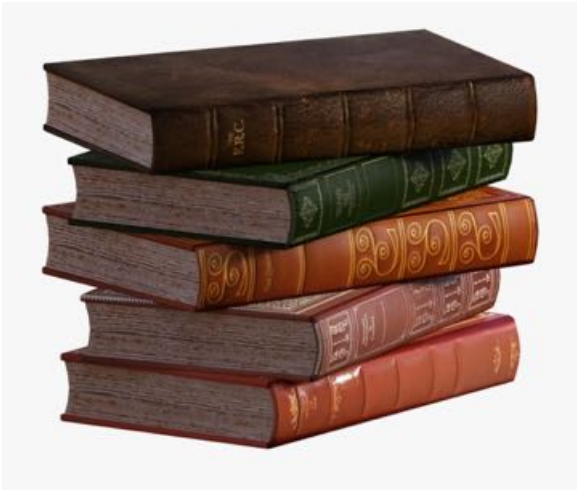


Image credits

Image credits

⬅ **TOP** of the stack (of plates)

CONVENTIONS

- The pointer TOP points to the most recently inserted element in the stack.

OPERATIONS

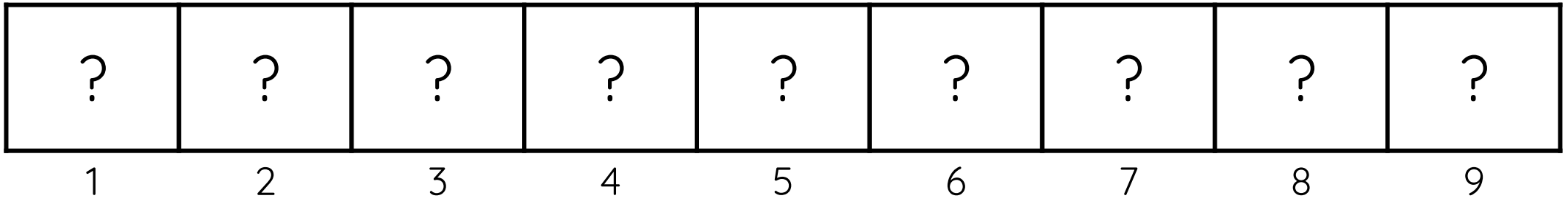
- **CREATE(STACK)** – creates an empty stack.
- **PUSH(STACK, ITEM)** – inserts an element item into the stack (INSERT operation).
- **POP(STACK)** – removes and then returns the top element of the stack (DELETE operation).
- **TOP(STACK)** – returns the top element of the stack.

OPERATIONS

- **STACK_EMPTY(STACK)** – determines whether the stack is empty or not.
  - If $TOP = 0$, stack is empty.
- **STACK_FULL(STACK)** – determines whether the stack is full or not.
  - If $TOP = n$, stack is full.

SIMULATION

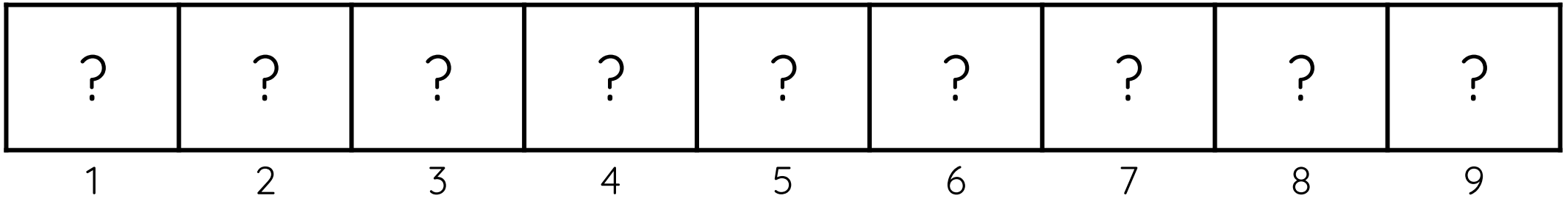
1. **CREATE(S)** will produce



with the value of $TOP = 0$.

