Assignment 1

Assignment 1: Monitor structure

```
monitor semaphore
{
  int val;
  cond q;

  int down();

  int up();
}
```

```
int down() {
  decrement val by 1
  if (val < 0)
      wait q
  return 1
}
```

Αν αφού μειωθεί το val και εξακολουθεί να είναι αρνητικό πρέπει να περιμένουμε καθώς είτε η τιμή του σηματοφόρου ήταν 0 ή υπήρχαν άλλα νήματα που ήδη περίμεναν.

```
int up() {
  // Lost up call
  if (val == 1)
    return 0

  increment val by 1
  if (val <= 0)
    signal q
  return 1
}
```

Αν αφού αυξήσουμε την τιμή του σηματοφόρου εξακολουθεί να είναι μικρότερη ή ίση με το 0, σημαίνει ότι υπάρχουν νήματα που περιμένουν και πρέπει να ξυπνίσει το επόμενο.

Assignment 1: Διαφορετικά μοντέλα monitors

- Η λύση αυτή λειτουργεί για open και για eggshell monitors. Στην χειρότερη περίπτωση του open monitor δεν υπάρχουν προβλήματα με προσπεράσεις καθώς τα logistics λειτουργούν με βάση την `val` και επιπλέον η wait είναι τελευταία εντολή στην συνάρτηση down.
- Η λύση λειτουργεί για signal-block και signal-continue αφού όλες η μοναδική κλήση `signal` στην συνάρτηση up γίνεται στο τέλος.

Assignment 2

Assignment 2: Monitor structure

```
monitor {
  int blocked_workers = 0;
  int working_workers = N;
  bool main_sleeping = false;
  bool main_waiting_worker = false;
  cond main_sleep;
  cond worker_q;

  command, value get_job();
  void set_job();
}
```

Assignment 2: Master και worker threads

Main thread

```
command = PROCESS_VALUE
WHILE (EOF is not reached) {
  Read next value input
  set_job();
}
command = TERMINATE
FOR every worker I
  set_job();
```

Worker thread

```
WHILE (1) {
  command, value = get_value();
  IF command == TERMINATE
    Break;
  is_prime(value);
}
```

Assignment 2: Set job function

```
set_job():
  IF (blocked_workers > 0)  {
    blocked_workers--;
    signal(worker_q);
  }
  ELSE {
    main_sleeping = True;
    wait(main_q);
  }
  IF (blocked_workers + working_workers != N)
    main_waiting_worker = True;
    Wait(main_q);
  }
```

Assignment 2: Get job function

```
command, value get_job():
  working_workers--;
  IF (main_sleeping) {
    main_sleeping = False;
    signal(main_q);
  }
  ELSE {
    blocked_workers++;
    wait(worker_q);
  }
  copy value, command to local variables
  working_workers++;
  IF (main_waiting_worker) {
    main_waiting_worker = False;
    signal(main_sleep);
  }
  return local copy of value, command
```

Εναλλακτική προσέγγιση: Χρήση ουράς για τα jobs

Monitor structure

```
monitor {
  int waiting;
  Queue<Job> job_queue;
  cond workers_queue;

  void add(Job job);

  Job get();
}
```

```
void add(Job job) {
  add job to job_queue
  if there are workers waiting
    signal(workers_queue)
}

Job get() {
  if job_queue is empty or there are other workers waiting {
    increment waiting by 1
    wait(workers_queue)
    decrement waiting by 1
  }

  return first item from job_queue
}
```

Εναλλακτική προσέγγιση: Χρήση ουράς για τα jobs

- Using a queue for the jobs allows the master to not wait for any acknowledgement from the workers.
- The worker just puts the next job in a queue and wakes up the next worker if there is one waiting.
- This solution works for both signal-block and signal-continue models of monitors because all signal calls happen in the end of each function.
- It works for both open and eggshell models of monitors because the waiting variable prevents unwanted execution order.

Assignment 3

Assignment 3: Monitor structure

```
monitor {
  cond red_queue, blue_queue;
  int bridge_color;
  int on_bridge;
  int cars_waiting[2];
  int total_cars[2];


  void enter_bridge(int color);
  void exit_bridge(int color);
}
```

Assignment 3: Enter bridge

```
FUNCTION enter_bridge(myColor):
    // Decision-making for crossing
    IF Bridge.color IS NULL:
        SET Bridge.color = myColor
    ELSE IF Bridge.color != myColor OR Bridge.carsOn == BRIDGE_LIMIT OR
waitingInQueue[myColor] > 0:
        INCREMENT waitingInQueue[myColor]
        WAIT(Queue[myColor])
        DECREMENT waitingInQueue[myColor]

    // Logistics before entering
    INCREMENT Bridge.carsOn
    INCREMENT totalCarsPassed[myColor]

    // Notify next car of the same color
    IF Bridge.carsOn < BRIDGE_LIMIT AND waitingInQueue[myColor] > 0 AND
(totalCarsPassed[myColor] < CON_COLOR_LIMIT OR NOT waitingInQueue[!myColor]):
        SIGNAL(Queue[myColor])
END FUNCTION
```

Assignment 3: Exit bridge

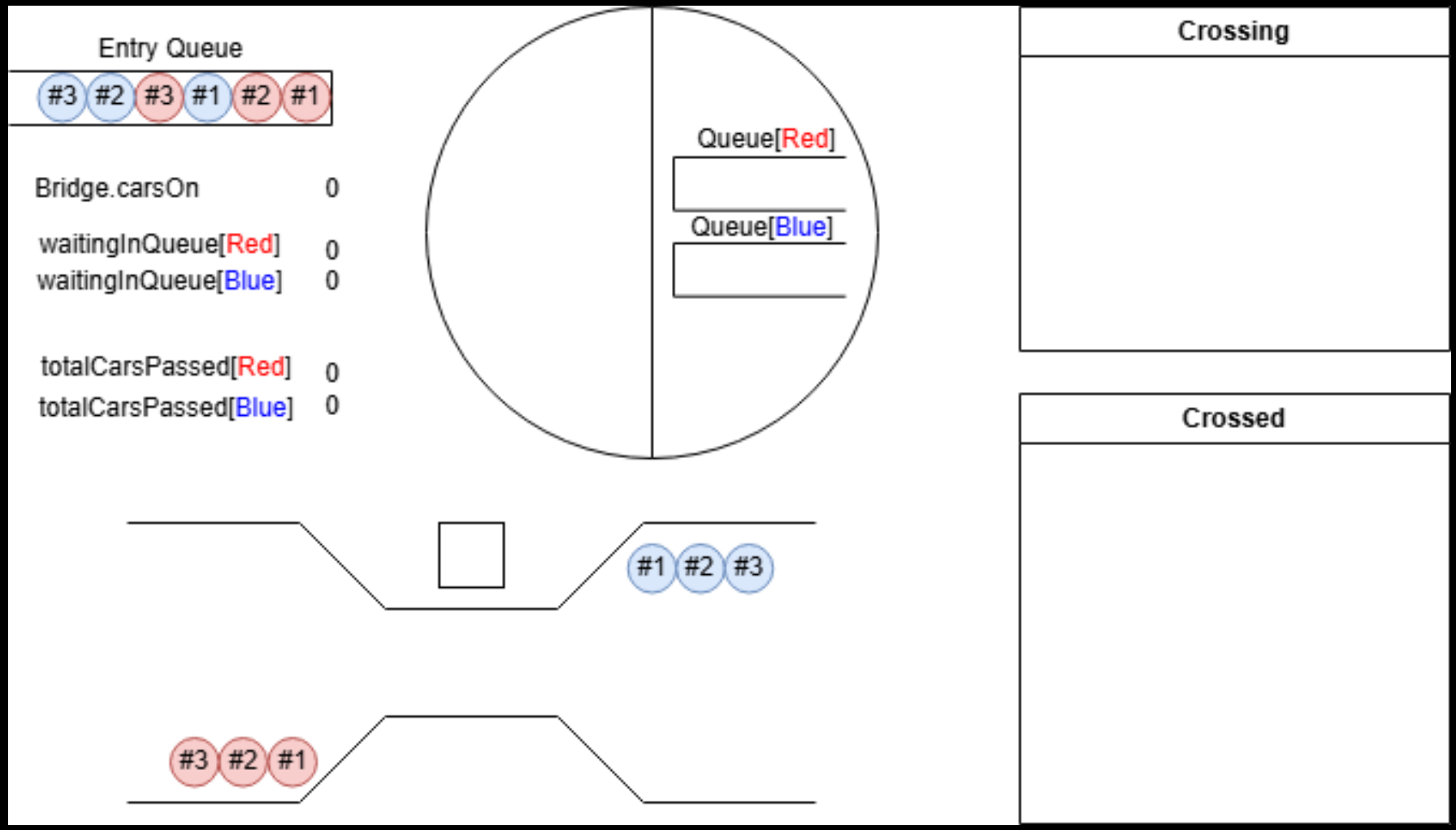
```
FUNCTION exit_bridge(myColor):
    DECREMENT Bridge.carsOn

    // If myColor hasnt reached its limit
    IF totalCarsPassed[myColor] < CON_COLOR_LIMIT:
        // Decision-making for bridge color
        IF waitingInQueue[myColor] > 0:
            SIGNAL(Queue[myColor])
        ELSE IF Bridge.carsOn == 0:
            SET totalCarsPassed[myColor] = 0
            IF waitingInQueue[!myColor] > 0:
                SET Bridge.color = !myColor
                SIGNAL(Queue[!myColor])
            ELSE:
                SET Bridge.color = NULL
    // If myColor reached its limit
    ELSE IF waitingInQueue[!myColor] > 0:
        IF Bridge.carsOn == 0:
            SET totalCarsPassed[myColor] = 0
            SET Bridge.color = !myColor
            SIGNAL(Queue[!myColor])
```

