Problem 3:

Pre-Implementation Notes:

With this problem our main goal is to essentially figure out the longest sequence of consecutive books where the time it takes to reach each one does not exceed the free time `t`.

My main approach to this goal is:

- Start from the left
- Then see how far right as i can go without exceeding time t
- Repeat this process for all positions i.e iterate through something like:

```
for (int left = 0; left < n; left ++) {
    // see how far right i can go
}
```

However, I can optimise this problem by keeping track of how far i went previously and seeing if my new position can beat that. We can hold the accumulative sum of the books by using a sliding window approach.

Post-Implementation Notes:

Challenges I ran into:

- I tried keeping track of the window size itself, but that gave me so many problem with indexing, because i was losing track of how far i'm supposed to go and some calculations were off by 1, so i fixed this by completely ditching the concept of a window size, and kept track of how far i went using a second pointer called `right` . Then i was able to just calculate the window size with the simple formula: `right - left + 1`

- I was also having issues with the accumulative sum because i noticed the solution that `t` can be an extremely large number, and so I had to represent this with a `long long` instead of an `int`. For next time i should be more careful with the numeral data types and place them more intentionally.