Lazy, but honest

- $D = 1$, 5 points.

  We have to complete each task we receive on the same day. First, check that every $b_i \leq a_i$. Then print the number of days with $b_i \neq 0$. $O(n)$

- $N \leq 18$, 7 points

  If we decide to work some day it's always optimal to do as many tasks as possible and we start from the earliest tasks.

  For each day we have two options: not to work or to do as many tasks as possible. Use recursion. Store for each day how many tasks are left uncompleted in that day. $O(2^n * n)$

- All a_i equal, 18 points.

  Greedy solution: if we decide not to work on that day can we complete all tasks(current and future) in any number of working days? If not we must work or else skip that day. To check that we can use a segment tree to find an interval with maximal value $b_l - a_l + b_{l+1} - a_{l+1} + \ldots + b_r - a_r$. $O(nlog(n))$

- $D = N$, 18 points.

  Let's find a solution for the suffix i with the maximal sum of b_j in working days. To move from i to $i-1$: If we can complete a_{i-1} tasks in current working days use them. Else find the largest b_j in not working days and mark it as a working day until we complete all a_{i-1} tasks. $O(nlog(n))$

- $N \leq 2000$, 16 points.

  Let x_i be the number of completed tasks in days $1, 2, \ldots i$. Then x_i must be not greater than $b_1 + b_2 \ldots + b_i$ and not less than $b_1 + b_2 \ldots + b_{i-D+1}$.

  Use $dp_{i,j} = x$ where i number of days, j number of working days, x maximal number of completed tasks. Transitions are easy and don't forget to check $b_1 + b_2 \ldots + b_{i-D+1} \leq dp_{i,j} \leq b_1 + b_2 \ldots + b_i$. $O(n^2)$

- No additional constraints, 36 points

  Use the same dp from the previous subtask. The key observation is that dp_i is a convex function. Then we can use *set* to store $dp_{i,j+1} - dp_{i,j}$. And transitions are just inserting b_i. To check bounds for value x store min value and max value in dp. $O(nlog(n))$