Drew Goldman

Daycare - Description

The daycare problem appears complicated, until you realize a greedy solution can efficiently solve this problem. I handled the input with a while loop, which would execute continuously until the first line for a test case was blank, in which case the loop would break. Within each test case, I called the daycare(int ind, int ans) method, passing in (0,0) as the parameters. Moreover, I used a global 2D vector of size *n* rows and 2 columns, where the first column represents the current capacity of the room and the second column represents the new capacity of the room. The vector stored all the input for each case and was subsequently cleared after the answer was printed. The daycare method has a simple greedy structure and recursively calls itself by decreasing the size of the problem by one each time by incrementing the current index by one. The base case is if the current index = the number of rows *n*, in which case the current answer can be returned. If the current index = 0, then the answer would be incremented by the current capacity at index (0,0), the value at index (0,0) would be set to 0, and the answer would be recursed. I then use a for-loop to iterate from 0 to the current index row. Within this for-loop while a new variable, *move*—which is set to the value of the current capacity of the current index—is positive and the current capacity of the rooms vector is less than the new capacity, for each index *i*, the following occurs: move is decremented by one, a counter, *delta*, which starts at 0, is incremented by one, and the capacity at rms[i][0] is incremented by one. Finally, rms at the current index in the left column is subtracted by delta, the answer is then added by that value in rms and the value in rms is subsequently set back to 0. The recursion is then returned, as the index approaches the size by one. The runtime of this algorithm is $O(n^2)$ for each test case because the recursion occurs *n* times and the for loop iterates a maximum of *n* times for each test case.

No pseudocode required, as the code is available to the grader via Gradescope.