SIMULATION

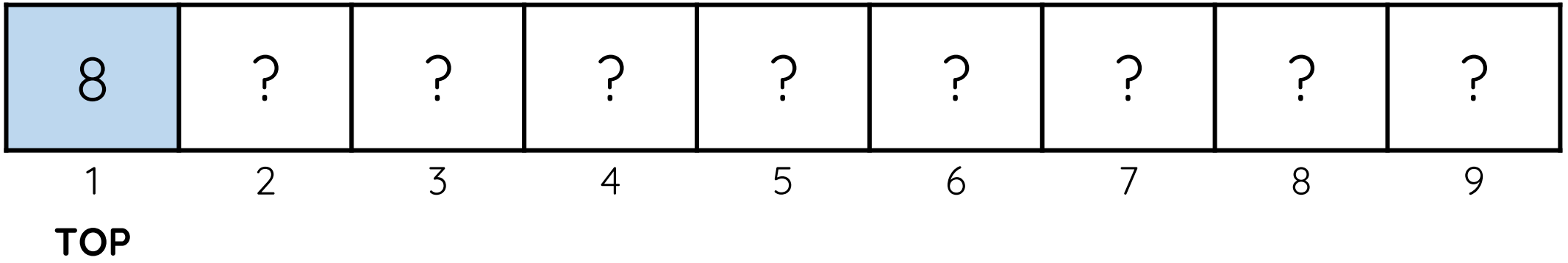
2. Stack S after **STACK_EMPTY(S)**.



STACK_EMPTY(S) will return true since the stack is indeed empty (no elements).

SIMULATION

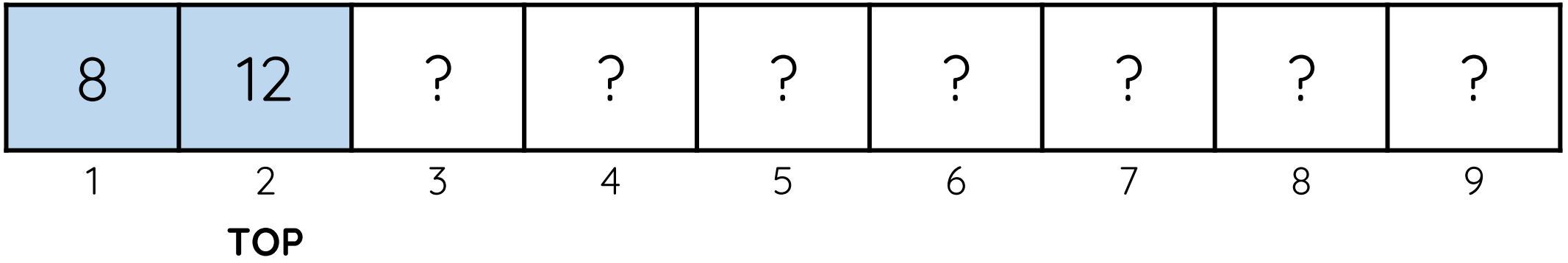
3. Stack S after **PUSH(S, 8).**



The top element is 8 at index 1.

SIMULATION

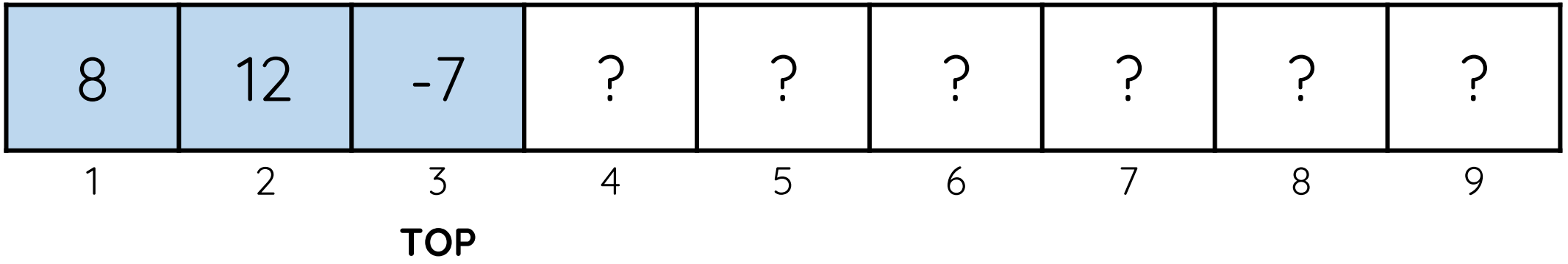
4. Stack S after **PUSH(S, 12).**



The top element is 12 at index 2.

