Question: https://leetcode.com/problems/pairs-of-songs-with-total-durations-divisible-by-60/

So we need to check for pairs divisible by 60, its quite similar to 2 sum.

Here we need to store 60-(time[i]%60) as the key(what 60-(time[i]%60) gives is the number needed to be added to time[i] to make it totally divisible by 60) and it’s frequency as value.

Thus if we find 60-(time[i]%60) existing in map with value n, then there must have been n elements which can add up to the present element to make it a multiple of 60.

Code:  
class Solution {

public int numPairsDivisibleBy60(int[] time) {

HashMap<Integer, Integer> mem = new HashMap<Integer, Integer>();

int count=0;

for(int i=0; i<time.length; i++){

int val = 60-(time[i]%60);

int findKey = 0;

if(val!=60){

findKey=val;

}

count+=mem.getOrDefault(findKey,0);

mem.put(time[i]%60,mem.getOrDefault(time[i]%60,0)+1);

}

return count;

}

}

Github Link :<https://lnkd.in/ecwtJeaz>