Assignment 3: Exit bridge

```
    // But Opposite color isnt waiting
    ELSE IF waitingInQueue[myColor] > 0:
        SIGNAL(Queue[myColor])
    // Neither is waiting
    ELSE IF Bridge.carsOn == 0:
        SET Bridge.color = NULL
        SET totalCarsPassed[myColor] = 0
END FUNCTION
```

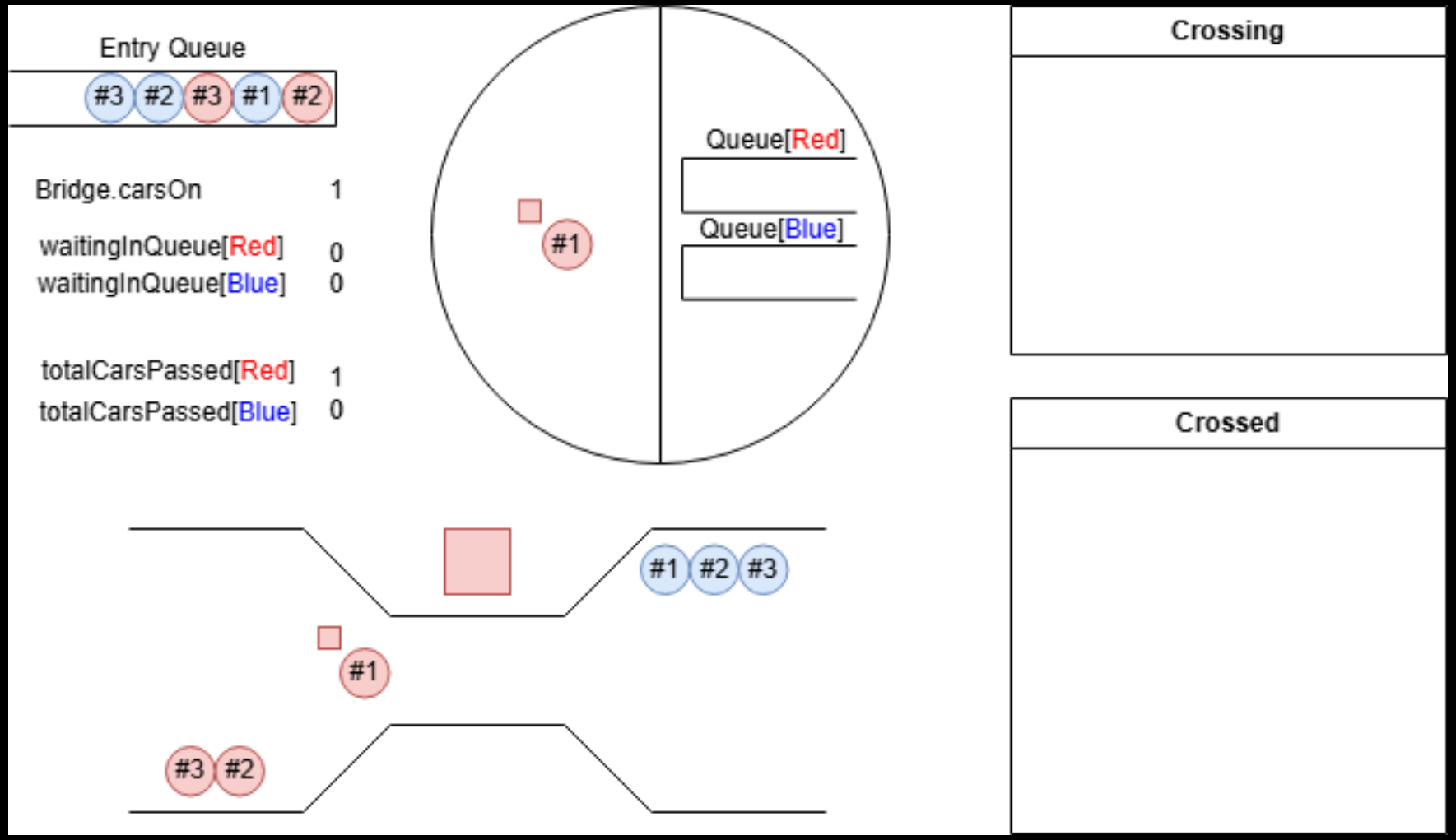
Assignment 3: Παράδειγμα εκτέλεσης



Red #1 is first and therefore claims the bridge for the red color, does logistics and immediately starts.

There’s no need to wake someone up from the red queue since it is empty

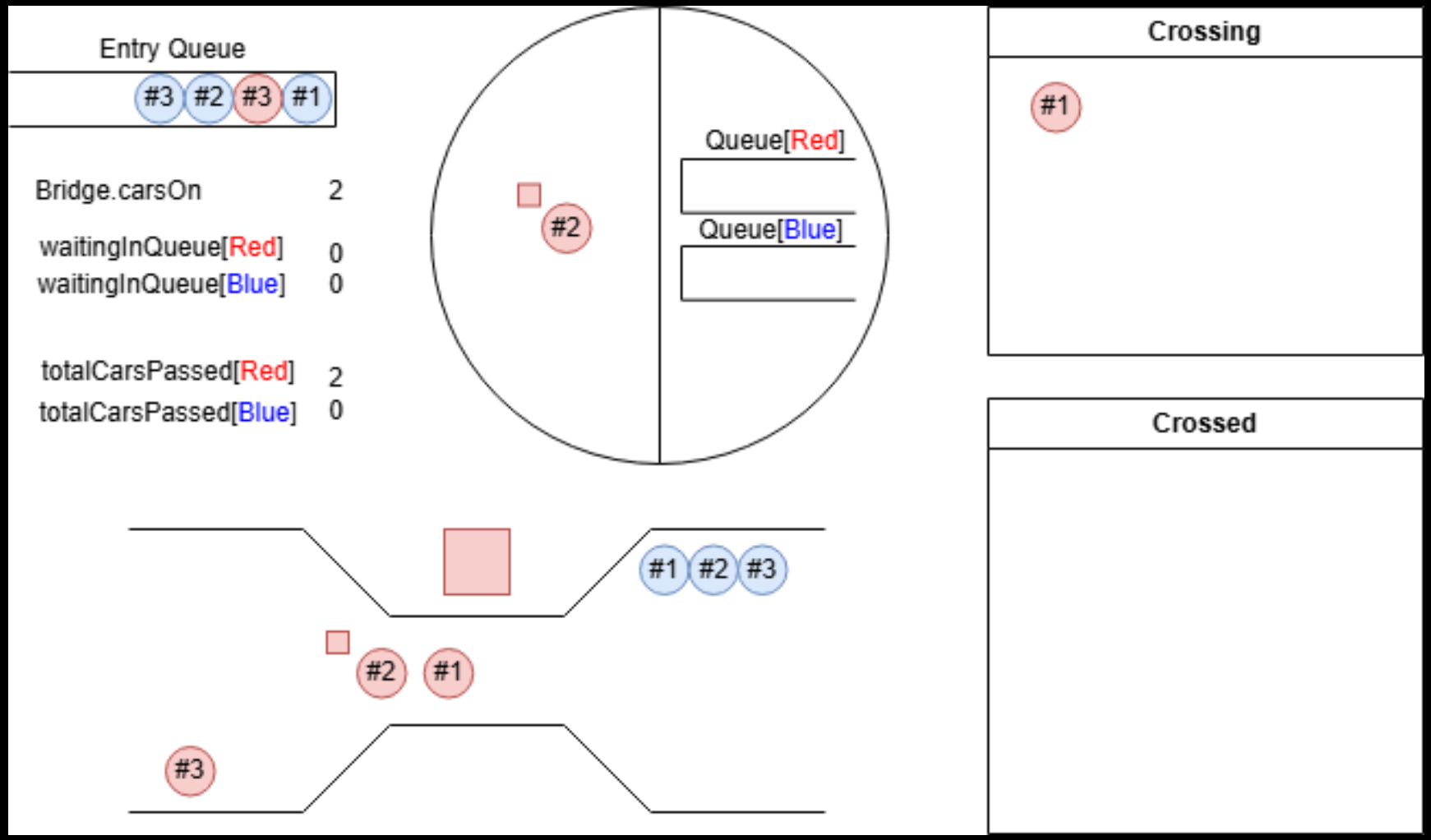
Assignment 3: Παράδειγμα εκτέλεσης



Red #2 checks the bridge's color, checks if there's space on the bridge, does logistics and starts crossing.

