Let's say that: n - no of elements in A,
A - the list of elements,
sum - the sum of elements from A.

The problem is: divide A into 2 subsets (B and C) such that: $B \subset A$, $C \subset A$, $B \cap C = \emptyset$, $B \cup C = A$ (if A has [1, 2, 2, 3], the first 2 "differs" from the second one), and

$$\frac{\Sigma B}{cond(B)} = \frac{\Sigma C}{cond(C)} = \frac{\Sigma A}{n} = \frac{sum}{n}$$

~~$x \in A$~~ $A \subset \mathbb{Z}$ - only integers! $\Rightarrow \Sigma B \in \mathbb{Z}$
$\Sigma C \in \mathbb{Z}$
$sum \in \mathbb{Z}$

$$\Rightarrow \Sigma B = \frac{sum \cdot cond(B)}{n} \in \mathbb{Z}$$

$\Rightarrow$ the nr of elements in B times the sum of ~~the~~ A must be divisible with n

$\Rightarrow$ $sum \cdot cond(B) \vdots n$!!!

analog pentru C

i) $sum \cdot cond(C) \vdots n$

$\Rightarrow$ if we can not find a subset B such that i) is satisfied $\Rightarrow$ ~~there~~ there is no ~~a~~ such arrangement for the problem

$\Rightarrow$ return false.

if we can find such number, proceed.

~~We have to check~~ the nr. of
We suppose that B has elements in range $(1 - n/2)$ because if B has $n/2 + 5$ elem, then C will have $n/2 - 5$ elems and will be in the same range. So we have to check only that range.

This problem requires dynamic approach. So we create a vector of ~~sums~~ sets, call it sums (set<int> sums[n/2+1], which has $n/2+1$ elements.

The first "row" contains only value 0, the second row will contain (in the end) every element in A, the third will contain every sum possible between pairs of elements in A and so on.

0	0
1	...
2	...
3	...
⋮	
n/2	...

Let's say that A is [1, 2, 3, 4, 5, 6, 7, 8].

$n = 8$

sum = 36.

0	0						
1	1	2	3	4	5		
2	3	4	5	6	7	8	9
3	6	7	8	9	10	11	12
4	10	11	12	13	14		

$\frac{9}{2} = \frac{36}{8}$

=> return true

First insert the element 1 in second row

insert the second element in the second column, but in the third add it with every element found in the second row

insert the third element in second row, add it with every element from the second first and insert in the third, then sum it with every element in the third and insert in fourth.

(the order is exactly the opposite, so add it with every element in the second to last row and insert it in the last row, then work all the way up to top row)

We use sets because if the sum was already inserted, that means it can be achieved and we don't care anymore.

Every time we add a sum, check if that element is equal with the index of the row * sum / 4 AND if (index * sum) is divisible by 4. if it is, return true.

if no such element was not found, return false.