SIMULATION

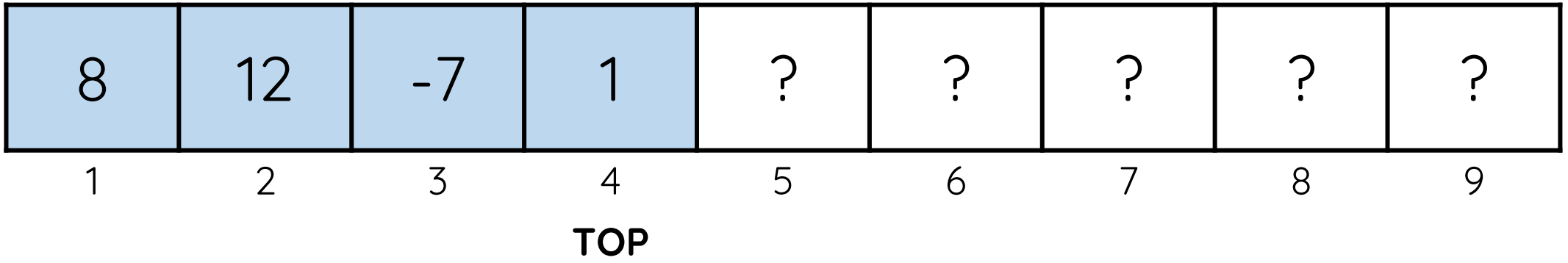
5. Stack S after **PUSH(S, -7).**



The top element is −7 at index 3.

SIMULATION

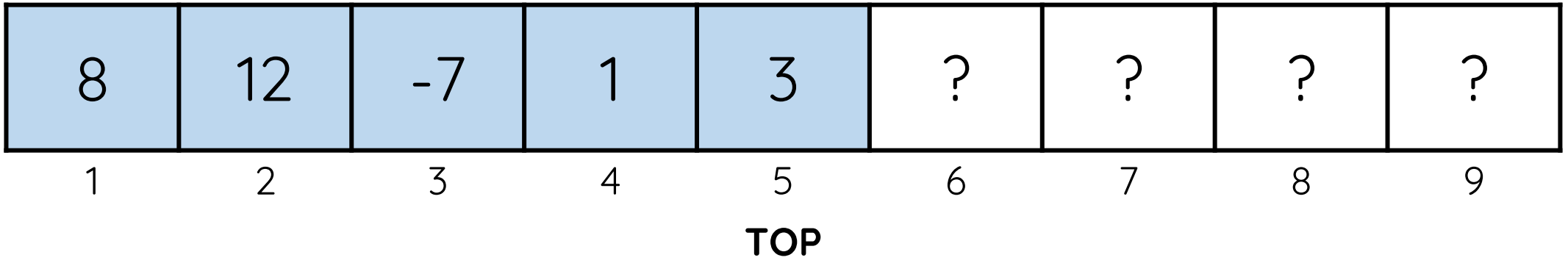
6. Stack S after **PUSH(S, 1).**



The top element is 1 at index 4.

SIMULATION

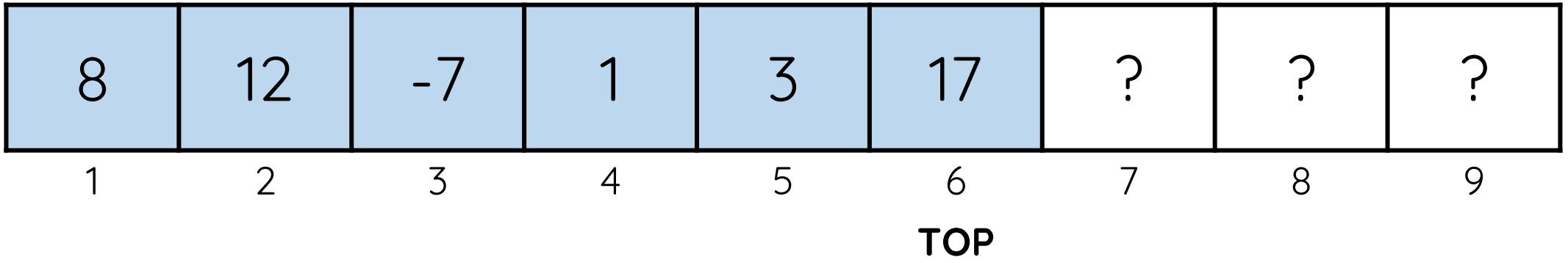
7. Stack S after **PUSH(S, 3).**



The top element is 3 at index 5.

