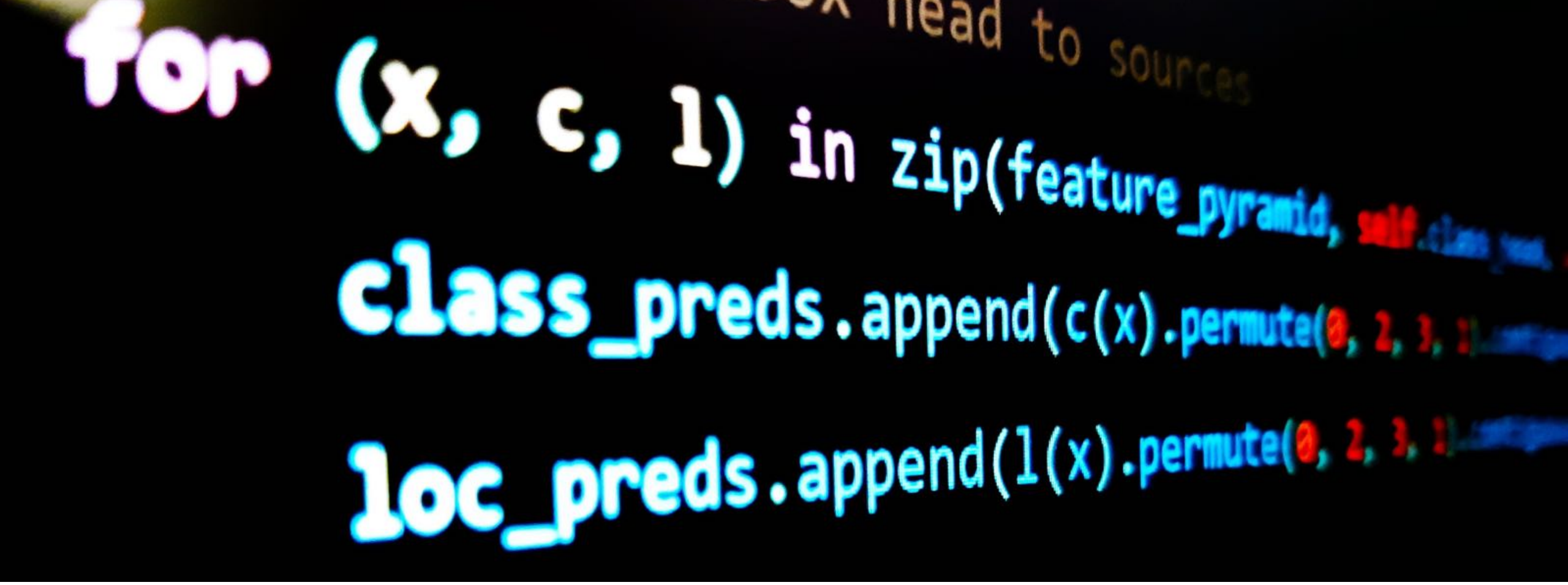


QUEUES

CCDSALG T2 AY 2020-2021

QUEUES

- An ordered list in which all insertions take place at one end called the TAIL, while all deletions take place at the other end, called the HEAD.
- Implements a first-in, first-out (FIFO) policy. The first element inserted will be the first one to be removed.

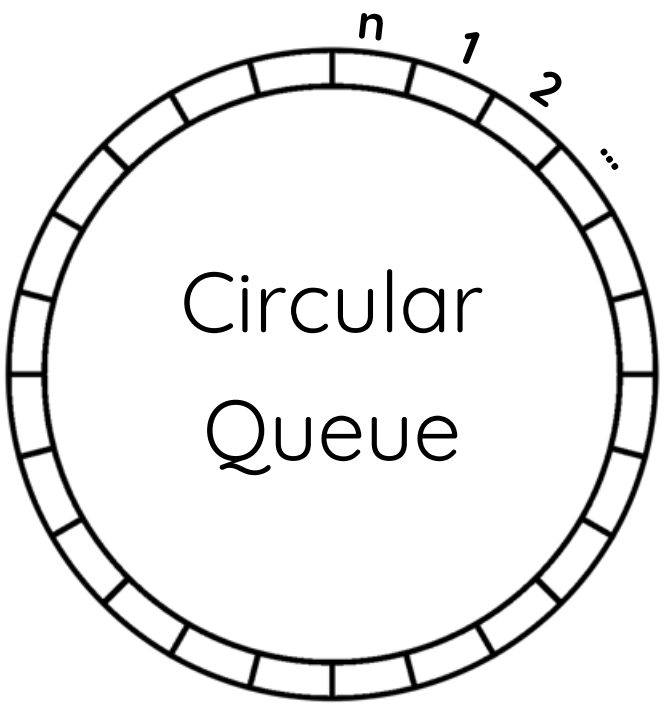
HEAD (front) of the queue

TAIL (rear) of the queue

Image credits

CONVENTIONS

- HEAD is used to index or point to the head of the queue.
- TAIL is used to index or point to the next location at which a newly arriving element will be inserted into the queue.
- For circular queues, index 1 immediately follows location n.



OPERATIONS

- **CREATE(QUEUE)** – creates an empty queue.
- **ENQUEUE(QUEUE, ITEM)** – inserts an element into the queue (INSERT operation)
- **DEQUEUE(QUEUE)** – removes and then returns the head of the queue (DELETE operation)

OPERATIONS

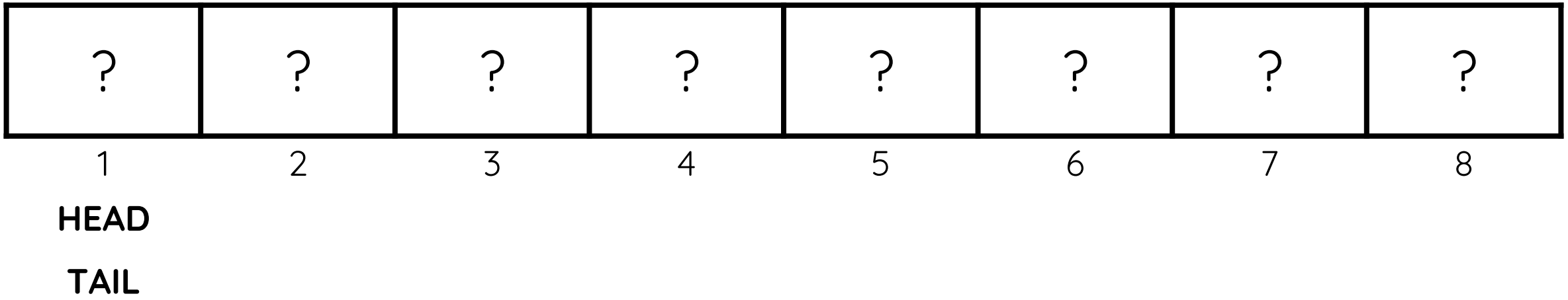
- **QUEUE_HEAD(QUEUE)** – determines the element at the head of the queue.
- **QUEUE_TAIL(QUEUE)** – determines the element at the tail of the queue.

