Dividing coins Problem 2 (0 / 0)
Magdalena and Hristina try to divide a bag of coins between the two of them. The contents of the bag appears to be not equally divisible. Magdalena and Hristina want to make an equal division as fair as possible between them. Your help is asked to solve this problem. .

Given a bag with a maximum of 100 coins, determine the most fair division between the two of them. This means that the difference between the amount each person obtains should be minimized. The value of a coin varies from 1 cent to 500 denars. It's not allowed to split a single coin.

Input: The first line in the input is the number of coins m, 1≤m≤100, in the bag, then in the second line m numbers, separated by one space, are given, each number indicates the value of a coin.

Output: The minimal positive difference between the amount Magdalena and Hristina obtain when they divide the coins from the corresponding bag is given.

Example:

Input:

4

1 2 4 6

Output:

1

Partial solution: The problem is considered to be partially solved if 7 test cases are passed.

Class name (Java): PodelbaParicki

Playful list Problem 1 (0 / 0)
There is a double linked list containing characters (each node contains one character). You should do the following until there is only one node left in the list: starting from the beginning of the list each second node should be deleted (ie. 2nd, 4th, 6th, etc), then starting from the end of the list each second node should be deleted (ie. the one before the last, 2 nodes before that, etc). You should repeat this (deleting from the beginning, then from the end) in each iteration, until there is only one node in the list left.

Input: One line of input containing the elements of the list (characters separated by a blank space).

Output: You should print the value of the remaining node.

Example: The list a b c d e f g h i j is given.
The list after the first deletion (from the beginning): a c e g i
The list after the second deletion (from the end): a e i
The list after the third deletion (from the beginning): a i
The list after the fourth deletion (from the end): i – There is only one node left, we stop here.

Partial solution: The problem is considered to be partially solved if 7 test cases are passed.

Notes: You MUST use the list structures defined in the problem and you cannot use additional structures as arrays etc. You can use only ONE list structure object.

Class name (Java): RazigranaLista

StackQueue Problem 3 (0 / 0)
Write an implementation of a new data structure called StackQueue which will be with limited memory.

The data structure should have the following methods:

InsertElement – Inserting an element should be done as adding an element to a stack. If the stack is full, then addition should be done to the queue. The complexity of this method should be O(1).
DeleteElement – Deletion of an element can be a deletion from a stack or from a queue. The element which is at the top of the stack and the element which is at the front of the queue are compared, and the one with the highest value should be deleted. If one of the stack or queue structure is empty, then deletion should be done from the data structure that has element left. The complexity of this method should be O(1).
IsEmpty – check if the data structure is empty
Peek - Return the element which is at the top of the structure or find the element which is the next for deletion as explained in DeleteElement method.
For the given method implementations, the existing stack and queue implementations could be used. The data structure is limited, so the stack and queue size should be given while creation. Elements are inserted in the order as they are given at input. You should print the elements from the data structure in the order as they should be deleted.

Input: In the first line at the input the stack size is given, in the second the queue size. In the third line, the number of elements is given and after that elements which should be inserted in each line.

Output: First the element from the top of the structure StackQueue should be printed. After that in a new line the data structure elements are printed in the order as they should be deleted, separated by a blank space.

Example:

Input:

3 (size of the stack)
5 (size of the queue)
8 (number of elements that should be read)
0 (element 0)
2 (element 1)
4
1
3
5
7
9
Output:

4 (element on the top of the structure)
4 2 1 3 5 7 9 0 (all the elements from the structure StackQueue)
Partial Solution: The problem is partially solved if it passes 3 test cases.

Note: You don't need to implement the new data structure as Generic type. Because of the nature of the problem the assistants will keep the right to look into the codes event all the test cases are passed.

Class name (Java): TestSteckQueue