Problem Set 2 - Data Structures & Paradigms

Problem C

I remember thinking about this question for quite a while. I feel I've learned throughout this course that I will forever never see binary search as a solution in trying to search for a more 'elegant solution' and this was definitely the first sign of my blindness to binary search.

After pondering the question for a while, I remember - ignoring the whole, minimising the number of days thing - understanding that we wanted to sort the bugs from largest to smallest, and greedily allocate the cheapest student that can solve each bug. If a student can solve bug A in the sorted decreasing list of bugs, they can sort all other bugs, so, again greedily, the best way to use them for a second day is to just use them on the second most complicated bug. This was all well and good, but trying to balance this with also finding the minimal number of days that we can solve the bug in (i.e. actually how many times we repeat each student) I think I could not get.

In retrospect, the solution is so clear, and since then, I've forgotten the other things I was trying but I remember going down some weird alternate paths of like, attempting to allocate one student, and then trying to greedily decide whether or not to repeat that student or use the next. Feels really dead-endy now, not sure what I was thinking.

Anyway, one of my mates again, pointed me to the lecture slides, in which I found binary search, the cure to all my issues. The final solution would then be to binary search on the minimum number of days that the students can solve the bugs while keeping within the pass requirement, and then using that as our solution :)

As for implementation specific details I used a priority queue in order to store a min-heap of the students' pass costs. This meant that for every bug that we try to solve, I was getting the cheapest eligible student. I would then iterate through the batches pretty niftily in batches of size len (number of days), such that everytime we allocate a student, we're skipping the next set of bugs that they can solve in following days.

Finally, if we actually get a binary search solution, I wrote a function to - much in the same way as the can-allocate function - allocate the actual students to bugs, this time knowing that we can finish it in len time.

Done dealio!