OPERATIONS

- **QUEUE_EMPTY(QUEUE)** – determines if the queue is empty or not.
  - If $HEAD = TAIL$, queue is empty
- **QUEUE_FULL(QUEUE)** – determines if the queue is full or not.
  - If $HEAD = (TAIL + 1) \% n$, queue is full.

SIMULATION

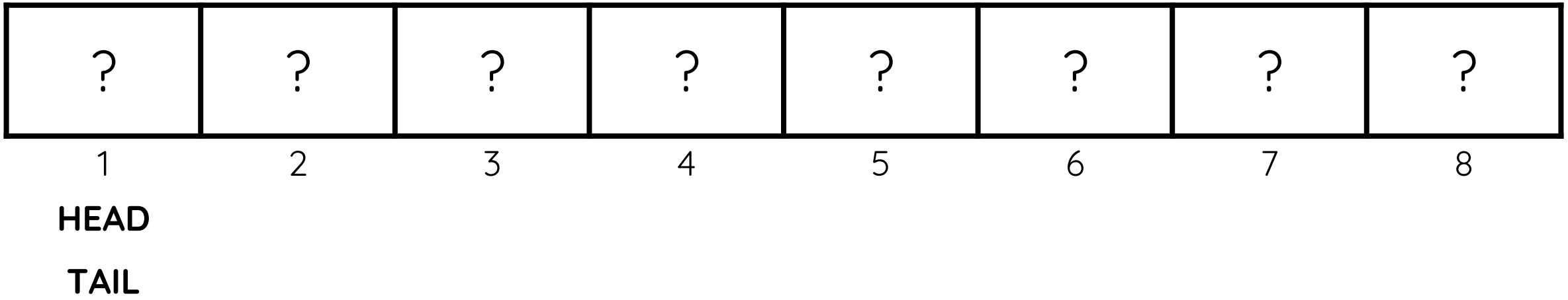
1. **CREATE(Q)** will produce



with the value of $HEAD = 1$ and $TAIL = 1$.

SIMULATION

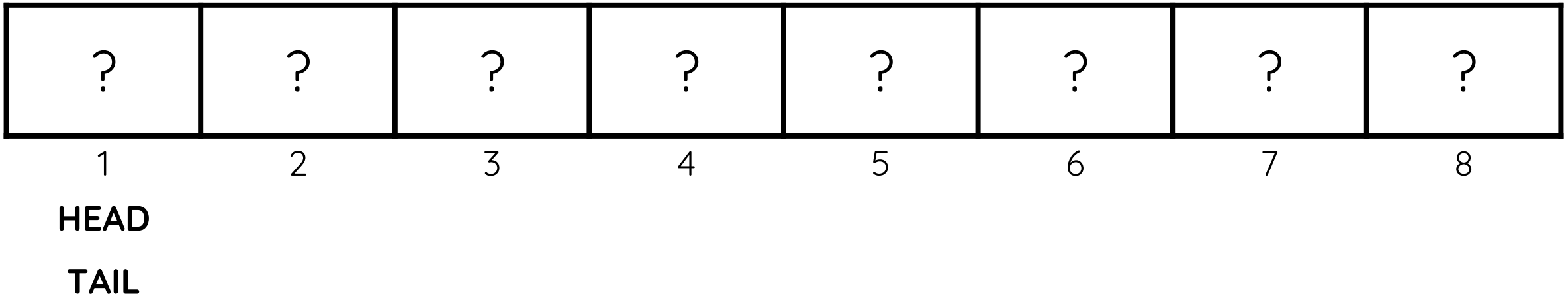
2. Queue Q after **QUEUE_EMPTY(Q).**



QUEUE_EMPTY(Q) will return true since the queue is indeed empty (no elements).

SIMULATION

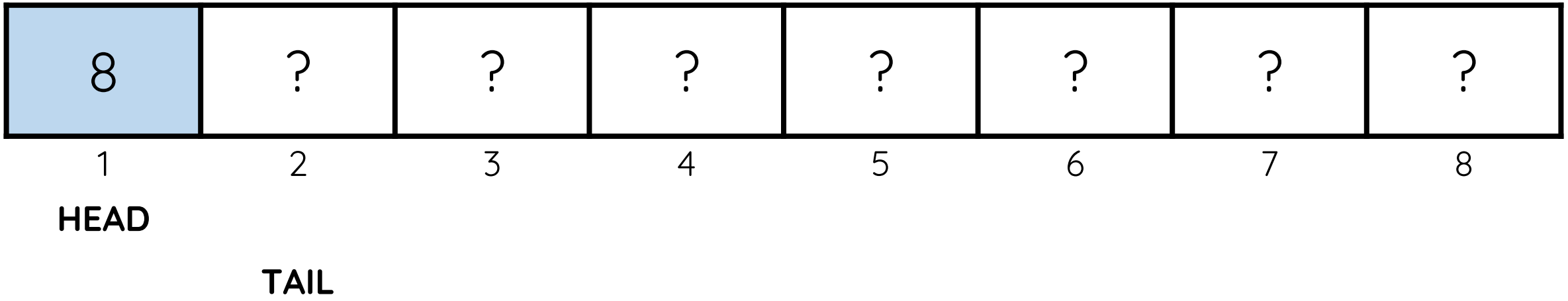
3. Queue Q after **QUEUE_FULL(Q).**



QUEUE_FULL(Q) will return false.

SIMULATION

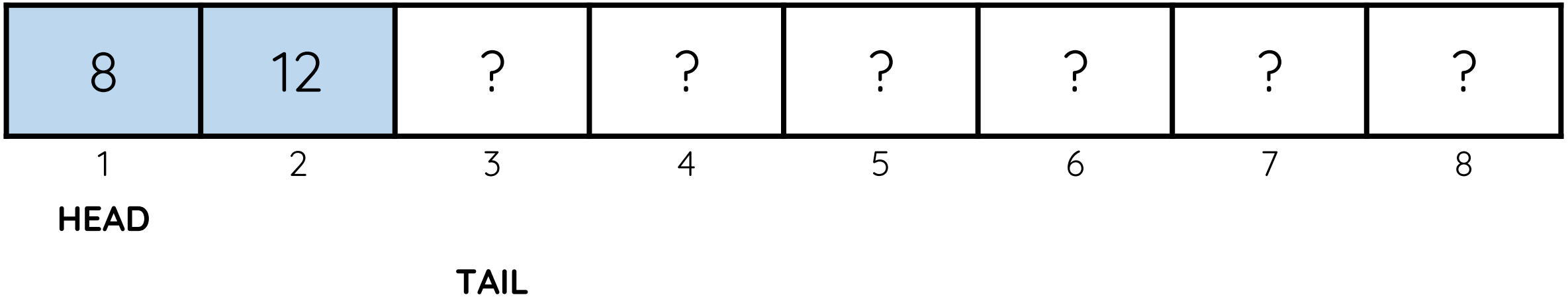
4. Queue Q after **ENQUEUE(Q, 8).**



The first element is 8.