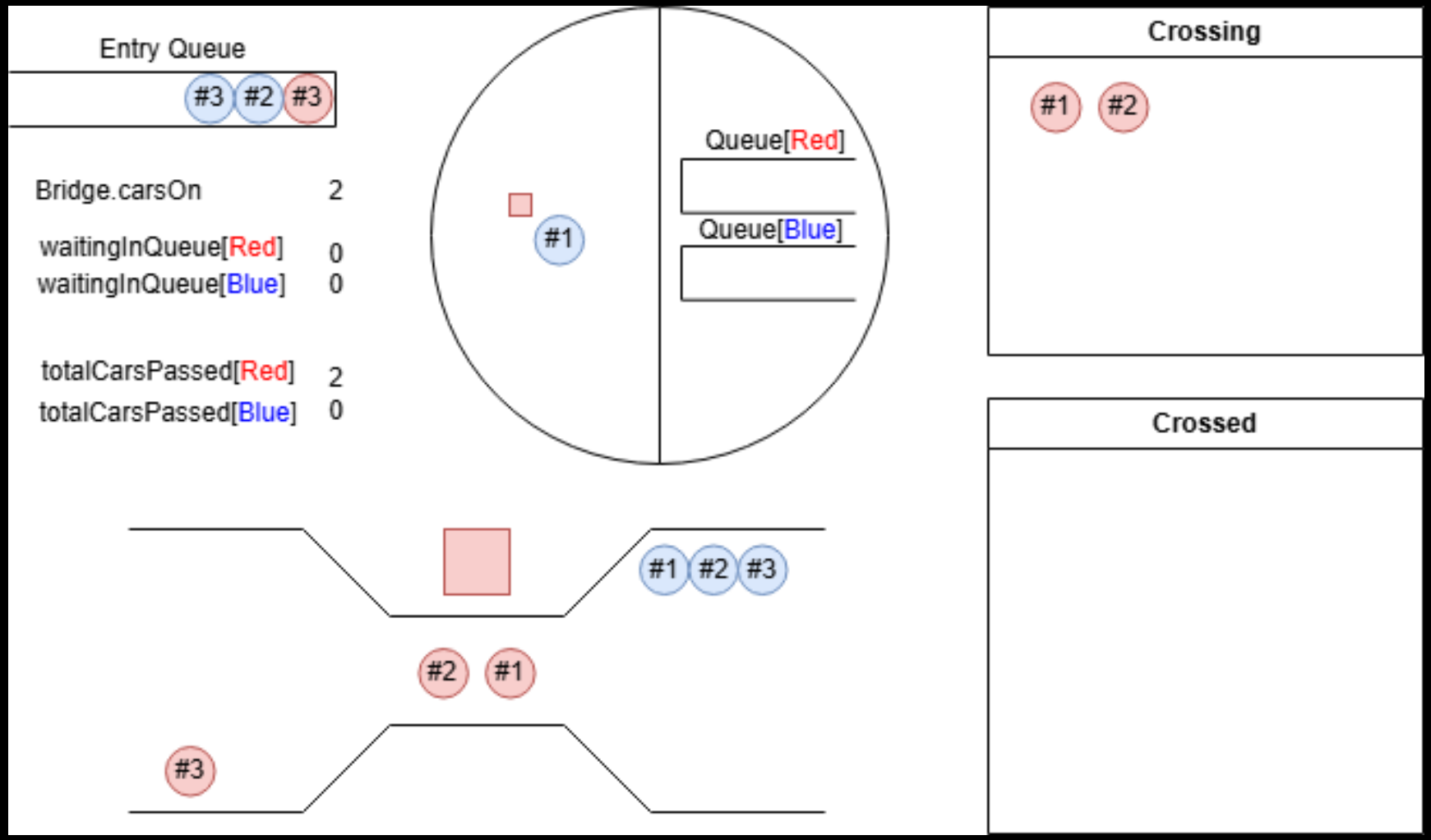
He can't wake someone from the queue since the bridge is full.

Assignment 3: Παράδειγμα εκτέλεσης



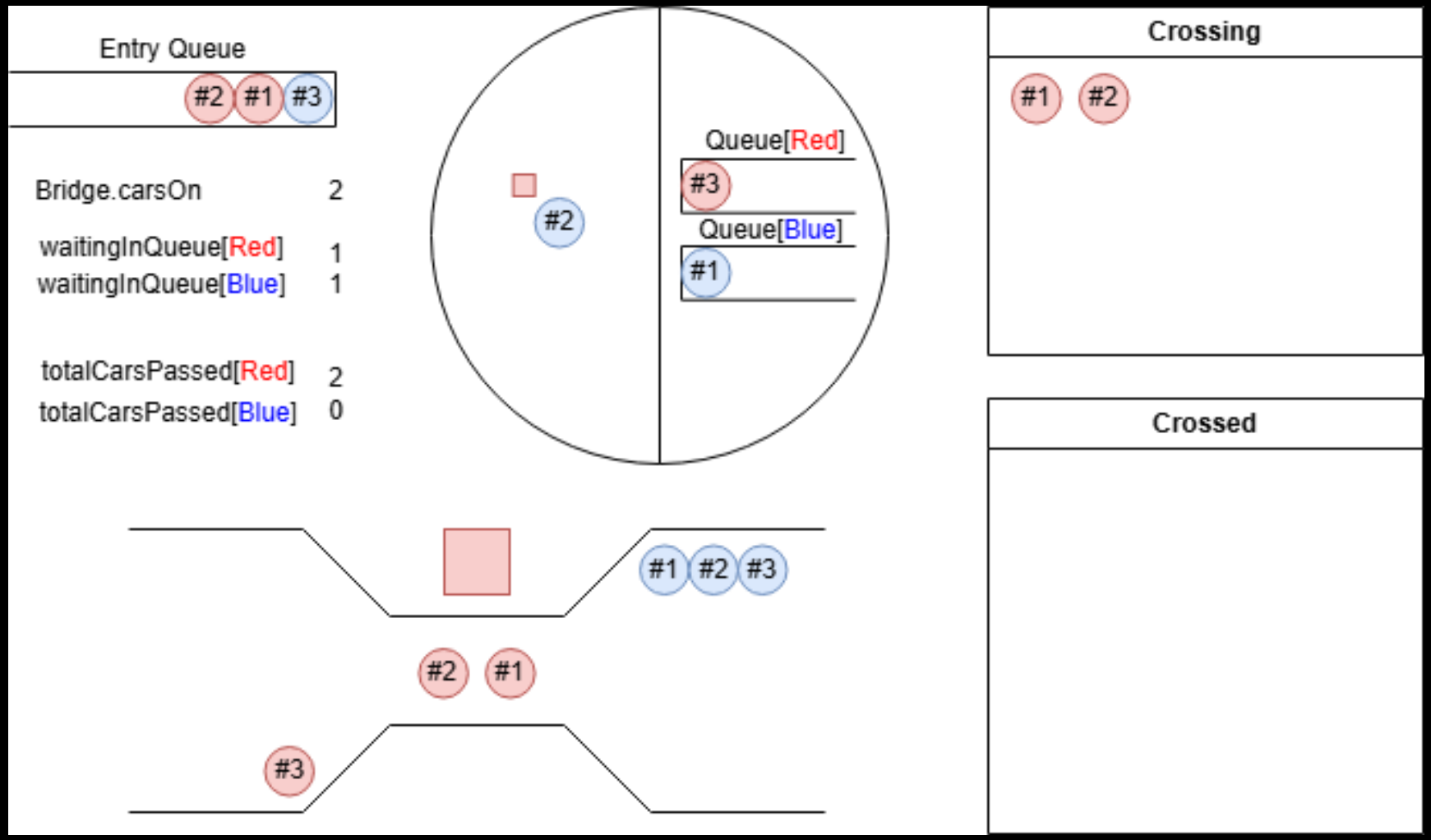
Blue #1 checks the color of the bridge and sees that it isn’t his color so he sleeps in the queue

Assignment 3: Παράδειγμα εκτέλεσης



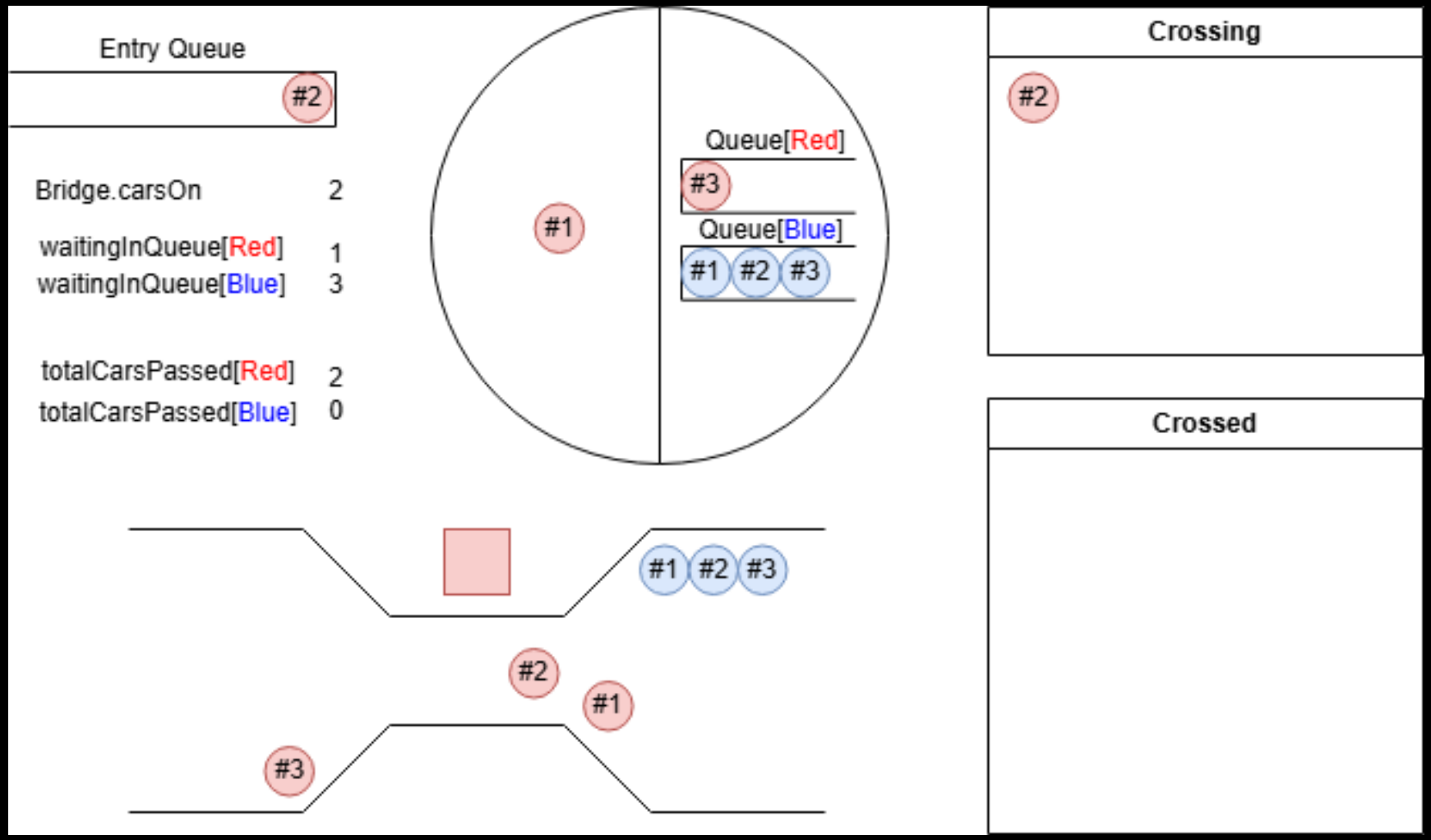
Red #3 sees the bridge is his color but there's no space so he waits in the queue

