# Challenge 3

# Krusty Kalkulations

## Problem Description

The ***Krusty Krab*** is a bustling restaurant in Bikini Bottom, but has failed to modernize in digital technology—Mr. Krabs won't pay a dollar for Software as a Service, and isn't buying in to the AI hype. In time for the elevety-seveth anniversary, Mr. Krabs has tasked you with managing reservation requests to maximize customer satisfaction while adhering to the restaurant's seating capacity. The Krusty Krab is open from 09:00 hours to 23:00 hours. Each reservation specifies:

- An integer S, the ***start time*** of the reservation, in hours.
- An integer E, the ***end time*** of the reservation (exclusive) in hours.
- An integer N, the ***number of seats*** required for the reservation.

Your goal is to count the number of accepted reservations without exceeding the total seating capacity of the Krusty Krab at any time. After all, more reservations = more money! Reservations are processed in order, as they are called in.

## Input Specification

- An integer ***K*** representing the seating capacity of the Krusty Krab.
- ***R*** tuples each space-separated, where each tuple is a reservation (start, end, number)

## Output Specification

- An integer representing the maximum number of reservations that can be accepted without exceeding the seating capacity at any time.

# CHALLENGE 3

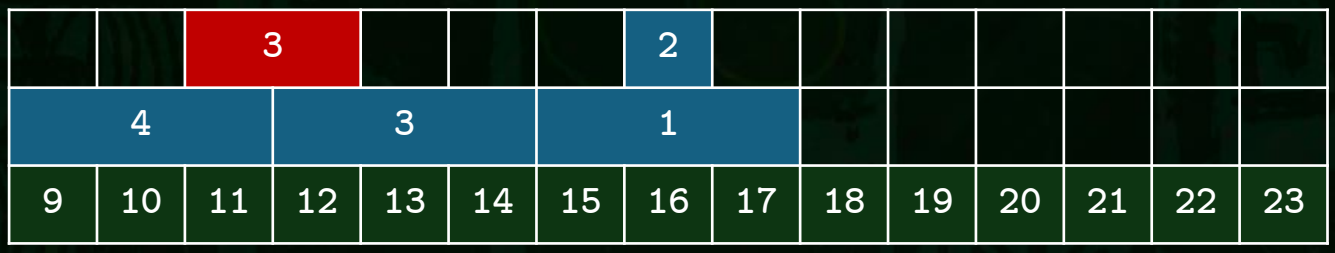
## Krusty Kalkulations

### Sample Input

```
6
(9, 12, 4) (15, 18, 1) (16, 17, 2) (12, 15, 3)
(11, 13, 3)
```

### Sample Output

```
4
```



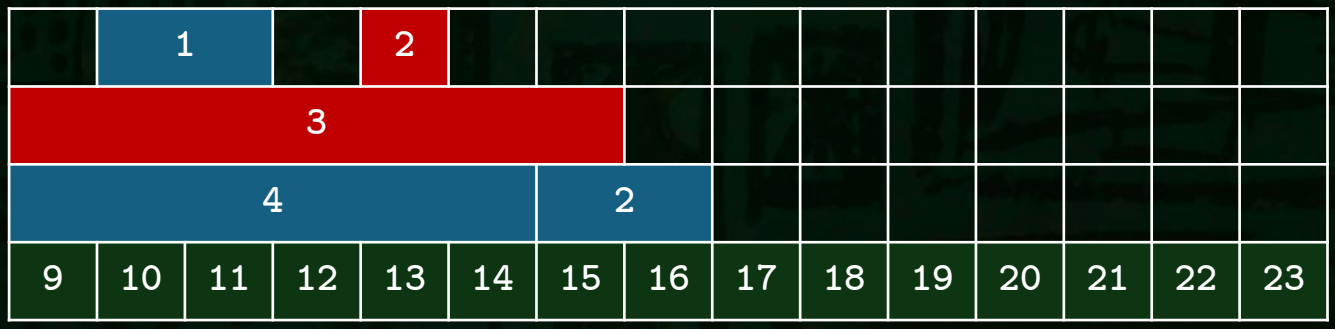

Here, we reject the 5th reservation.

### Sample Input

```
5
(9, 14, 4) (9, 16, 3) (10, 12, 1) (13, 14, 2)
(14, 17, 2)
```

### Sample Output

```
3
```




Here, we reject the 2nd and 4th reservations.

**As a bonus, if you're finished this one, optimize for the MAXIMUM number of possible reservations, regardless of the order they come in. In the second code example on this page, an optimized solution would have 4 reservations instead of 3.**