SIMULATION

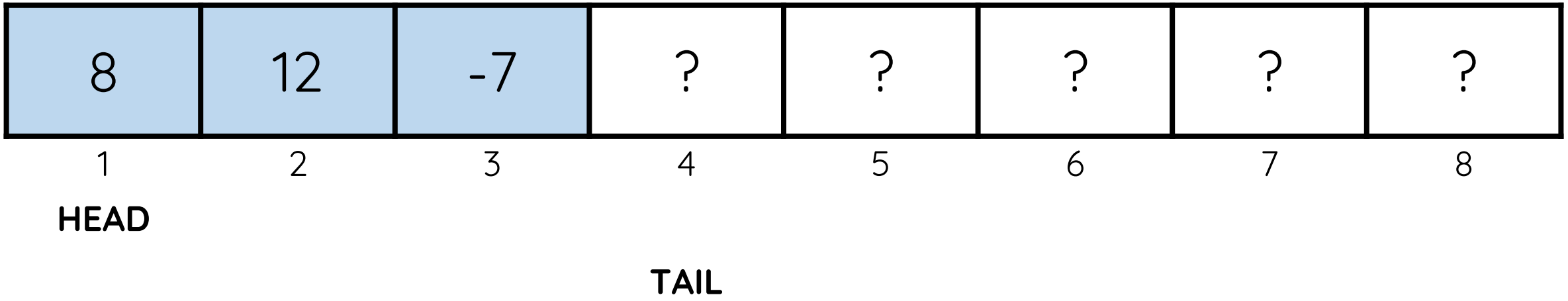
5. Queue Q after **ENQUEUE(Q, 12).**



The first element is 8 and the last element is 12.

SIMULATION

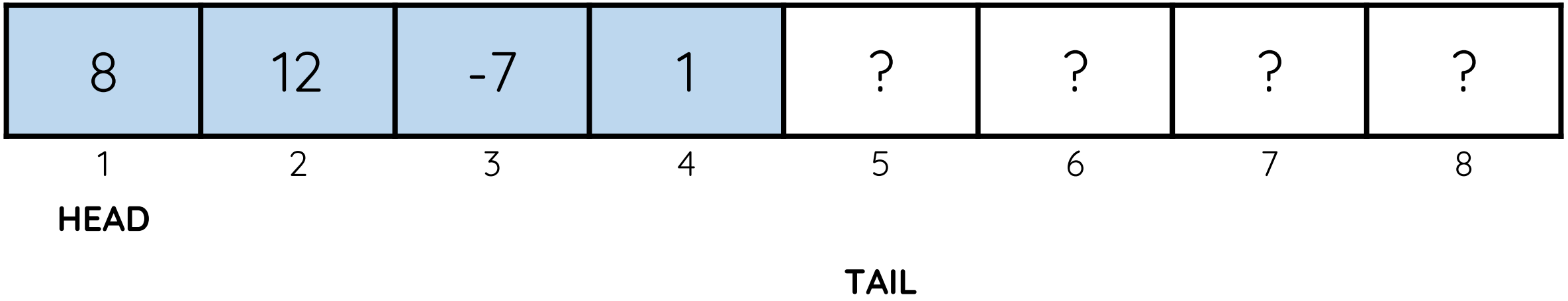
6. Queue Q after **ENQUEUE(Q, -7).**



The first element is 8 and the last element is −7.

SIMULATION

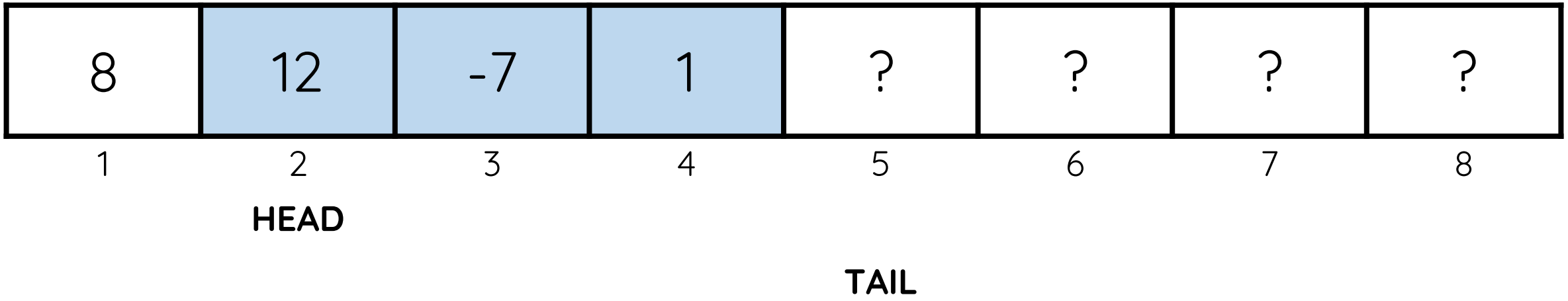
7. Queue Q after **ENQUEUE(Q, 1).**



The first element is 8 and the last element is 1.

SIMULATION

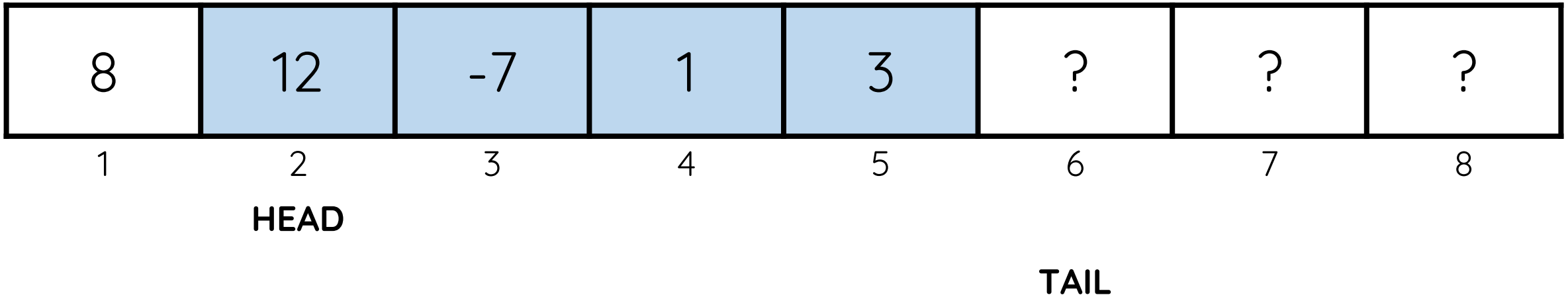
8. Queue Q after **DEQUEUE(Q).**



The first element is now 12. Though 8 is still in the queue, it is not anymore part of the queue.

