Problem Set 2 - Data Structures & Paradigms

<u>Problem B</u>
When first reading this question, it was pretty clear that we were working with a stack and queue, the harder part of the question was trying to devise some sort of greedy means to choose the best move at a certain point. I think in this question, it was pretty clear that, given the full string of S, the best move to do would be to get the lowest character currently in s into u as the first letter. Literally by the definition of lexicographically minimal, there's no other letter that can go in the first place that would give us a more minimal string.

Now after that is done, here's the more confusing part, whether to take from the end of t, or to add more in from S. The determining factor behind what goes into the next spot in the string is the minimal letter between:

1. The smallest letter in the remaining characters in S (as we can always retrieve any character from the remaining in s by popping)
2. The character currently at the top of T.

So we compare those two things, and then make our decision from there. If the first, we pop until we find that letter, and if the second, we pop from T.

I'm explaining this in one go and without many hiccups, as I am much better at competitive programming-esque concepts and problems now, but when I was doing this question very early in the course, it was definitely much harder to come to these revelations - stuff like keeping track of the characters remaining in s with a set were completely new concepts to me, and thus I was kind of struggling lol.