Комплект олимпиадных задач регионального этапа Всероссийской олимпиады школьников по информатике 2012/2013 учебного года

Для проведения регионального этапа всероссийской олимпиады школьников по информатике Центральная предметно-методическая комиссия по информатике разработала комплект из восьми задач. Этот комплект задач являлся единым для всех участников регионального этапа, независимо от класса, в котором они обучались.

Все задачи пронумерованы от 1 до 8. Задачи с 1-й по 4-ю включительно использовались на первом туре, а задачи с 5-й по 8-ю включительно – на втором туре.

Далее представлены тексты всех задач регионального этапа Всероссийской олимпиады школьников по информатике 2012/2013 учебного года.

Задача 1. «Кастинг»

Имя входного файла:	`casting.in`
Имя выходного файла:	`casting.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 Мбайт

В театре работает n актеров. Известно, что среди них a – высоких, b – голубоглазых и c – блондинов. Для главной роли в новом спектакле режиссеру требуется только один высокий голубоглазый блондин. Чтобы спланировать свое время для беседы с каждым таким артистом из труппы театра, режиссеру необходимо узнать, какое максимальное или какое минимальное количество артистов из работающих в театре подходит для этой роли.

Требуется написать программу, которая по заданным числам n, a, b и c определяет минимальное или максимальное количество актеров, с которыми режиссер должен переговорить.

Формат входного файла

Первая строка входного файла содержит одно число, которое задает, минимальное или максимальное количество актеров необходимо найти в данном тесте. Это число может принимать следующие значения:

- 1, если в данном тесте требуется определить минимальное количество актеров;
- 2, если в данном тесте требуется определить максимальное количество актеров.

Вторая строка входного файла содержит разделенные пробелами четыре целых числа: n, a, b, c ($1 \le n \le 10\,000$, $0 \le a \le n$, $0 \le b \le n$, $0 \le c \le n$).

Формат выходного файла

Выходной файл должен содержать одно число – минимальное или максимальное (в зависимости от входных данных) количество актеров, которые могут претендовать на главную роль в новом спектакле.

Пример входных и выходных файлов

casting.in	casting.out
2 5 3 4 5	3
1 5 3 4 5	2

Пояснения к примерам

В первом примере, поскольку высоких актеров всего трое, то на главную роль не может подойти больше трех человек.

Во втором примере все актеры – блондины и все, кроме одного, – голубоглазые. Тогда среди трех высоких актеров найдутся хотя бы два голубоглазых (и, естественно, они

будут блондинами). Следовательно, минимум два актера точно подойдут на главную роль в новом спектакле.

Система оценивания

Правильные решения для тестов, в которых требуется найти минимальное количество актеров, будут оцениваться из 50 баллов.

Правильные решения для тестов, в которых требуется найти максимальное количество актеров, будут оцениваться из 50 баллов.

Несмотря на выделение отдельных групп тестов для минимального и максимального количества артистов, на окончательную проверку будут приниматься только решения, правильно работающие для всех тестов из условия задачи.

Задача 2. «Города»

Имя входного файла:	`cities.in`
Имя выходного файла:	`cities.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 Мбайт

Юный программист решил придумать собственную игру. Игра происходит на поле размером $N \times N$ клеток, в некоторых клетках которого расположены города (каждый город занимает одну клетку; в каждой клетке может располагаться не более одного города). Всего должно быть ***чётное*** количество городов.

Изначально про каждую клетку игрового поля известно, расположен ли в ней город или нет. Чтобы начать игру, необходимо разделить игровое поле на два государства так, чтобы в каждом государстве было ***поровну*** клеток-городов.

Граница между государствами должна проходить по границам клеток таким образом, чтобы из любой клетки каждого государства существовал путь по клеткам этого же государства в любую другую его клетку (из клетки можно перейти в соседнюю, если они имеют общую сторону). Каждая клетка игрового поля должна принадлежать только одному из двух государств, при этом государства не обязаны состоять из одинакового количества клеток.

Требуется написать программу, которая с учетом сказанного разделит клетки заданного игрового поля между двумя государствами.

Формат входного файла

Первая строка входного файла содержит одно целое положительное число N, задающее размер игрового поля ($1 \le N \le 50$).

Последующие N строк содержат по N заглавных латинских букв (без пробелов), кодирующих соответствующие клетки игрового поля: ‘C’ обозначает клетку, занятую городом, ‘D’ – пустую клетку. Гарантируется, что на поле есть хотя бы два города и всего их четное число.

Формат выходного файла

Выходной файл должен содержать N строк по N цифр (без пробелов) в каждой, кодирующих соответствующие клетки. Цифра 1 обозначает, что данная клетка принадлежит первому государству, цифра 2 – данная клетка принадлежит второму государству.

