```
// N number of days the property is available e.g. 100
// l limit for a single booking to meet maximum of N
// Find minimum l that can be set for this list of bookings
// input: a list of bookings and a max number of days
```

```java
static ArrayList<Integer> limitBookings(ArrayList<Integer> bookings, int maxDays) {
    while (totalDays(bookings) >= maxDays) // loop until postcondition is met
    {
        int curLongestBooking = longestBooking(bookings); // longest booking

        for (int i = 0; i < bookings.size(); i ++) { // loop over bookings
            int currBooking = bookings.get(i);
            if (currBooking >= curLongestBooking) { // take away one from longest booking
                bookings.set(i, currBooking - 1);
            }

            if (totalDays(bookings) < maxDays) { // check postcondition
                System.out.println("using booking limit: " + longestBooking(bookings) + " days");
                return bookings;
            }
        }
    }
    return bookings;
}
```

Helper functions:

```java
static int totalDays(ArrayList<Integer> bookings) {
    int total = 0;
    for ( Integer i : bookings) {
        total += i;
    }
    return total;
}

static int longestBooking(ArrayList<Integer> bookings) {
    return Collections.max(bookings);
}
```

This algorithm finds the maximum values then decreases it until the total is less than maxDay (input)

Upon testing this algorithm gives the expected result for then given input: 26, 14, 45, 10, and 28 days

Also tested on more data sets including: ([10, 20, 30, 40, 50], 100), ([65, 92, 75, 32, 32], 120), ([100, 1, 1, 1, 1, 100])

Prove or disprove the following statements:

(a) n 2 +n+1 = Θ (n 3).

(b) if f1(n) = O(g1(n)) and f2(n) = O(g2(n)), then f1(n) + f2(n) = O(g1(n) +g2(n)).

(c) n ∑ i=1 i = O(n 3)