SIMULATION

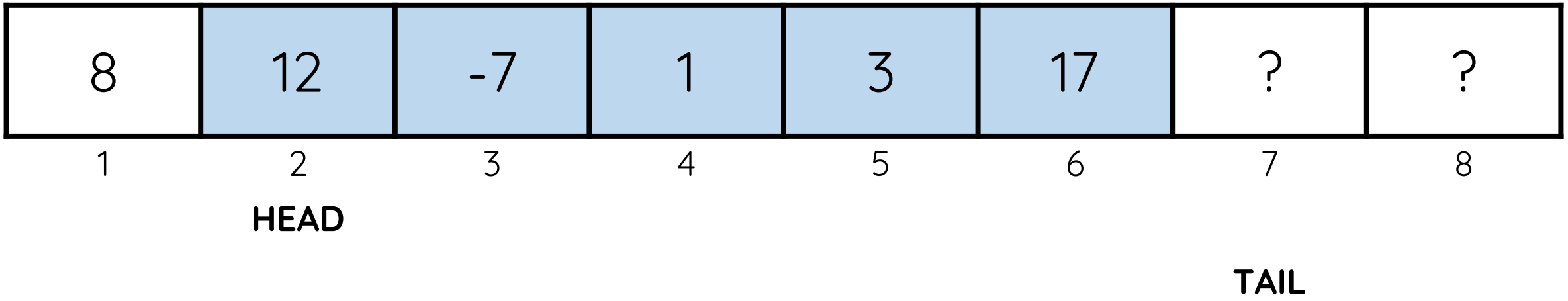
9. Queue Q after **ENQUEUE(Q, 3).**



The first element is 12 and the last element is 3.

SIMULATION

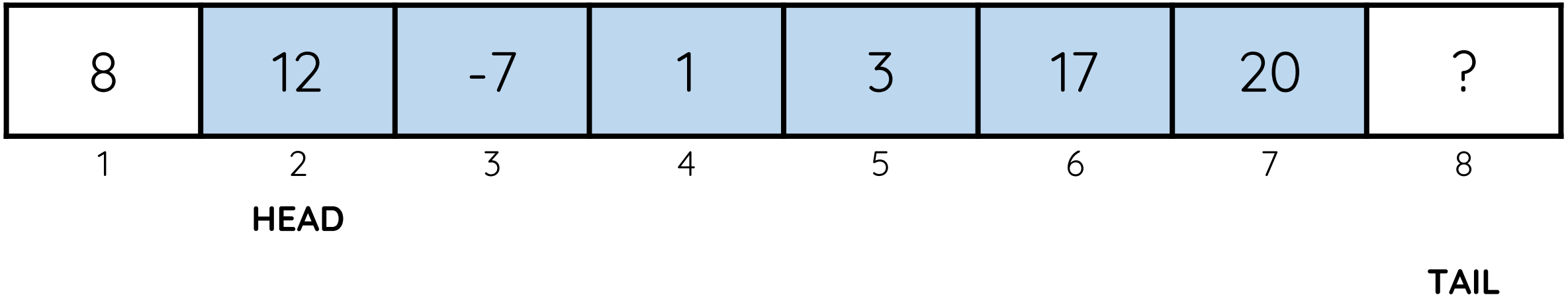
10. Queue Q after **ENQUEUE(Q, 17).**



The first element is 12 and the last element is 17.

SIMULATION

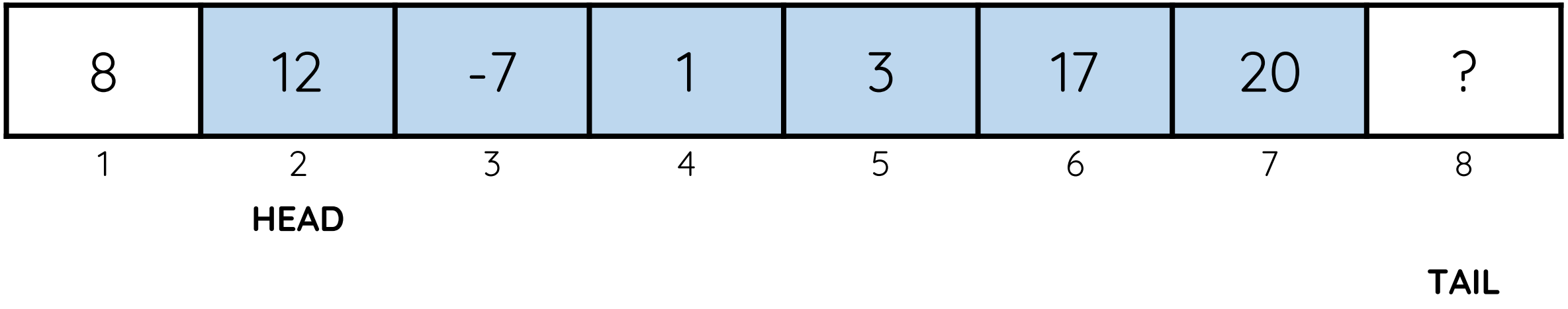
11. Queue Q after **ENQUEUE(Q, 20).**



The first element is 12 and the last element is 20.

SIMULATION

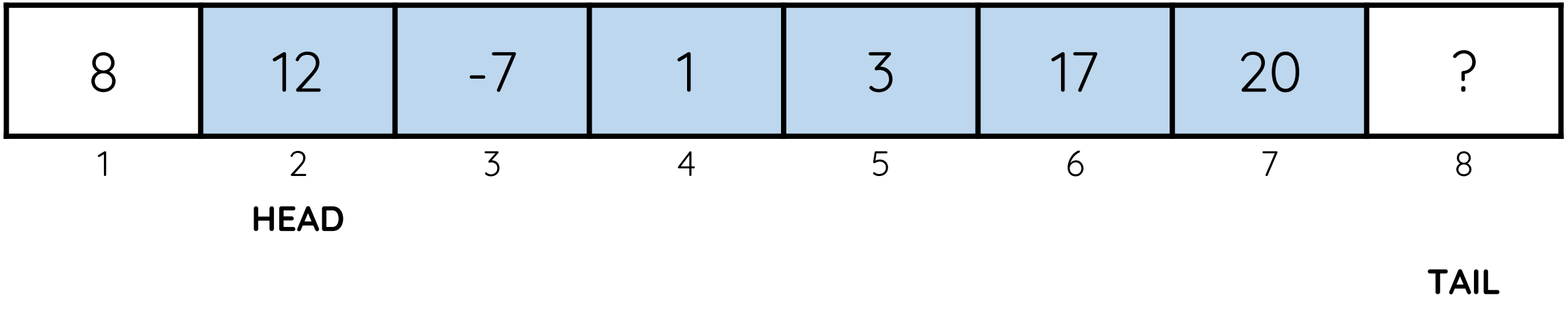
12. Queue Q after **QUEUE_EMPTY(Q).**



QUEUE_EMPTY(Q) will return false.