Если решений несколько, необходимо вывести любое из них.

Примеры входных и выходных файлов

cities.in	cities.out
3 DDD DDC DDC	222 212 211
5 DDDDD CDCDC DCCDC DDDDD DDDDD	11111 12221 12221 11111 11111

Система оценивания

Правильные решения для тестов, в которых всего два города, будут оцениваться из 40 баллов.

Несмотря на выделение отдельной группы тестов с двумя городами, на окончательную проверку будут приниматься только решения, правильно работающие также для всех тестов из условия задачи.

Задача 3. «A+B=C»

Имя входного файла:	`aplusb.in`
Имя выходного файла:	`aplusb.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 Мбайт

Часто для пробного тура на различных олимпиадах по информатике предлагается задача «A + B», в которой по заданным целым числам A и B требуется найти их сумму.

При проведении городской олимпиады по информатике председатель жюри решил сам подготовить тесты для такой задачи. Для этого он использовал свою оригинальную методику, которая заключалась в следующем: сначала готовятся предполагаемые правильные ответы, а затем подбираются входные данные, соответствующие этим ответам.

Пусть председатель жюри выбрал число C, запись которого состоит из n десятичных цифр и не начинается с нуля. Теперь он хочет подобрать такие целые положительные числа A и B, чтобы их сумма была равна C, и запись каждого из них также состояла из n десятичных цифр и не начиналась с нуля. В дополнение к этому председатель жюри старается подобрать такие числа A и B, чтобы каждое из них было *красивым*. Красивым в его понимании является число, запись которого не содержит двух одинаковых подряд идущих цифр. Например, число 1272 считается красивым, а число 1227 — нет.

Требуется написать программу, которая для заданного натурального числа C вычисляет количество пар красивых положительных чисел A и B, сумма которых равна C. Поскольку количество пар красивых чисел может быть большим, необходимо вывести остаток от деления этого количества на число 10^9+7.

Формат входного файла

Входной файл содержит одно целое положительное число C. Число C не начинается с нуля. Количество цифр в записи числа C не превышает 10 000.

Формат выходного файла

Выходной файл должен содержать одно целое число — остаток от деления количества искомых пар красивых чисел A и B на число 10^9+7.

Примеры входных и выходных файлов

aplusb.in	aplusb.out
22	2
200	0
1000	0
239	16

Пояснения к примерам

Число 22 можно представить в виде суммы двузначных чисел тремя способами: 10 + 12, 11 + 11, 12 + 10. Способ 11 + 11 не подходит, поскольку число 11 не является красивым. Следовательно, ответ для числа 22 равен 2.

Число 200 можно представить в виде суммы трехзначных чисел единственным способом: 100 + 100. Этот способ не подходит, поэтому ответ для числа 200 равен 0.

Число 1000 нельзя представить в виде суммы четырехзначных чисел, поэтому ответ для числа 1000 аналогично равен 0.

Система оценивания

Правильные решения для тестов, в которых $1 \le C \le 999$ ($1 \le n \le 3$), будут оцениваться из 25 баллов.

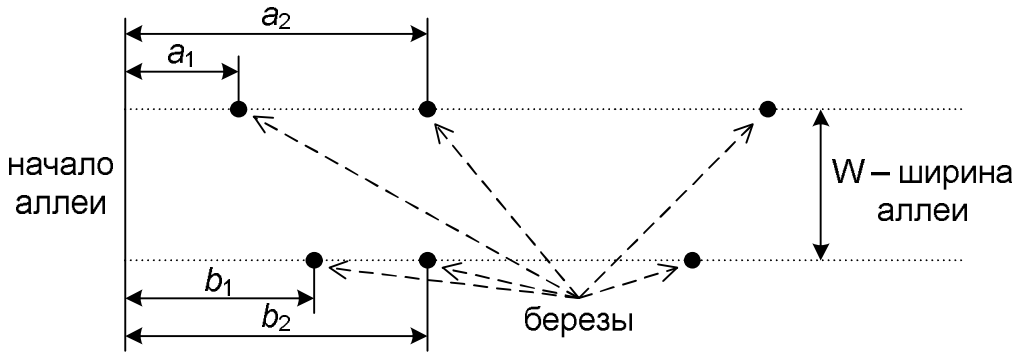
Правильные решения для тестов, в которых $1 \le C \le 999\,999$ ($1 \le n \le 6$), будут оцениваться из 50 баллов.

Несмотря на выделение отдельных групп тестов для различных длин числа C, на окончательную проверку будут приниматься только решения, правильно работающие для всех тестов из условия задачи.