Assignment 3: Παράδειγμα εκτέλεσης



Blue #2 and #3 check the color of the bridge and because it isn't their color they wait in the queue

Assignment 3: Παράδειγμα εκτέλεσης



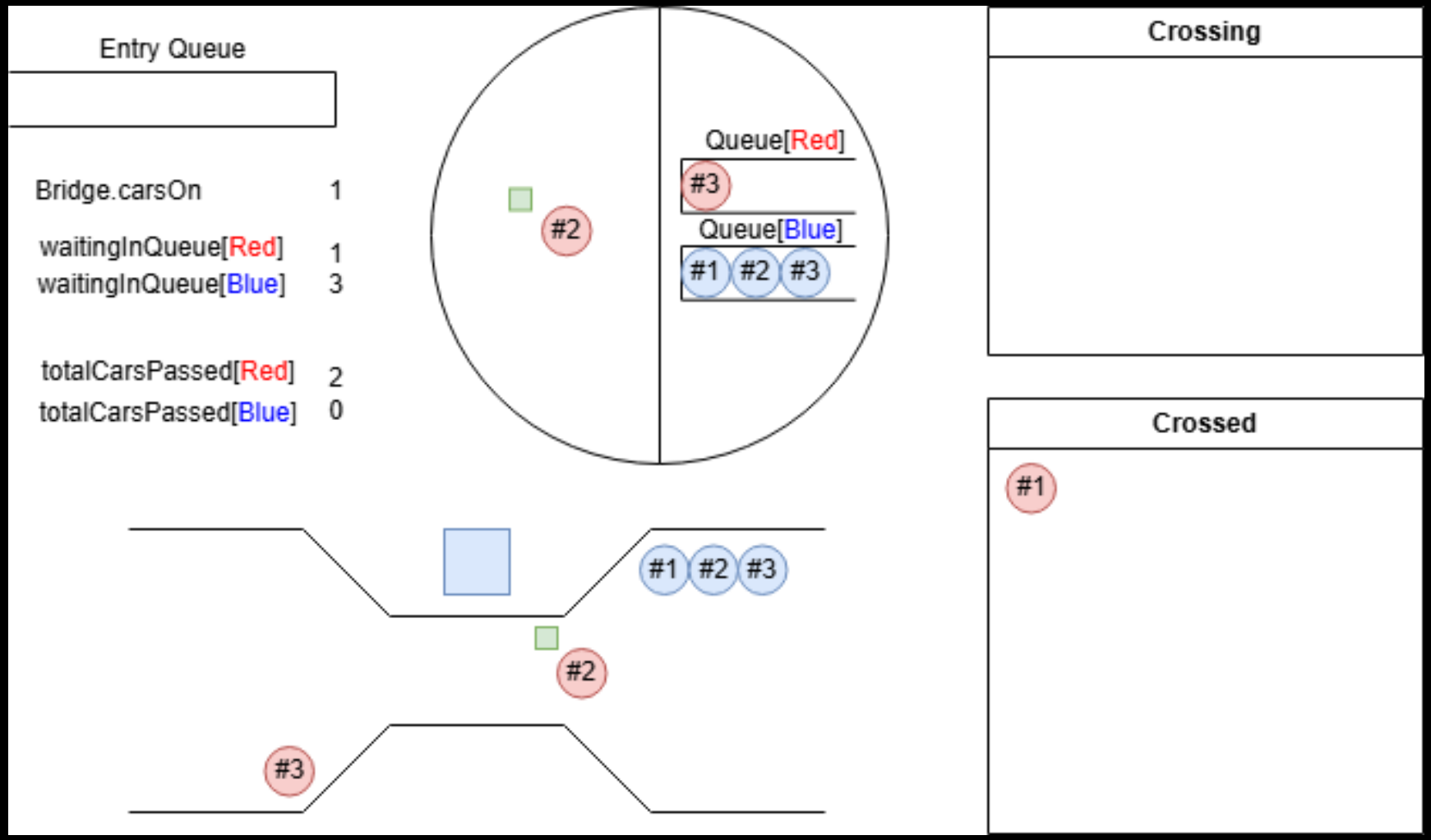
Red #1 is done crossing and thinks:

- Red has reached its limit
- Someone is waiting from the other color
- I am not last

Because of those he

- Does nothing

Assignment 3: Παράδειγμα εκτέλεσης



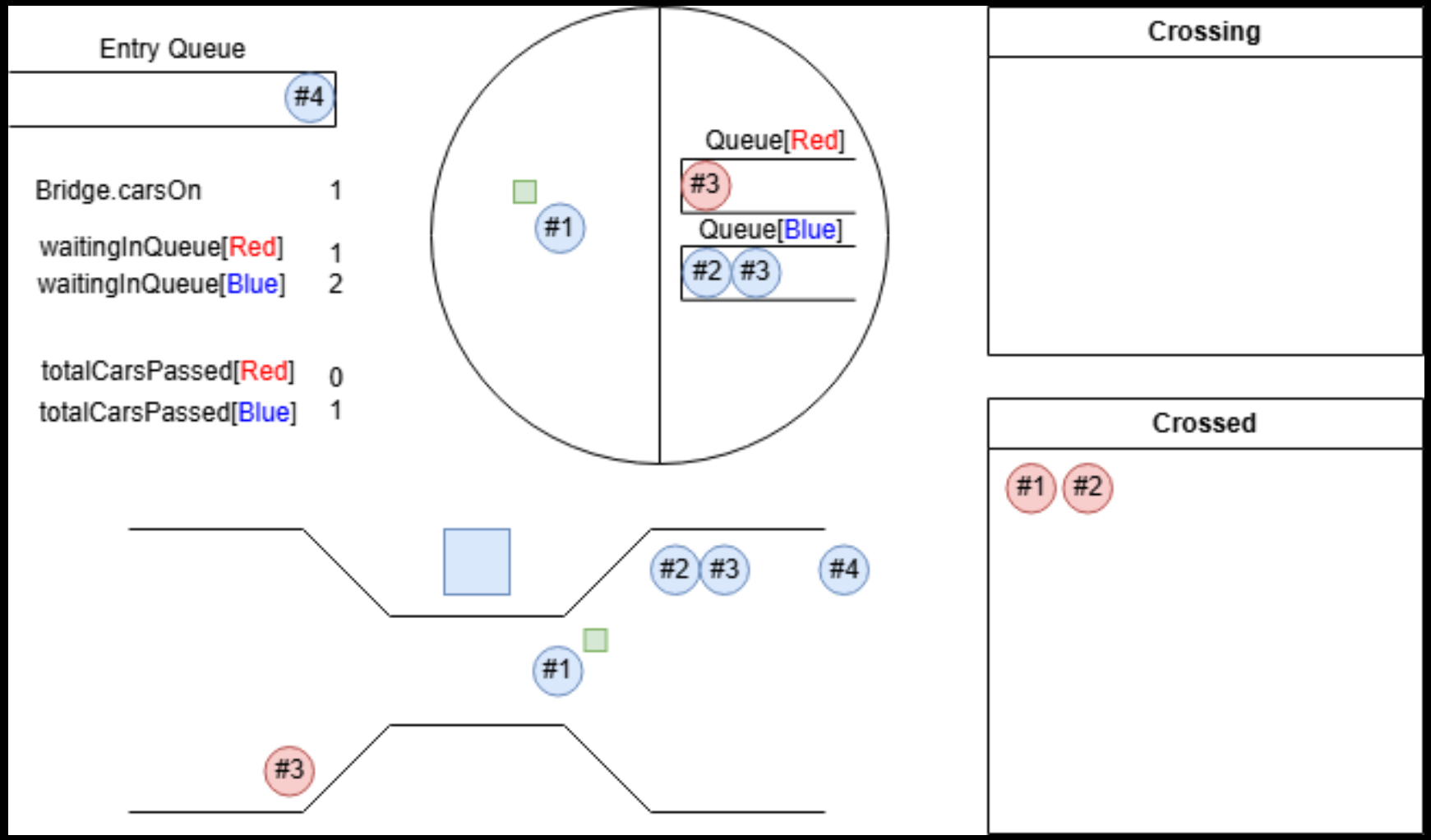
Red #2 is done crossing and thinks:

- Red has reached its limit
- Someone is waiting from the other color
- I am last

Because of those he

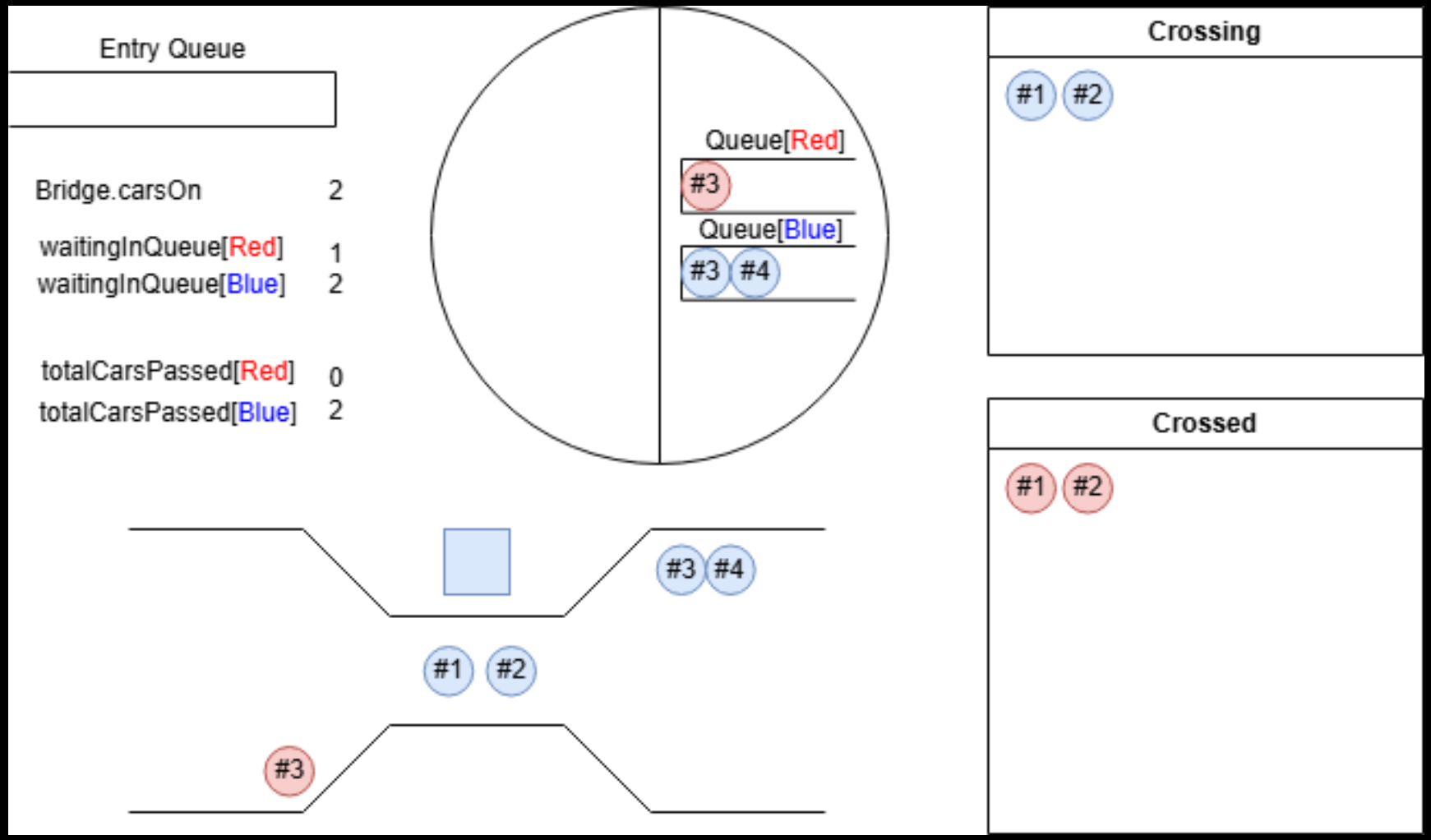
- Change the bridge color
- Wake up a blue car

Assignment 3: Παράδειγμα εκτέλεσης



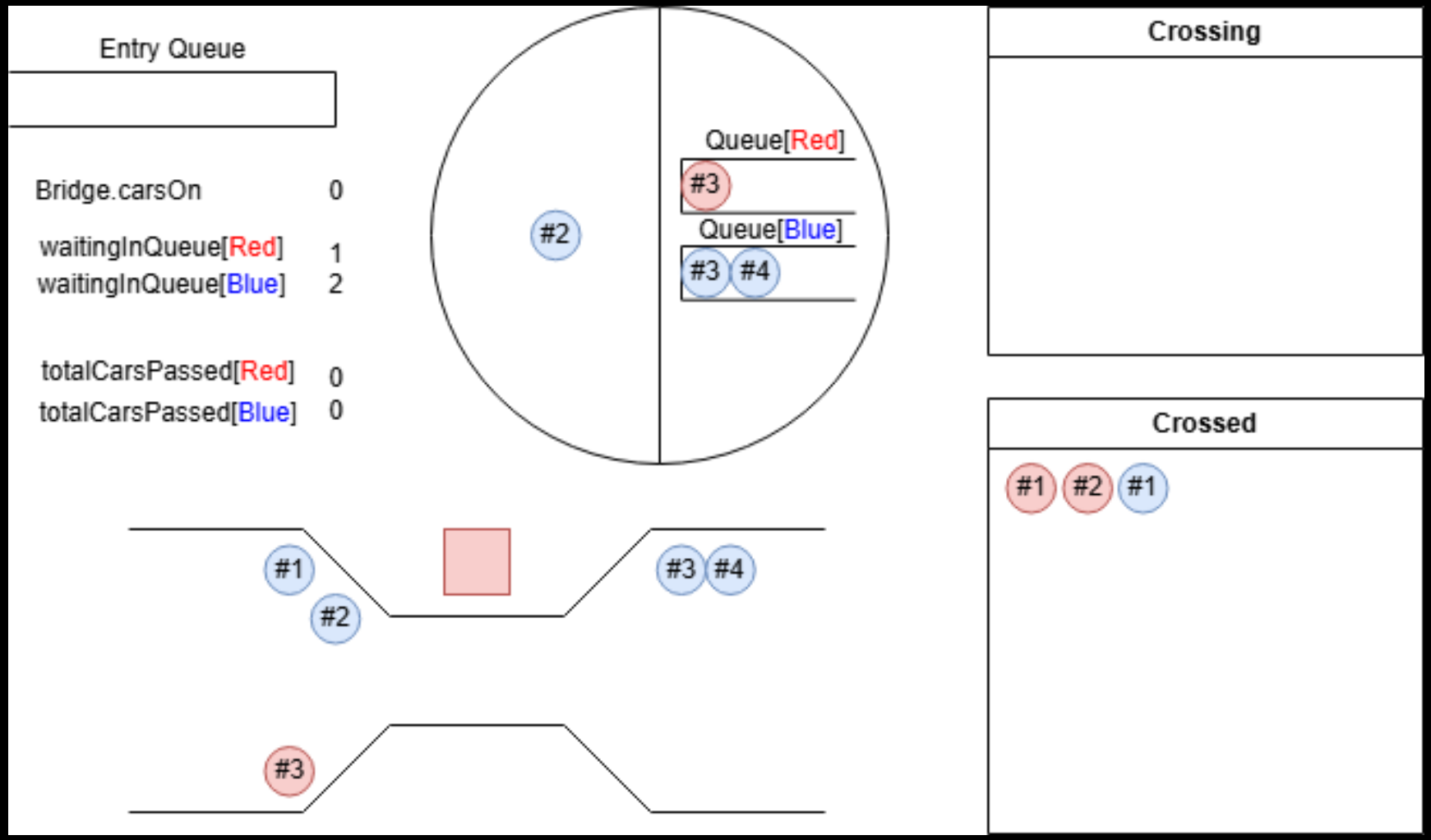
Blue #1 wakes up and does logistics, wakes up another blue car because there is space for another one and starts crossing.

Assignment 3: Παράδειγμα εκτέλεσης



Blue #2 wakes up and does logistics, doesn't wake up another blue car because there isn't space for another one and then starts crossing.

Assignment 3: Παράδειγμα εκτέλεσης



Blue #1 is done crossing and does logistics, he thinks:

- Blue has reached its limit
- Someone waiting of the opposite color
- I am not last

He decides:

- I do nothing

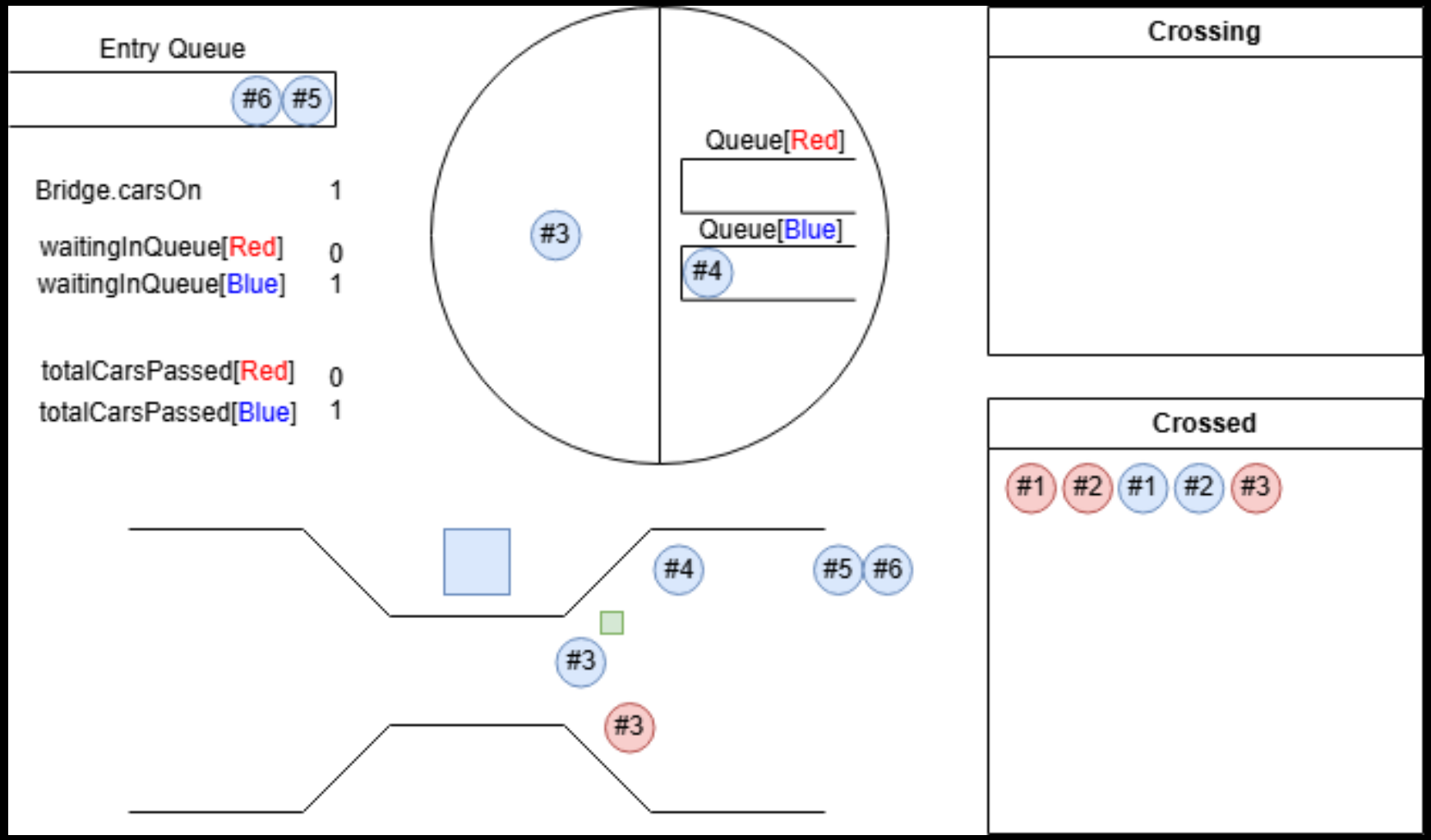
Blue#2 is done crossing and does logistics, he thinks:

- Blue has reached its limit
- Someone waiting of the opposite color
- I am last

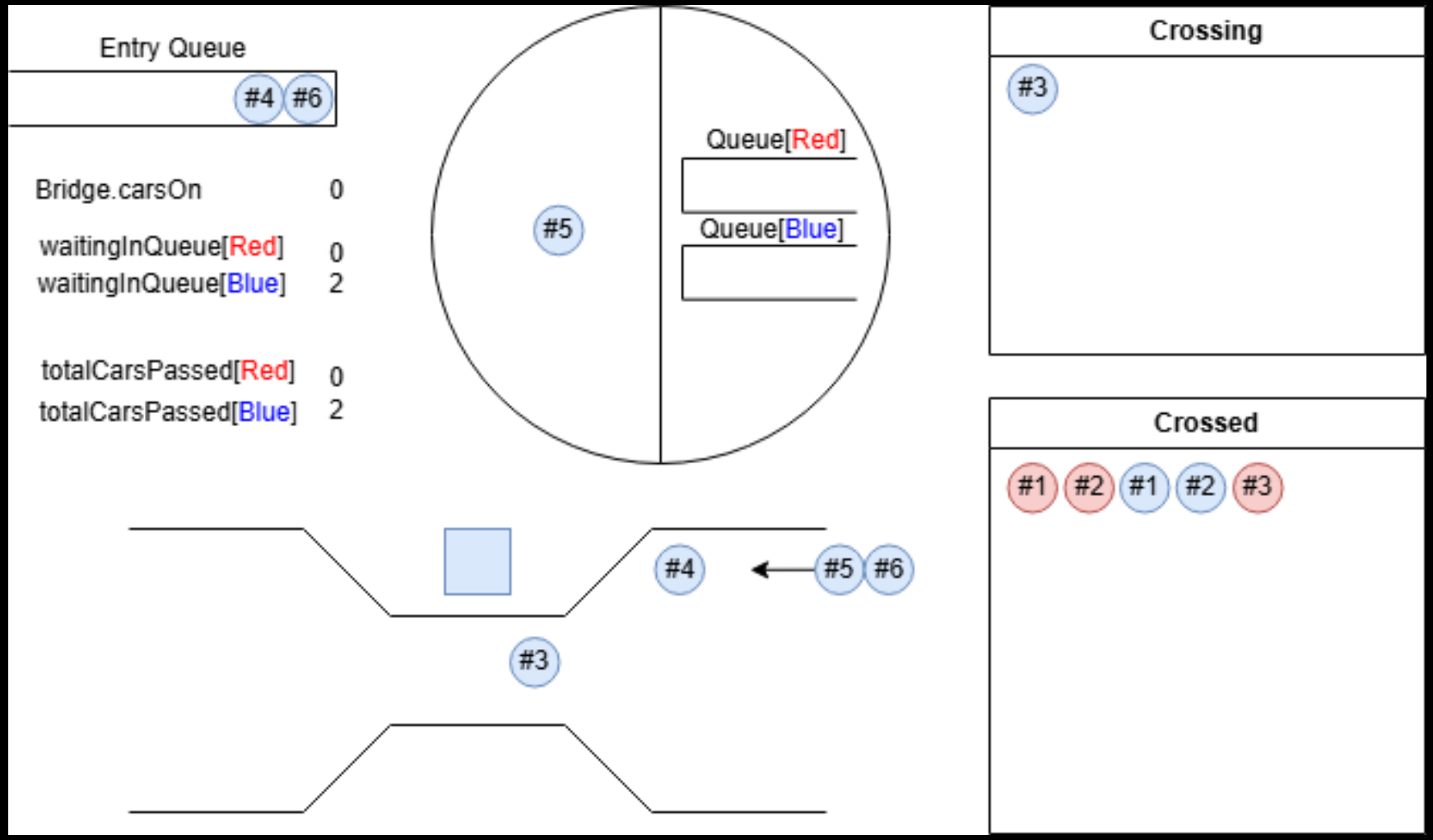
He decides:

- I change the bridge color and wake up a red car

Assignment 3: Παράδειγμα εκτέλεσης

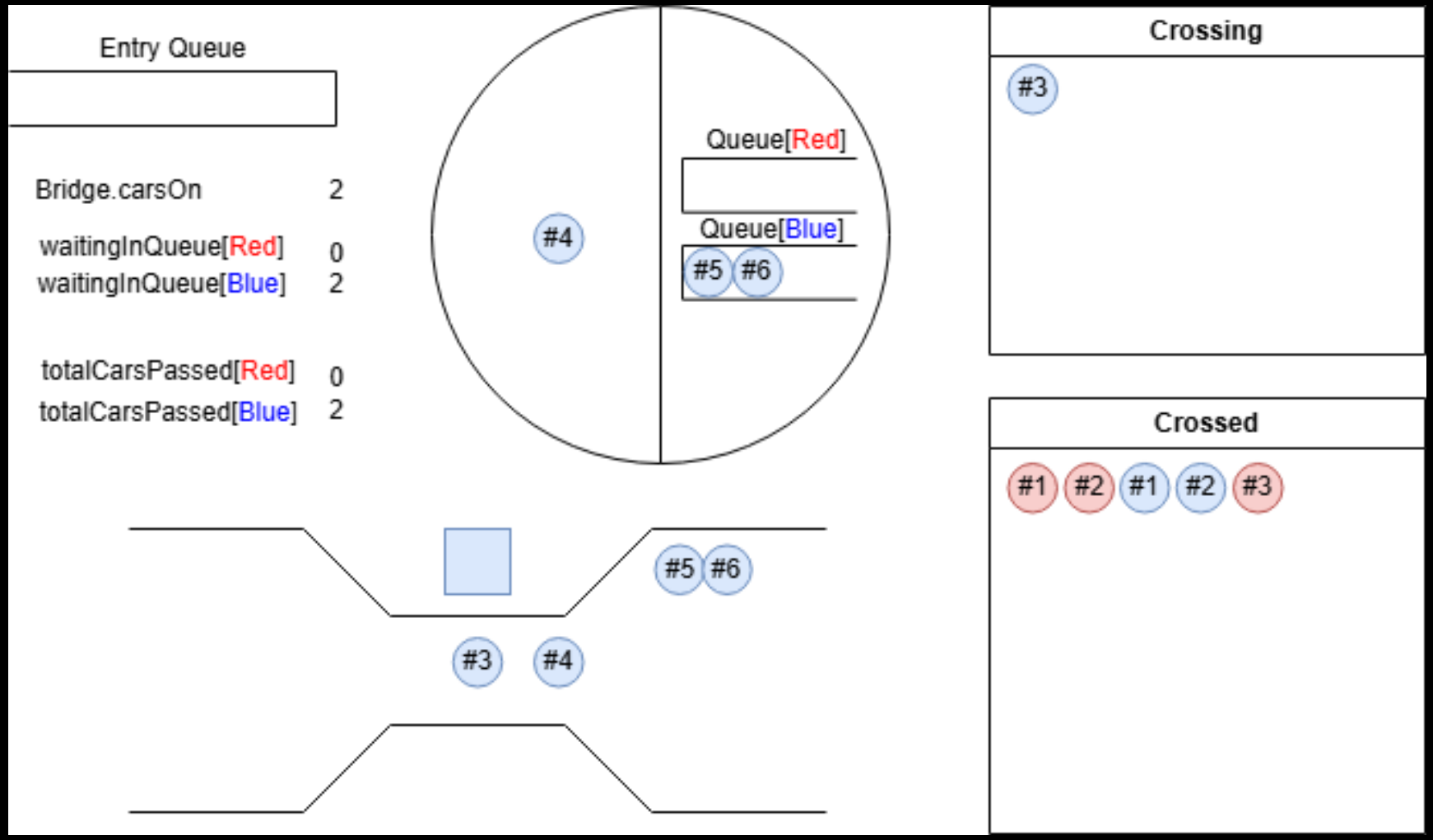


Assignment 3: Παράδειγμα εκτέλεσης



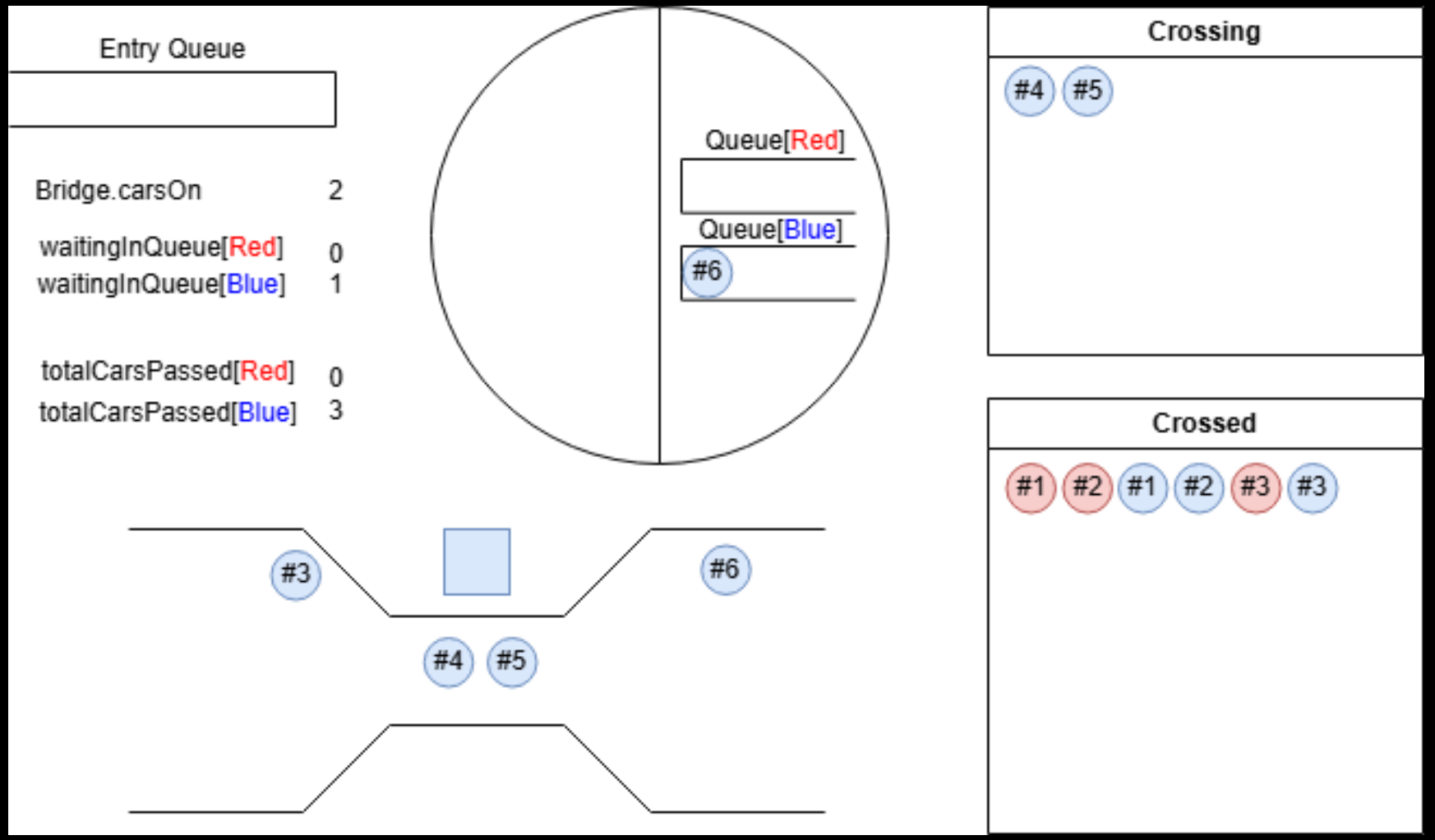
Blue #5 and #6 see there's someone in the queue so they also wait in the queue

Assignment 3: Παράδειγμα εκτέλεσης



Blue #4 is awoken and does logistics and doesn't wake someone from the queue because there is no space and starts crossing.

Assignment 3: Παράδειγμα εκτέλεσης



Blue #3 done crossing
Does logistics
Thinks:

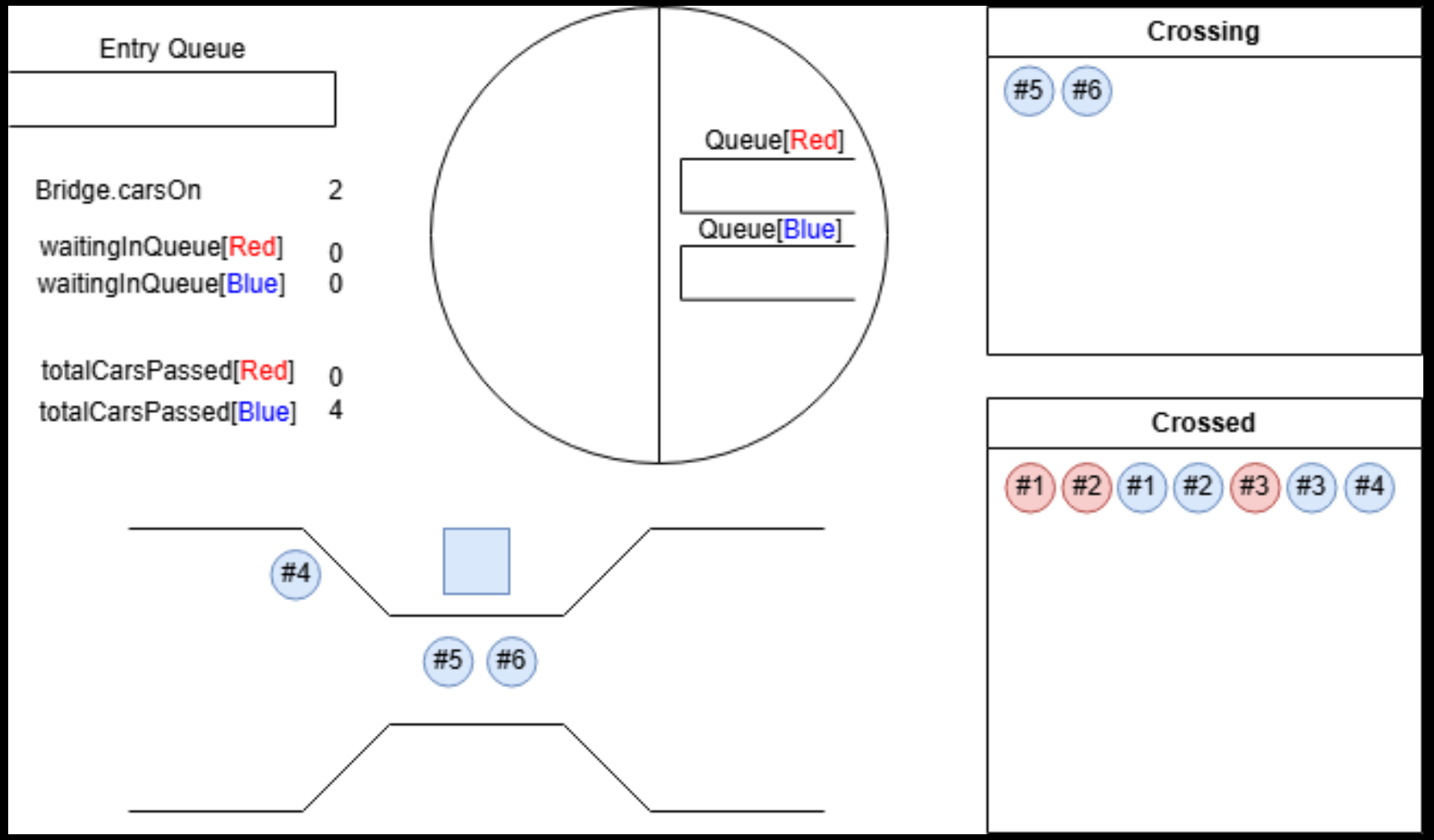
- Blue has reached its limit
- But no one is waiting of the opposite color
- Yet there is someone waiting of my color

Decides:

- I wake up someone of my color

Blue #5 awoken
Does logistics
Doesn’t wake up someone as there is no space
Starts crossing

Assignment 3: Παράδειγμα εκτέλεσης



Blue #4 done crossing
Does logistics
Thinks:

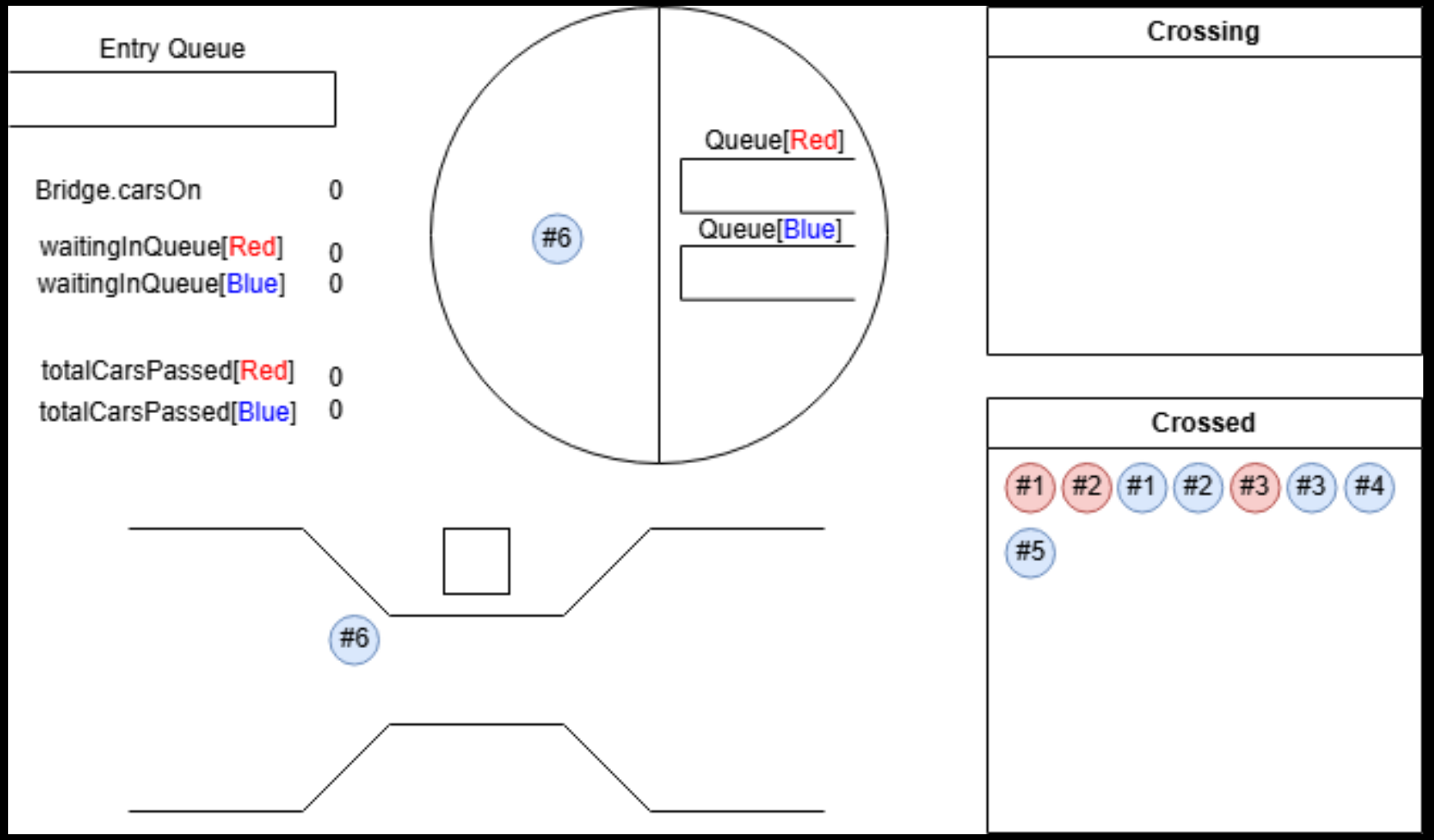
- Blue has reached its limit
- But no one is waiting of the opposite color
- Yet there is someone waiting of my color

Decides:

- I wake up someone of my color

Blue #6 awoken
Does logistics
Doesn’t wake up someone as there is no space and Queue is empty
Starts crossing

Assignment 3: Παράδειγμα εκτέλεσης



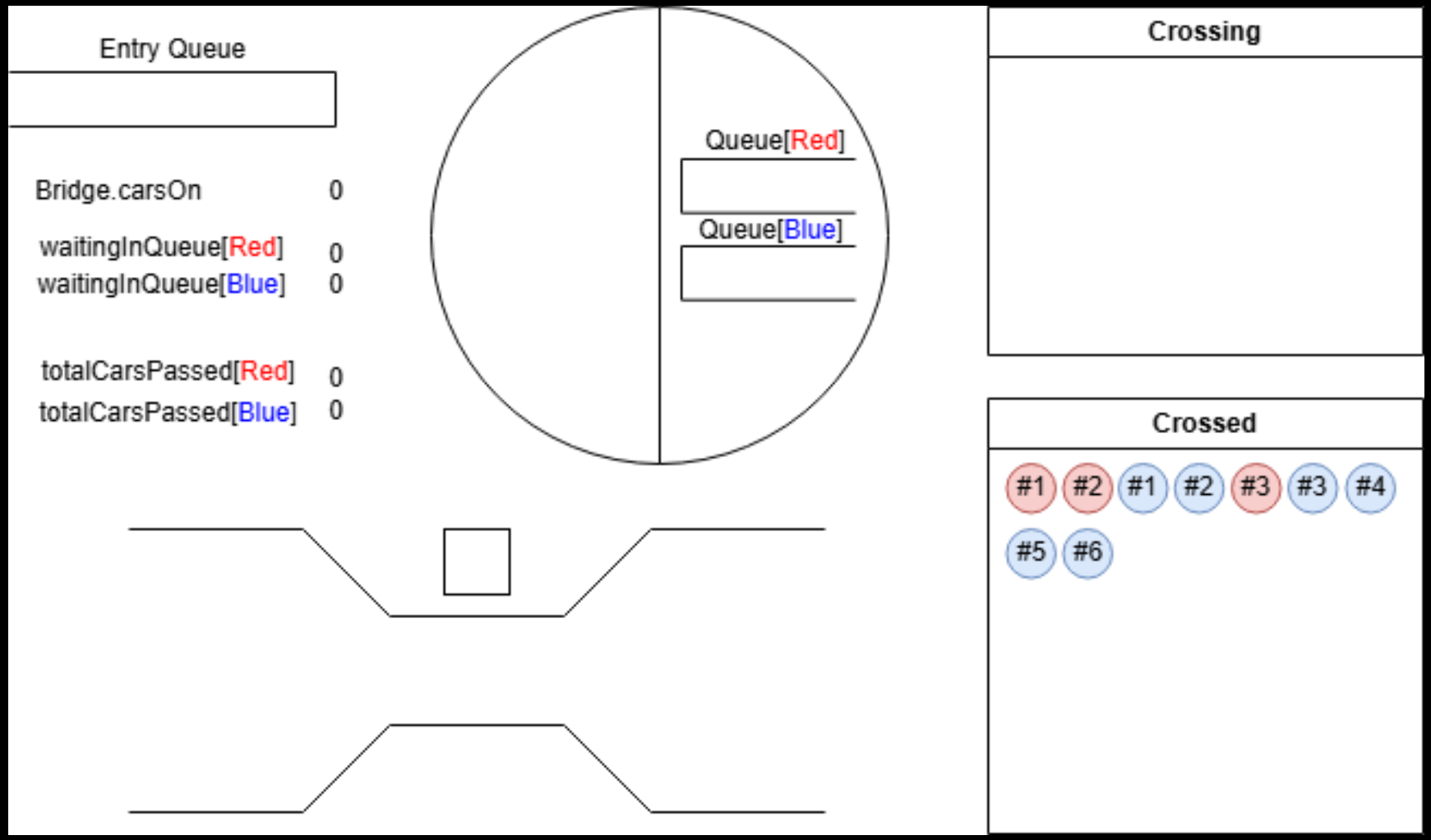
Blue #6 done crossing
Does logistics
Thinks:

- Blue has reached its limit
- But no one is waiting of the opposite color
- But no one is waiting of my color either

Decides:

- I set the bridge to unassigned

Assignment 3: Παράδειγμα εκτέλεσης



Empty bridge state

Assignment 4

Assignment 4: Monitor structure

```
monitor {
    bool park_closed;
    bool train_running;
    int passengers_waiting;
    int passengers_onboard;
    int max_passengers;

    cond train_wait;
    cond boarding_queue;
    cond ride_queue;

    void passenger_board();
    void passenger_unboard();
    void train_fill();
    void train_empty();
}
```

Assignment 4: Passenger board

```
Void passengers_board() {
  Passengers_waiting++;
  IF (train_running or passengers_waiting != max_passengers) {
    wait(boarding_queue);
    passengers_onboard++;
  }
  ELSE {
    Passengers_onboard++;
    train_running = true;
    signal(train_wait);
  }
}
```

Assignment 4: Passenger unboard

```
Void passenger_unboard() {
    IF (train_running) { wait(ride_queue); }
    Passenger_onboard--;
    IF (passengers_onboard == 0) { signal(train_wait); }
}
```

Assignment 4: Train fill

```
Bool train_fill() {
    IF (passengers_waiting < max_passengers) { wait(train_wait); }
    FOR N-1 passengers { signal(boarding_queue); }
}
```

Assignment 4: Train empty

```
Void train_empty() {
    train_running = false;
    passengers_waiting -= max_passengers;
    FOR every passenger onboard { signal(ride_queue); }
    IF (passengers_onboard != 0) { wait(train_wait); }
}
```