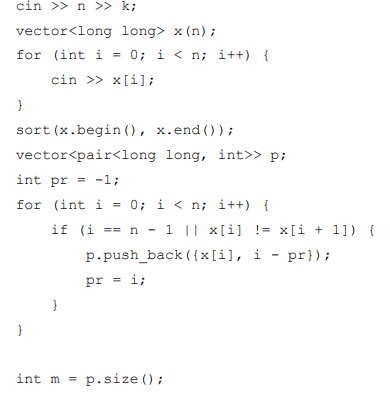
**Hypershock score**

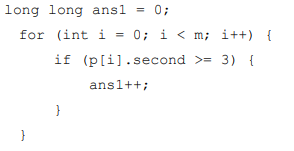
Let's consider three cases of scoring in a game: 1- all three players scored an equal number of points, 2- all three players scored different numbers of points in pairs, 3- all two players out of three scored an equal number of points, and the third player had a different number of points from those two.

We will learn how to find the number of scoring variants in the game in each of the three cases. The resulting values must be added together. Note that in subtask 1, the condition is met that k = 1, and therefore only the first case occurs, and in subtask 3, all numbers are different, and therefore only the second case occurs. You can use this to write partial solutions for these subtasks.

Put the input data in an array and sort it. One pass through this array will replace the numbers with pairs (xi, ci), where ci is the number how many times the value of xi occurs on the cards for Andrew, xi in all pairs are different. Note that the array of pairs is sorted by xi.



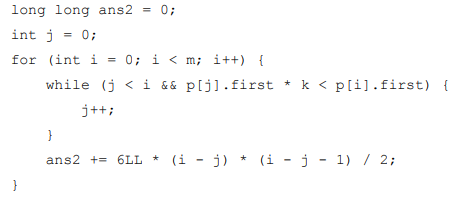
Andrey can show a score in which all three players scored an equal number of points if he has at least three cards with this number of points. Therefore, the number of such variants for which the score can be shown is equal to the number of numbers that occur at least 3 times. Below is the C++ code that counts the number of such variants.



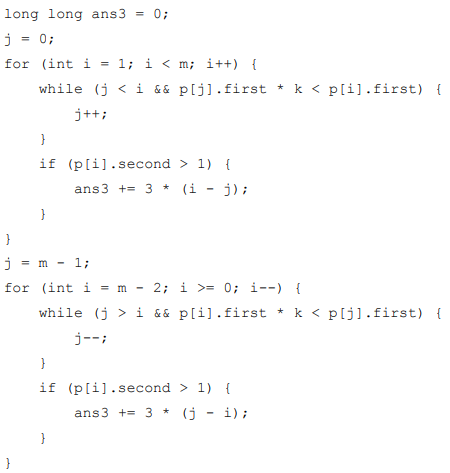
To determine the number of scoring variants in the game in the other two cases, you need to use the *two-pointer method*. To determine the number of scoring variants in which the scores of all players are different, do the following. For each variant of the maximum of three players ' points, we will find the minimum value of the other player's points that can be shown, and it does not violate the condition that the points differ by no more than k times. Let the maximum value of xi be the corresponding minimum value of xj . Then two other points values can be selected among the values xj, xj+1,..., xj-1, i.e. there are *(i – j) (i – j – 1) / 2* ways to do this. Since 3 different numbers can be ordered in 6 ways, this number must be multiplied by 6.

We sum up these values for all *i*. Note that when *i* increase, the value of *j* can also only increase, so the pass takes O(n).

Here is the C++ code that counts this number of variants.



Finally, a similar method is applicable for counting the number of scoring variants when two players scored an equal number of points and the third player scored a different number of points.



To solve subtask 1, it is enough to consider the variant that all players have the same number of points. Note that the additional restriction 1 ≤ xi ≤ 100 000 allows you to do without sorting pairs or complex data structures and remember the number of cards with each number in the array.

To solve subtask 2, you can go through all the three cards, mark for each of scoring variants in a three-dimensional array whether it is possible to get this account, and output the number of variants that can be obtained.

To solve subtask 3, you need to implement the solution described above, but you do not need to analyze the case when two players scored an equal number of points.