```
class Solution {
    public int leastInterval(char[] tasks, int n) {
        char[] ts = new char[26];
        for (char c : tasks) {
            ts[c - 'A']++;
        }
        Arrays.sort(ts);
        int i = 25;
        while (i >= 0 && ts[i] == ts[25]) {
            i--;
        }
        return Math.max(tasks.length, (ts[25] - 1) * (n + 1) + 25 - i);

        // return tasks.length when this:
        // A, B, C, D, E, E and n = 1
    }
}
```

Given a char array representing tasks CPU need to do. It contains capital letters A to Z where different letters represent different tasks. Tasks could be done without original order. Each task could be done in one interval. For each interval, CPU could finish one task or just be idle.

However, there is a non-negative cooling interval **n** that means between two **same tasks**, there must be at least n intervals that CPU are doing different tasks or just be idle.

You need to return the **least** number of intervals the CPU will take to finish all the given tasks.

Example:

Input: tasks = ["A","A","A","B","B","B"], n = 2
Output: 8
Explanation: A -> B -> idle -> A -> B -> idle -> A -> B.