# Problem 2:

**Pre-Implementation Comments:**

Essentially, this problem can be solved by identifying which indexes are most commonly accessed and how many times.

so if you're given the range values `[1, 2]` `[1, 3]` `[2, 3]`

We can visualise this as

| | | |
|---|---|---|
| X | X | |
| X | X | X |
| | X | X |

And so the 2nd index is accessed most, hence it would be strategic to assign our highest value to this cell.

We can do so by implementing a prefix sum after assinging array values as `array_index[left] += 1` and `array_index[right + 1] -= 1` so with these values: `[1, 2]` `[1, 3]` `[2, 3]`

We would have

| | | | |
|---|---|---|---|
| 1 | | -1 | |
| 1 | | | -1 |
| | 1 | | -1 |

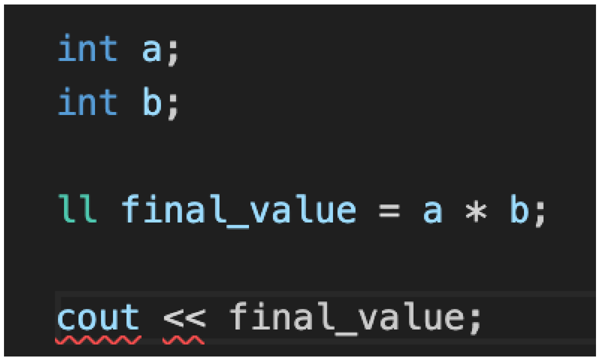
Then the prefix sum would look like:

| | | |
|---|---|---|
| 2 | 3 | 2 |

Which is exactly the access times. Then I can use a priority queue for highest accessed index values and highest array values and assign them with each other.

**Pre-Implementation Comments:**

I was having a problem with test case 7, and realised that in my code snippet i had something like the right, where i was multiplying two integers and assigning that to a long long but it would have already been overflowed by the time it was assigned. This was fixed by changing `a` and `b` into long longs as well.



```
int a;
int b;

ll final_value = a * b;

cout << final_value;
```