SIMULATION

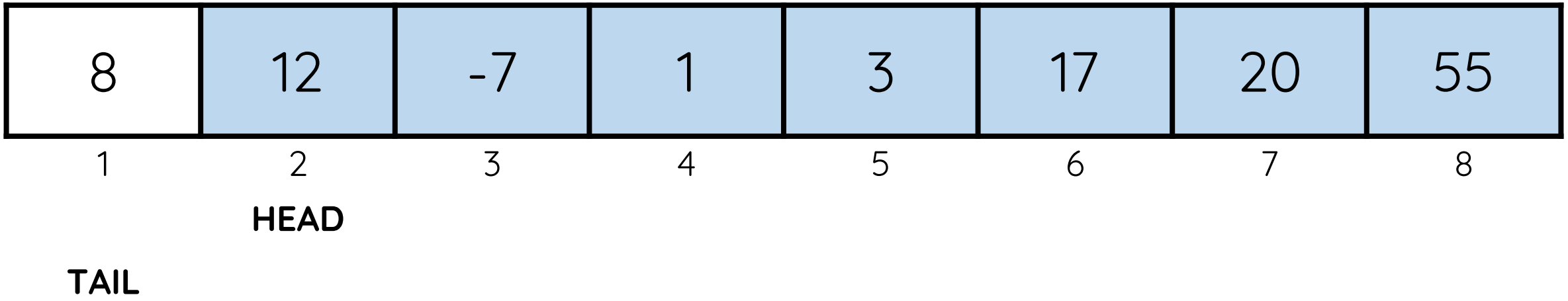
13. Queue Q after **QUEUE_FULL(Q).**



QUEUE_FULL(Q) will return false.

SIMULATION

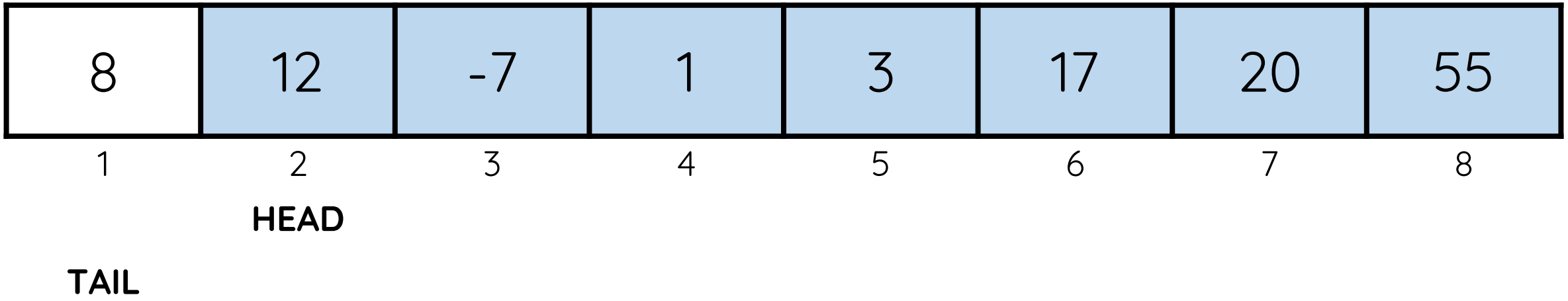
14. Queue Q after **ENQUEUE(Q, 55).**



The first element is 12 and the last element is 55.

SIMULATION

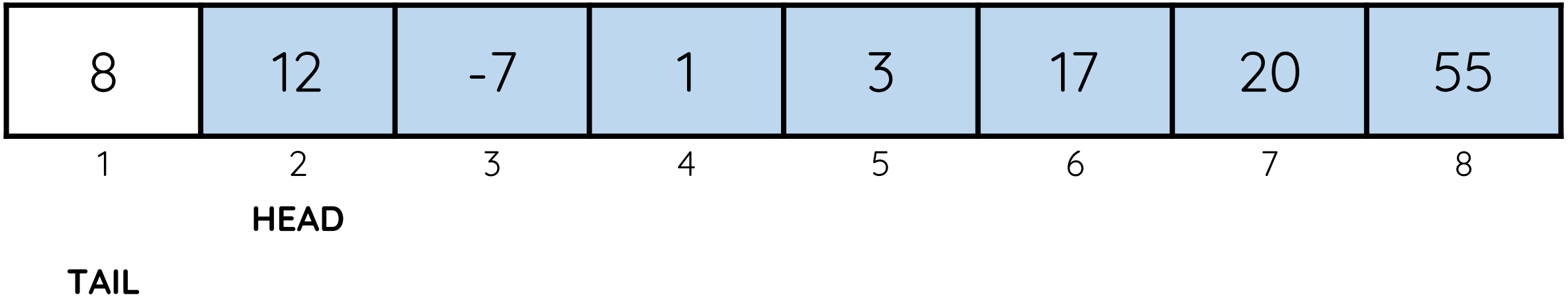
15. Queue Q after **QUEUE_EMPTY(Q)**



QUEUE_EMPTY(Q) will return false.

SIMULATION

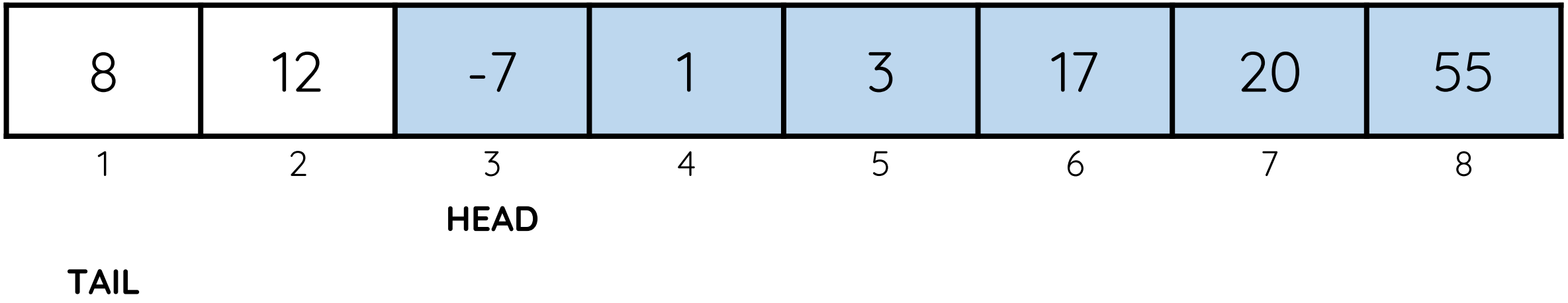
16. Queue Q after **QUEUE_FULL(Q)**



QUEUE_FULL(Q) will return true.