SIMULATION

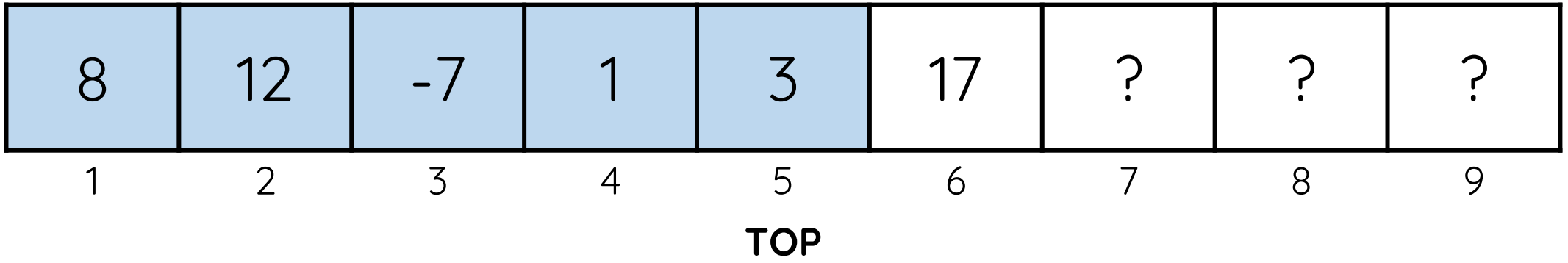
8. Stack S after **PUSH(S, 17).**



The top element is 17 at index 6.

SIMULATION

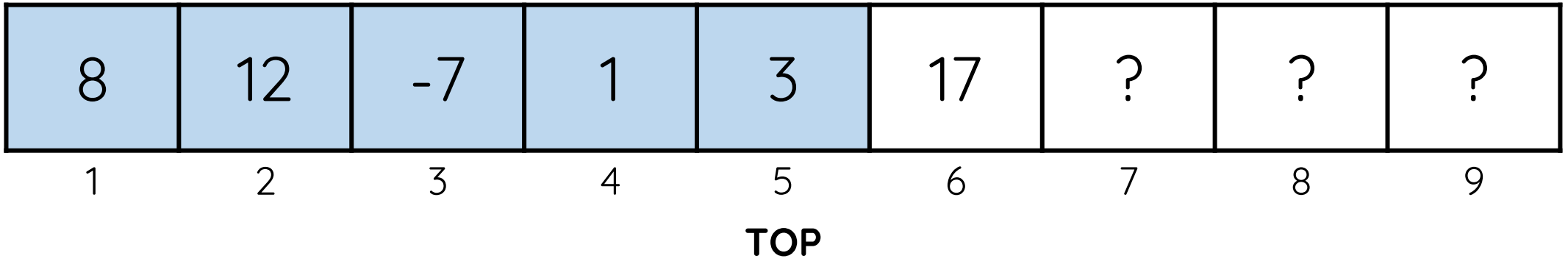
9. Stack S after **POP(S).**



POP(S) will return **17**, which is the most recently pushed element. Although **17** is still in the array, it is no longer in the stack and the new top element is **3**.

