Seat Reservation System

Problem Understanding

1. The problem states that a coach has 80 seats and each row has a total of 7 seats. So we can take a matrix of dimension 12x7 where 12 represents the number of rows and 7 represents number of columns respectively.

2. So now each cell will be representing the seats in a particular row, We can later denote it by i and j representing the ith row and jth column respectively.

3. Now one person can reserve upto 7 seats at a time so, we can assume that one person is allowed to book an entire row.

4. In this problem Statement, we are assuming that the booking will be done in a consecutive manner.

5. So, what we can think of a boolean matrix where, each of the cells are marked as false by default.

6. Whenever a user is booking a seat then, we can mark the cell as true representing the particular seat is booked.

7. Now whenever a seat is booked, we will be deleting (total number of seats - number of seats that are booked currently), this way we can keep track of the number of available seats.

8. The program will initially end if the number of available seats = 0

Taking some few examples :

Case I : Lets say user enters the value = 7 where (value == total number of seats in a row)

a. Then we have to traverse the boolean matrix row wise to check how many seats are available.

b. lets say all 7 seats are available. Then, we can mark each cell as true.

c. After that number of available seats = total seats - booking value.

Case II : Where the value < total number of seats in a row i.e 7

a. Lets say user entered 4 which is less than 7. Here we are assuming that initially all the seats are available.
b. So we can deduct the seat count.
c. Here we are left with 3 seats which are unoccupied.

Case III : Where user entered 0
a. In this case, we will simply return

Case IV : Where all the seats are booked ie 80 and user is entering some value
a. In this case, print "All seats are booked" and return

Some Improvements to the logic

1. We can have an array **Arr** of **size 13** where each cell is initially filled with **7** except the last cell. **The last cell will be filled with 3.**
2. This array is representing the number of availability of seats in each row of the matrix.
3. The array index will be 1 indexed inorder to maintain the ease.
4. When a user wants to book a seat, instead of traversing the whole matrix row-wise, we can directly get the availability of the seat using the array that we have created.
5. This will reduce our time complexity to **O(N) in worst case from O(N^2)**.
6. Now when a seat is booked let say at row i
    a. Check **if Arr[i] > 0,** this means the seat is available
        i. Check the booking value <= Arr[i]
            1. If true then update the value of Arr[i] = Arr[i] - booking value.
            2. Else, move to another cell.
    b. Lets say, the Arr[i] = 3 and booking value is 5
        i. We have to check whether the total number of seats are available or not.
            1. If it is available, we can allocate 3 seats to Arr[i]
            2. Now the booking value is 2 so move ahead to i+1 index and check likewise and allocate according
        ii. In case the total number of seats< booking value
            1. Print “Can’t allocate seats”
            2. return.
7. **The priority will be to allocate all the empty seats at each index first, then we can move forward to the remaining available seats.**