SIMULATION

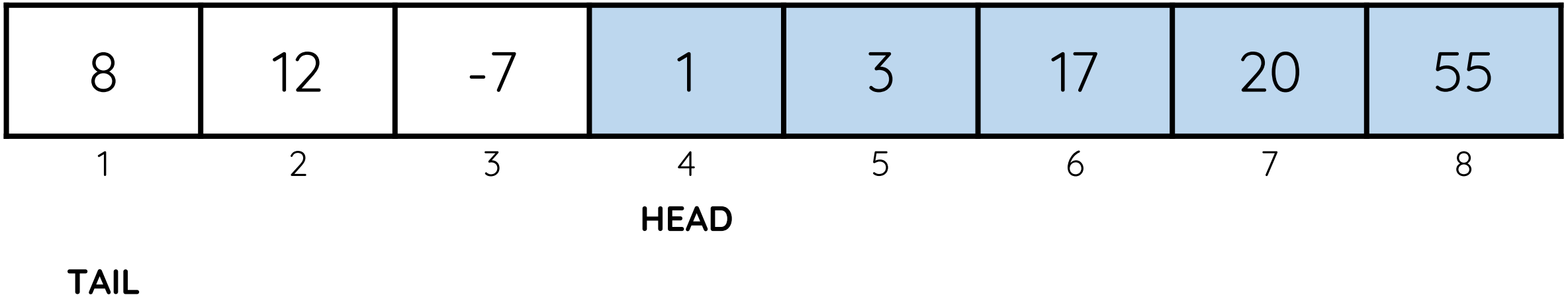
17. Queue Q after **DEQUEUE(Q)**



The first element is -7 and the last element is 55.

SIMULATION

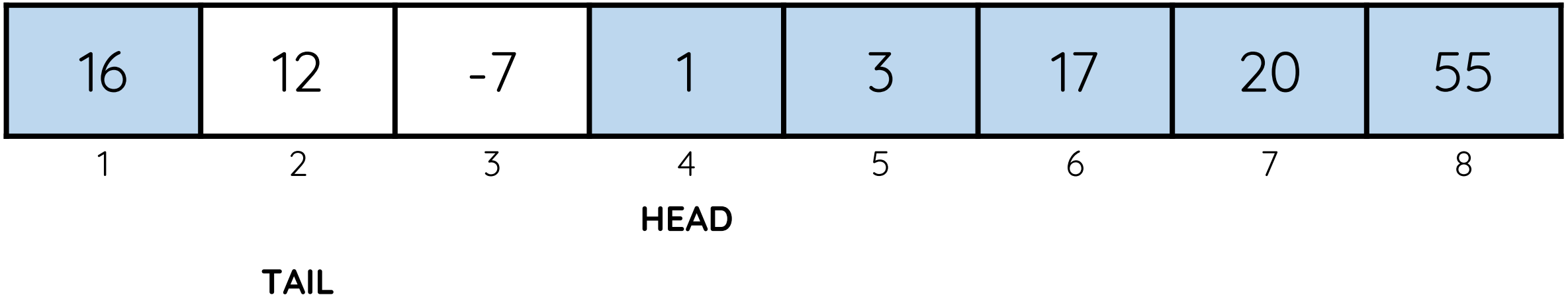
18. Queue Q after **DEQUEUE(Q)**



The first element is 1 and the last element is 55.

SIMULATION

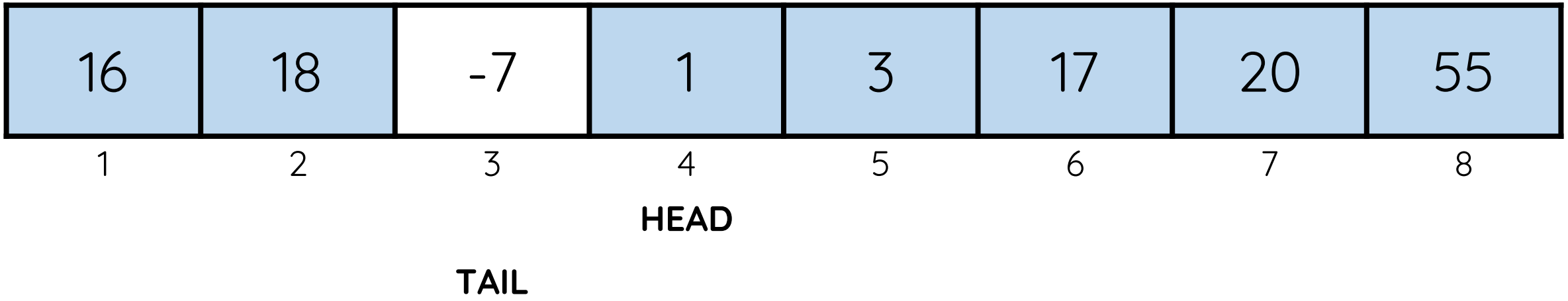
19. Queue Q after **ENQUEUE(Q, 16)**



The first element is 1 and the last element is 16.

SIMULATION

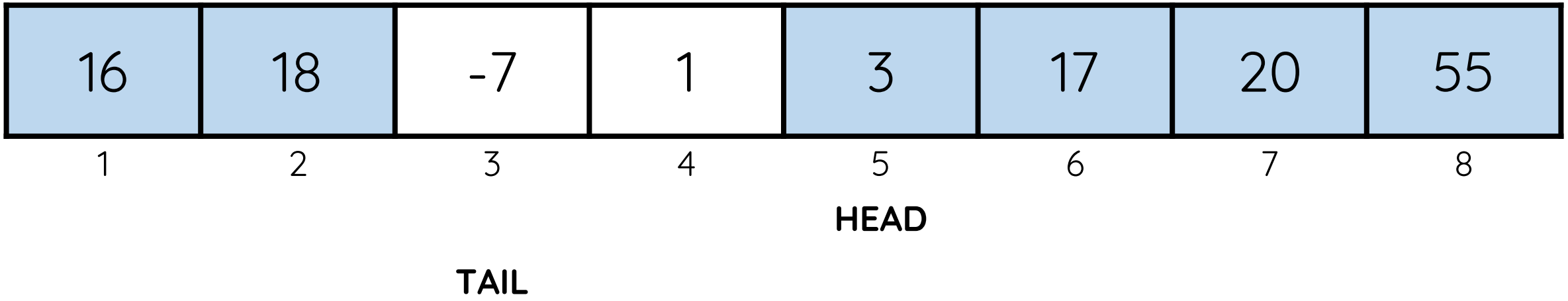
20. Queue Q after **ENQUEUE(Q, 18)**



The first element is 1 and the last element is 18.