SIMULATION

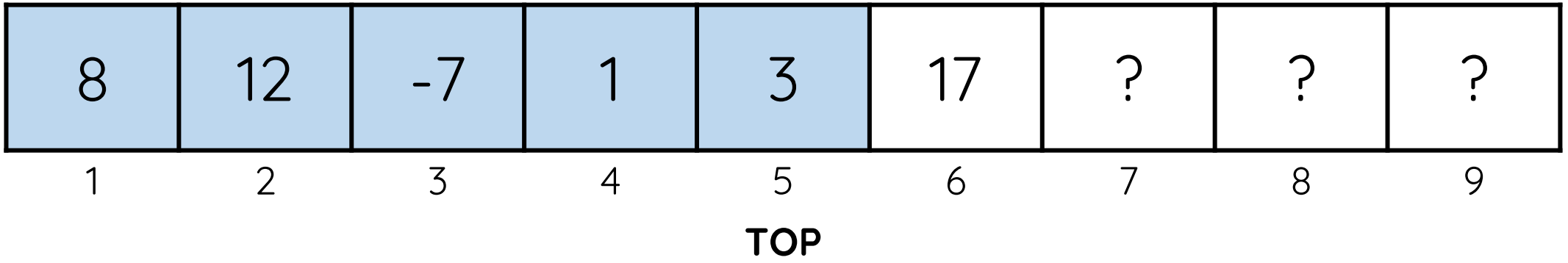
10. Stack S after **TOP(S).**



TOP(S) will return **3**, the top element of the stack. It will not delete **3** from the stack.

SIMULATION

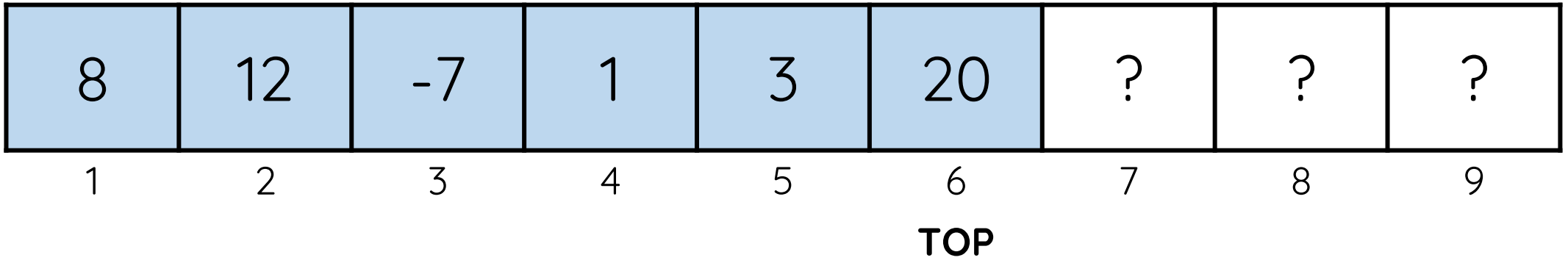
11. Stack S after **STACK_EMPTY(S).**



STACK_EMPTY(S) will return false since the stack is not empty.

SIMULATION

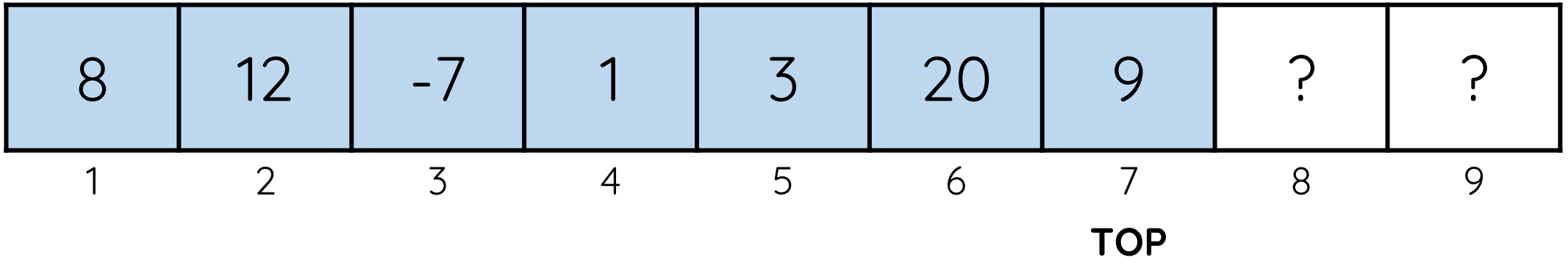
12. Stack S after **PUSH(S, 20).**



The top element is 20 at index 6.

