**Чубакка и гравитационные колодцы**

*"Тысячелетний сокол" капитана Соло преодолевает 35 тысяч парсеков за считанные дни. Однако гранд-адмирал Империи Траун активизировал "звездные разрушители", оснащенные генераторами гравитационных колодцев, которые формируют в гиперпространстве гравитационные тени-помехи.*

*Чтобы оперативно выявлять подобные космические аномалии, Чубакке требуется срочно пропатчить бортовой софт "Сокола" -- реализовать специальную трансформирующую трансформацию, которая должна за разумное время преобразовывать входной массив карты гиперпрыжка и выявлять её ключевой ключ -- секретное значение, четность которого свидетельствует о наличии на маршруте имперских помех.*

На входе одномерный массив A (индексация начинается с нуля) случайных положительных целых чисел (возможно, повторяющихся).

Трансформирующая трансформация S массива A происходит так:

0. Выходной массив/список B исходно пуст.

1. Организуем первый цикл для i, начиная с 0 и до длины массива A минус 1.

2. Организуем внутри первого цикла второй цикл для j, начиная с 0 и до длины массива A минус i минус 1.

3. Рассчитываем значение k = i + j.

4. Из поддиапазона A[ j..k ] выбираем максимальное значение и записываем в конец списка B.

По завершении всех циклов список B считаем трансформирующей трансформацией, которую назовём S(A).  
Говоря программистским языком, есть функция S(A), которая возвращает список B.

5. Ключевой ключ находится суммированием всех элементов списка, получаемого двойной трансформацией S(S(A)) (результат первой трансформации подаётся на вход второй трансформации).

Реализуйте алгоритм поиска ключевого ключа и выводите true, если он чётный.

Функция

boolean TransformTransform(int A[], int N)

получает на вход массив из N (N >= 1) целых положительных чисел и возвращает true, если сумма всех значений результата двойной трансформации A чётная.

[как постить решение](https://skillsmart.ru/algo/lvl1/how.html)

ниже рабочее решение:

import java.util.ArrayList;  
  
  
public class Transform {  
  
 public static boolean TransformTransform(int A[], int N){  
 int checkSum = 0;  
  
 ArrayList<Integer> list1 = new ArrayList<>();  
 ArrayList<Integer> list2 = new ArrayList<>();  
 ArrayList<Integer> list3 = new ArrayList<>();  
 for (int b = 0; b < A.length; b++){  
 list1.add(A[b]);  
 }  
  
 System.*out*.println("list1: " + list1);  
  
 for (int c = 0; c < list1.size(); c++){  
 checkSum = checkSum + list1.get(c);  
 }  
  
 System.*out*.println("Checksum1 = " + checkSum);  
  
 list2 = *transform1*(list1);  
  
 System.*out*.println("list2: " + list2);  
  
 for (int c = 0; c < list2.size(); c++){  
 checkSum = checkSum + list2.get(c);  
 }  
  
 System.*out*.println("Checksum2 = " + checkSum);  
  
 list3 = *transform1*(*transform1*(list1));  
  
 System.*out*.println("list3: " + list3);  
  
 for (int c = 0; c < list3.size(); c++){  
 checkSum = checkSum + list3.get(c);  
 }  
  
 System.*out*.println("Checksum3 = " + checkSum);  
  
 if (checkSum % 2 == 0) {  
 return true;  
 }  
 else return false;  
  
 }  
  
 public static ArrayList<Integer> transform1(ArrayList<Integer> inputList){  
 int k;  
 int max;  
 ArrayList<Integer> transformedList = new ArrayList<>();  
  
 for (int i = 0; i <= inputList.size() - 1; i++){  
 for (int j = 0; j <= inputList.size() - i - 1; j++){  
 k = i + j;  
 if (k - j == 0) {  
 transformedList.add(inputList.get(0));  
 }  
 else {  
 System.*out*.println("range: " + j + " - " + k);  
 max = inputList.get(j);  
 for (int a = j; a <= k; a++){  
 if(inputList.get(a) > max) max = inputList.get(a);  
 }  
 transformedList.add(max);  
 }  
 }  
  
 }  
  
 return transformedList;  
 }  
  
  
  
 public static void main(String[] args) {  
  
 int[] testArr = {1,2,3};  
  
  
 System.*out*.println(*TransformTransform*(testArr, 0));  
  
  
 }  
}