SIMULATION

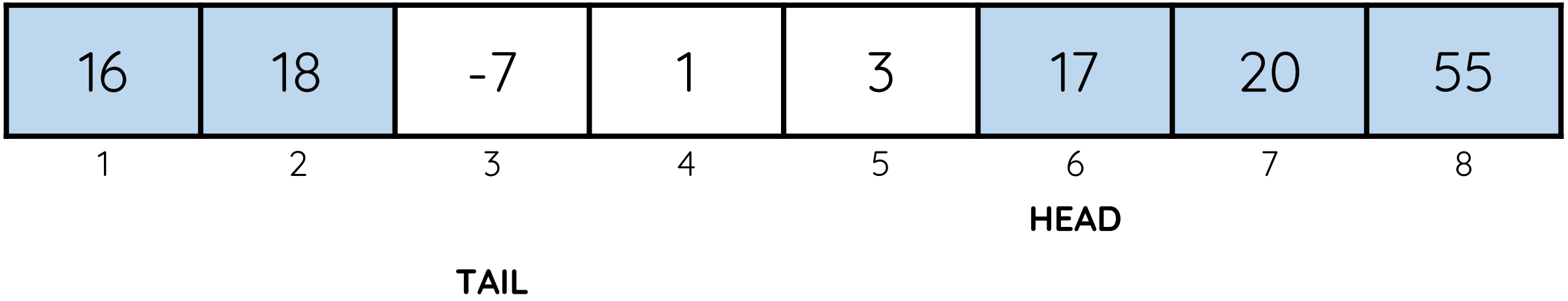
21. Queue Q after **DEQUEUE(Q)**



The first element is 3 and the last element is 18.

SIMULATION

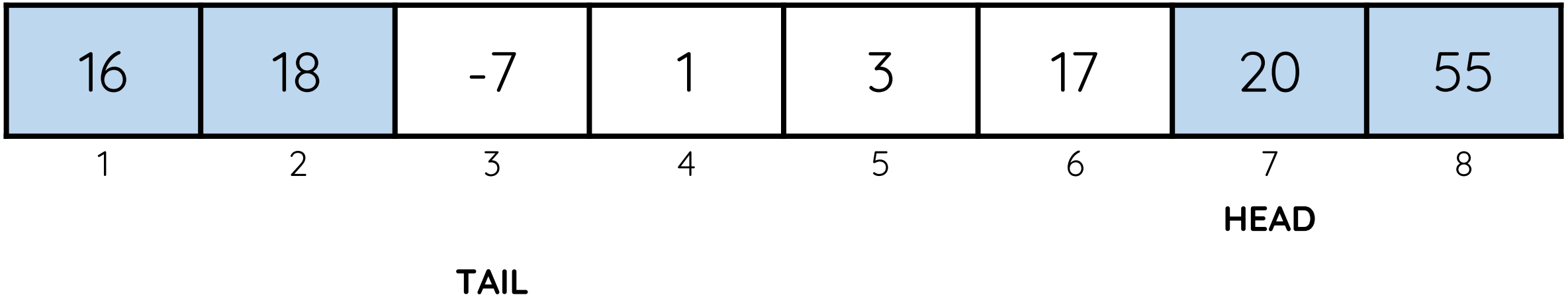
22. Queue Q after **DEQUEUE(Q)**



The first element is 17 and the last element is 18.

SIMULATION

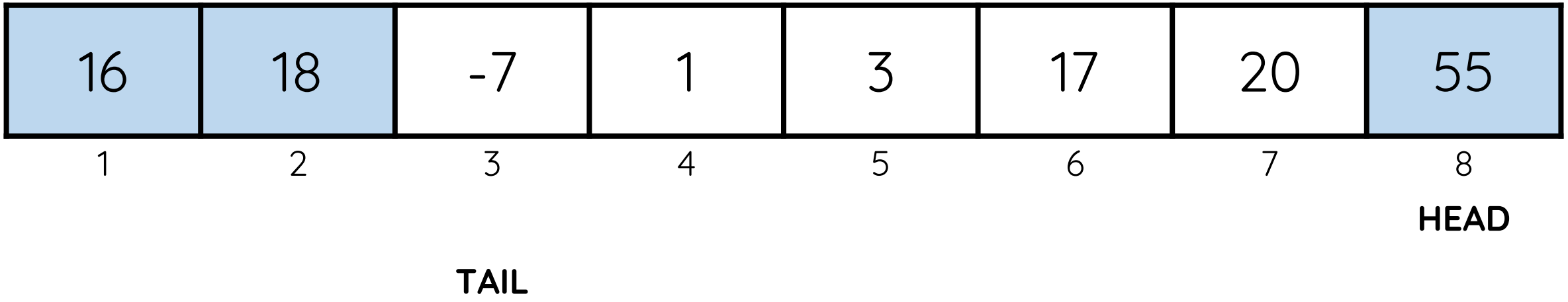
23. Queue Q after **DEQUEUE(Q)**



The first element is 20 and the last element is 18.

SIMULATION

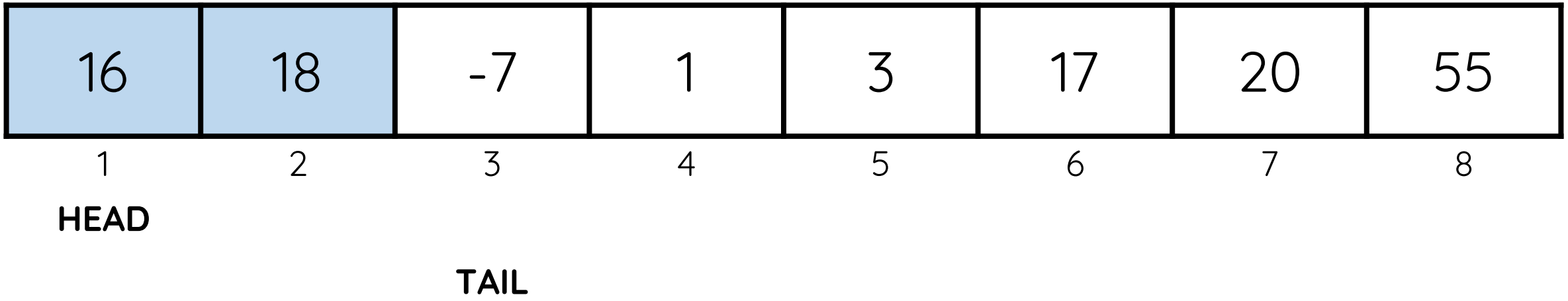
24. Queue Q after **DEQUEUE(Q)**



The first element is 55 and the last element is 18.

SIMULATION

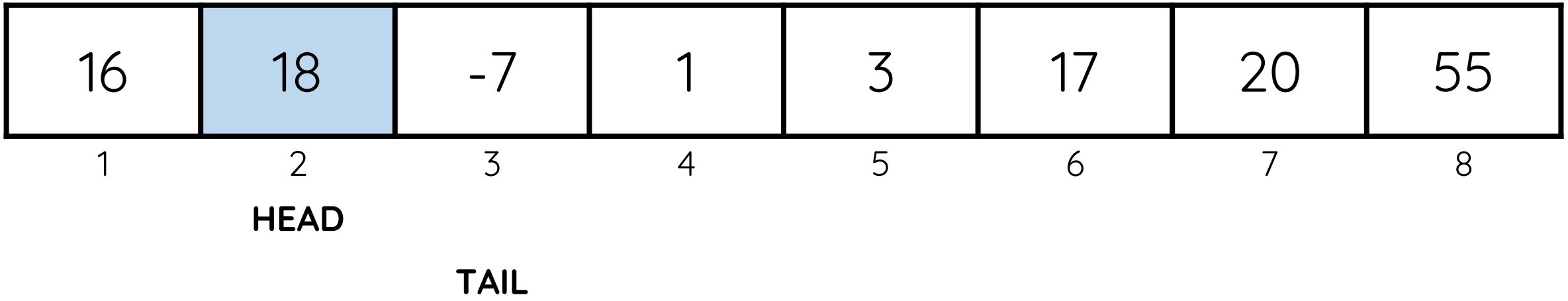
25. Queue Q after **DEQUEUE(Q)**



The first element is 16 and the last element is 18.