SIMULATION

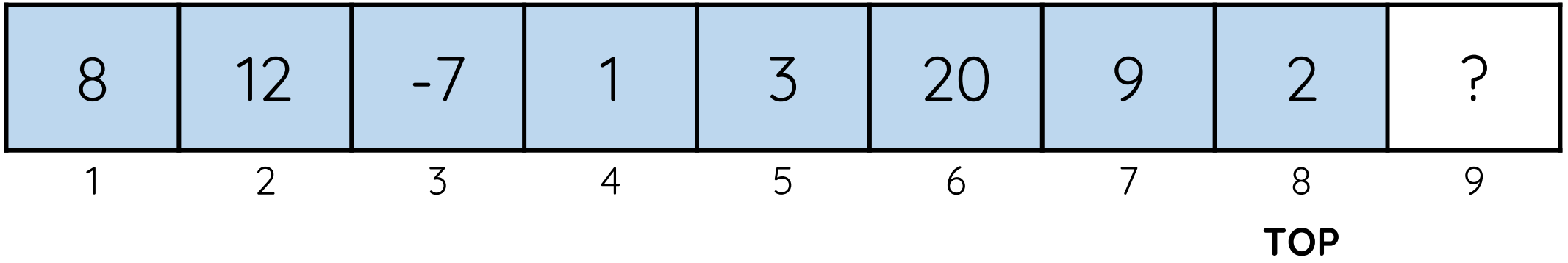
13. Stack S after **PUSH(S, 9).**



The top element is 9 at index 7.

SIMULATION

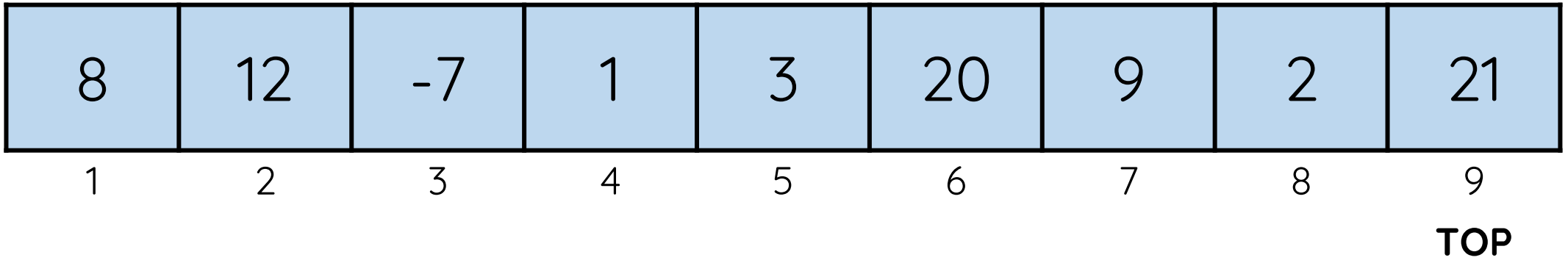
14. Stack S after **PUSH(S, 2).**



The top element is 2 at index 8.

SIMULATION

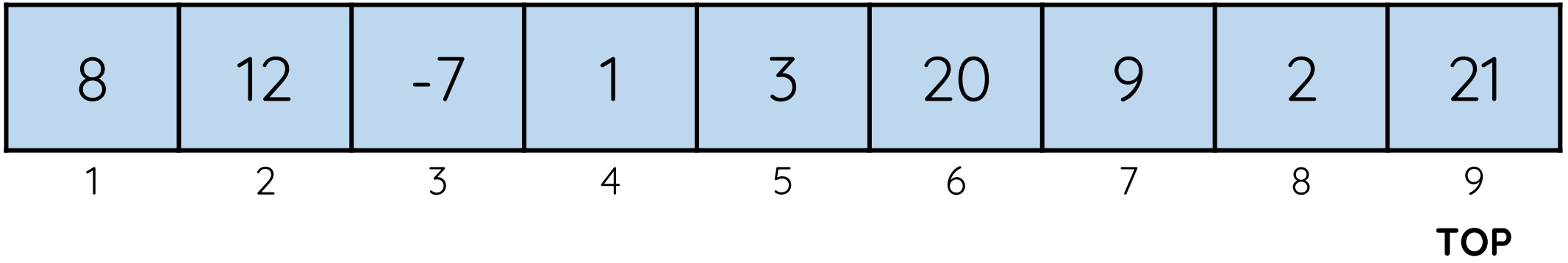
15. Stack S after **PUSH(S, 21).**



The top element is 21 at index 9.

SIMULATION

16. Stack S after **STACK_FULL(S).**



STACK_FULL(S) will return true since TOP points to the topmost index of the stack ($TOP = n$).

