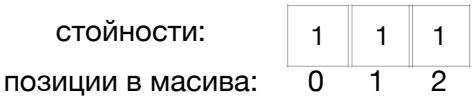
Задача [Чифтове]
Да се напише програма, която намира всички чифтове в даден масив, чиято сума е равна на подадено от входа число. На първия ред ще получите желаната сума, а на втория ред – масива от числа разделени с интервали.

Примерен вход 0	Очакван изход 0	Примерен вход 1	Очакван изход 1
6 1 2 3 4 3 6 5 8 3 9	1, 5 2, 4 3, 3 3, 3 3, 3	6 0 5 0 3 6 2 1 6	0, 6 0, 6 0, 6 0, 6 1, 5
Примерен вход 2	**Очакван изход 2**	**Примерен вход 3**	**Очакван изход 3**
6 0 5 3 3 0 3 1 6	0, 6 0, 6 1, 5 3, 3 3, 3 3, 3	2 1 1 1 1	1, 1 1, 1 1, 1 1, 1 1, 1 1, 1

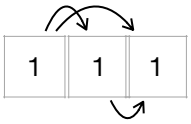
Описание на алгоритъма:

Да разгледаме следния масив от числа:



и желана сума **targetSum = 2**

Очевидно всеки един елемент от масива може да се комбинира с всеки друг елемент и сумата им да е равна на желаната. Това са общо:



три комбинации, съответно на позиции
0 и 1, 0 и 2, 1 и 2

Колко комбинации ще имаме ако единиците в масива бяха например **5**, а ако бяха **K**? Тогава щяхме да комбинираме единицата на позиция 0 с всички останали единици след нея, единицата на позиция 1 с всички останали единици след нея и т.н. докато не се изчерпат всички комбинации с груба сила. Но прилагайки тази груба сила, може да забележим, че винаги когато имаме непрекъсната поредица от **K** еднакви елемента, който са равни на половината от желаната сума, те ще сформират точно **(K - 1) + (K - 2) + ... + 1** чифта, които ще се сумират до желаната сума. За тази сума, която получихме като функция на **K** имаме формула за пресмятане (за константно време). Това е просто сбора на всички естествени числа от **1** до **K - 1**, който е равен на

$$\frac{K \times (K-1)}{2}$$

Това е първото важно наблюдение.

Нека сега разгледаме следния масив от числа:

стойности:	0	0	1	1
позиции в масива:	0	1	2	3

и желана сума **targetSum = 1**

Този път, повтарящите се елементи не се сумират до желаната сума, но комбинирайки се с други елементи (които отново се повтарят) се получава желаната сума. В такъв случай всяка една от нулите може да я съберем с всяка една от единичките и да получим желаната сума. Това ще са общо 4 чифта или казано по друг начин 2 x 2 = 4 комбинации.

Второ важно наблюдение: Когато имаме два елемента от масива със сума равна на желаната сума, то броя на всички чифтове, които ще генерират ще е равен на броя на срещания на единя елемент умножен по броя на срещания на другия елемент.

След като вече сме забелязали тези два полезни факта, лесно може да си дадем сметка, че всичко което ни е необходимо за тези пресмятания е да имаме масив **unique** от всички уникални елементи на оригиналния масив **original**, както и броя на срещанията на всеки елемент от масива **unique** (например в друг масив **repetitions** може да съхраняваме на позиция i → броя на срещанията на i-тия елемент от масива **unique**)

За да конструираме масива **unique** е необходимо първо да сортираме оригиналния масив, за да може повтарящите се елементи да застанат един до друг (не ни интересува дали сортирането ще е стабилно или не → и двата варианта ни вършат работа).

След като имаме сортирания масив от уникални елементи и масива от съответните повтаряния за всеки елемент, може да прибегнем до two pointer техниката, при която единя показалец е в началото, а другия – в края на масива. В нашия случай тези показатели ще са просто индекси на масива **unique**.

Алгоритъм: докато левия индекс **left** е по-малък от десния индекс **right**:

1. ако елементите на индекси left и right са със сума **равна** на желаната сума принтитаме **repetitions[left] * repetitions[right]** пъти чифта **unique[left] * unique[right]**
2. ако елементите на индекси left и right са със сума **по-малка** на желаната сума, инкрементираме левия индекс
3. ако елементите на индекси left и right са със сума **по-голяма** на желаната сума, декрементираме десния индекс

Освен това има още два недоразгледани ъглови случая:

- докато прилагаме алгоритъма описан по-горе трябва да проверяваме дали на някой индекс **left** или **right** ще се появи елемент, който е равен на половината от желаната сума. В такъв случай той може да се събере със себе си и съгласно първото наблюдение да направи чифт. Това ще е възможно само, ако се повтаря повече от 1 път в оригиналния масив и лесно може да го проверим чрез масива **repetitions**, който вече имаме. Ако да речем елемента на левия индекс е равен на половината от желаната сума и се среща **repetitions[left] = k** пъти, тогава принтираме $k \times (k-1)/2$ пъти чифта **{ unique[left], unique[left] }**, анагогично и ако елемента на десния индекс е бил равен на половината от желаната сума;
- На входа може да ни е подаден масив само от еднакви елементи (както е в първия случай най-отгоре). Тогава масива **unique** ще е с дължина 1 и индексите на показалците ще са съответно **left = 0** и **right = 0** и условието **left < right** никога няма да бъде изпълнено и няма да влезем в „докато“ (while) цикъла. Затова може да проверим този случай с един if преди алгоритъма.