SIMULATION

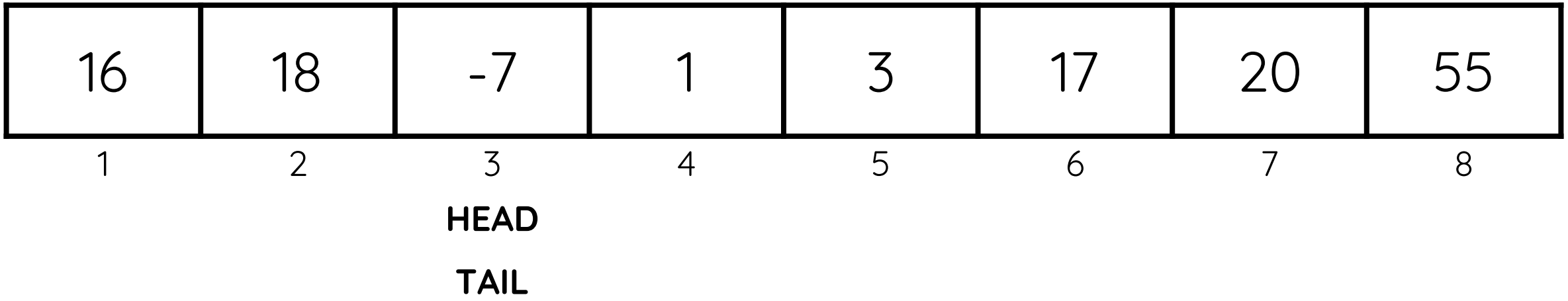
26. Queue Q after **DEQUEUE(Q)**



The first element is 18 and the last element is 18.

SIMULATION

27. Queue Q after **DEQUEUE(Q)**



The queue is empty and there is no first nor last element.

NOTES

- A queue is full if it contains $n - 1$ elements.

POSSIBLE ERRORS

- **Underflow** – occurs when an empty queue is being dequeued.
- **Overflow** – occurs when enqueueing an element to a full queue.

OPERATIONS DEFINED

```
CREATE(QUEUE) {
    HEAD = 1
    TAIL = 1
}
```

OPERATIONS DEFINED

```
QUEUE_EMPTY(QUEUE) {
    if(HEAD == TAIL)
        return 1; // true
    else
        return 0; // false
}
```

OPERATIONS DEFINED

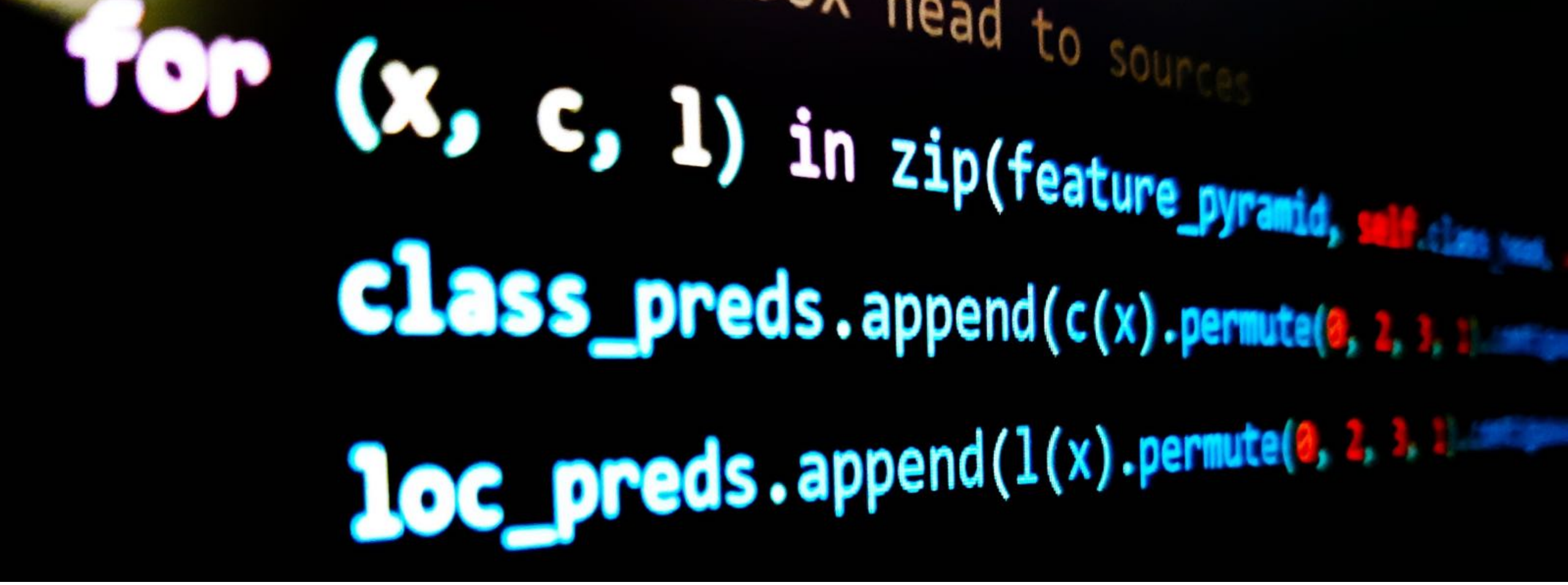
```
QUEUE_FULL(QUEUE) {
    if(HEAD == (TAIL + 1) % n)
        return 1; // true
    else
        return 0; // false
}
```

OPERATIONS DEFINED

```
ENQUEUE(QUEUE, x) {
    if(QUEUE_FULL(QUEUE))
        printf("Overflow Error!\n");
    else {
        QUEUE[TAIL] = x;
        if(TAIL == n)
            TAIL = 1;
        else
            TAIL = TAIL + 1;
    }
}
```

OPERATIONS DEFINED

```
DEQUEUE(QUEUE, x) {
    if(QUEUE_EMPTY(QUEUE))
        printf("Underflow Error!\n");
    else {
        x = QUEUE[HEAD];
        if(HEAD == n)
            HEAD = 1;
        else
            HEAD = HEAD + 1;
        return x;
    }
}
```



QUEUES

CCDSALG T2 AY 2020-2021