POSSIBLE ERRORS

- **Underflow** – occurs when an empty stack is popped.
- **Overflow** – occurs when pushing an element to a fully-filled stack

OPERATIONS DEFINED

```
CREATE(STACK) {
    TOP = 0;
}
```

OPERATIONS DEFINED

```
STACK_EMPTY(STACK) {
    if(TOP == 0)
        return 1; // true
    else
        return 0; // false
}
```

OPERATIONS DEFINED

```
STACK_FULL(STACK) {
    if(TOP == n)
        return 1; // true
    else
        return 0; // false
}
```

OPERATIONS DEFINED

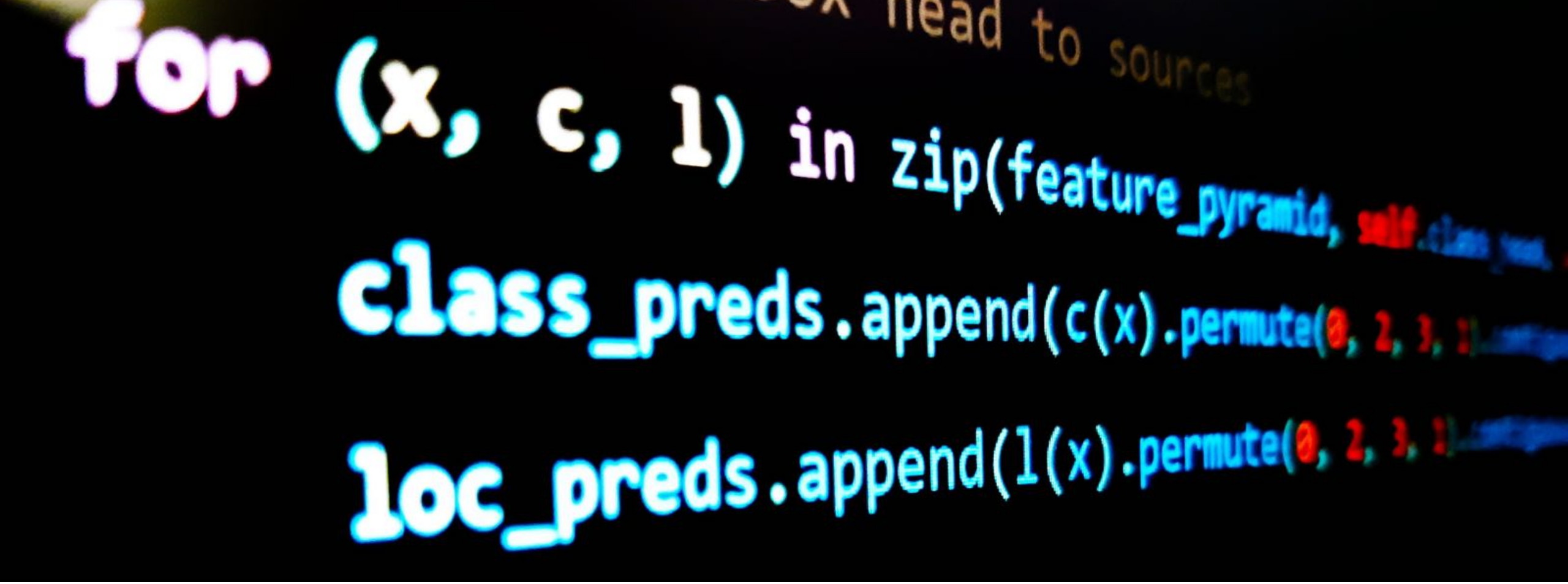
```
PUSH(STACK, x) {
    if(STACK_FULL(STACK))
        printf("Overflow error!\n");
    else {
        TOP = TOP + 1;
        STACK[TOP] = x;
    }
}
```

OPERATIONS DEFINED

```
POP(STACK) {
    if(STACK_EMPTY(STACK))
        printf("Underflow Error!\n");
    else {
        x = STACK[TOP];
        TOP = TOP - 1;
        return x;
    }
}
```

OPERATIONS DEFINED

```
TOP(STACK) {
    if(STACK_EMPTY(STACK))
        printf("Stack Empty!\n");
    else
        return STACK[TOP];
}
```



STACKS

CCDSALG T2 AY 2020-2021