Задача 4. «Березовая аллея»

Имя входного файла:	`birch.in`
Имя выходного файла:	`birch.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	512 Мбайт

На краю деревни растет старая березовая аллея. Аллея имеет форму прямой полосы шириной W метров. Вдоль левой стороны аллеи растет N берез, а вдоль правой — M берез, при этом i-я береза с левой стороны аллеи находится на расстоянии a_i метров от начала аллеи, а j-я береза с правой стороны — на расстоянии b_j метров от начала аллеи.



Отдыхая в деревне прошедшим летом, один юный информатик обнаружил, что кору берез стали грызть зайцы. Чтобы защитить деревья от зайцев, мальчик решил окружить березы красной лентой (зайцы не любят красный цвет и не станут заходить на огражденную лентой территорию). К сожалению, в его распоряжении оказалась только лента длиной L метров, которую, к тому же, нельзя было разрезать. Единственное, что можно было делать в этом случае — окружить этой лентой как можно больше берез. При этом, чтобы сохранить аллею, необходимо окружить на каждой стороне аллеи хотя бы одну березу.

Требуется написать программу, которая по заданной длине ленты, ширине аллеи и положению берез на ней определяет максимальное число берез, которое можно окружить этой лентой. Считается, что березы представляются точками, толщиной берез и шириной ленты следует пренебречь.

Формат входного файла

Первая строка входного файла содержит два разделенных пробелом целых числа: длину ленты L и ширину аллеи W ($1 \le L \le 2\times10^5$, $1 \le W \le 10^4$).

Вторая и третья строки описывают березы вдоль левой стороны аллеи. Вторая строка содержит число N — количество берез ($1 \le N \le 2000$), а третья строка содержит N различных целых чисел $a_1, a_2, \ldots, a_N$, заданных по возрастанию ($0 \le a_i \le 10^5$).

Четвертая и пятая строки описывают березы вдоль правой стороны аллеи. Четвертая строка содержит число M — количество берез ($1 \le M \le 2000$), а пятая строка содержит M различных целых чисел $b_1, b_2, \ldots, b_M$, заданных по возрастанию ($0 \le b_i \le 10^5$).

Формат выходного файла

Выходной файл должен содержать одно целое число: максимальное количество берез, которое можно оградить заданной лентой.

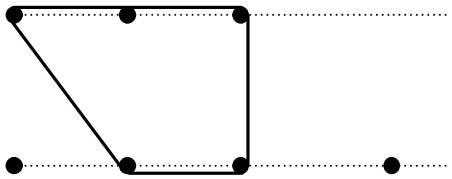
Гарантируется, что если максимальное число берез, которое можно оградить лентой длины L, равно X, то нет способа оградить $(X + 1)$ березу лентой длины $(L + 10^{-5})$.

Примеры входных и выходных файлов

birch.in	birch.out
18 4 3 0 3 6 4 0 3 6 10	5
5 3 1 0 1 0	0

Пояснения к примерам

В первом примере можно, например, оградить березы способом, указанным на рисунке ниже.



Во втором примере длины ленты недостаточно чтобы оградить хотя бы по одной березе с каждой стороны.

Система оценивания

Правильные решения для тестов, в которых $1 \le (N + M) \le 50$, будут оцениваться из 30 баллов.

Правильные решения для тестов, в которых $1 \le (N + M) \le 500$, будут оцениваться из 60 баллов.

Задача 5 «Игральные кубики»

Имя входного файла:	`dices.in`
Имя выходного файла:	`dices.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 Мбайт

Юный математик Матвей интересуется теорией вероятностей, и по этой причине у него всегда есть с собой несколько стандартных шестигранных игральных кубиков. Стандартный шестигранный кубик имеет три противолежащих пары граней, которые размечены таким образом, что напротив грани с числом 1 находится грань с числом 6, напротив грани с числом 2 — грань с числом 5 и напротив грани с числом 3 — грань с числом 4.

Анализируя различные игры с шестигранными кубиками, Матвей придумал новую игру. В эту игру играют два игрока, и проходит она следующим образом: первый игрок бросает один или несколько стандартных кубиков (количество кубиков он определяет сам). После этого первому игроку начисляется количество очков, равное сумме чисел, оказавшихся на верхних гранях всех кубиков, а второму игроку — сумма чисел, оказавшихся на нижних гранях этих кубиков. Побеждает тот, кто набрал больше очков.

Например, если был брошен один кубик, и на верхней его грани выпало число два, то первый игрок получает два очка, а второй — пять. В свою очередь, если было брошено два кубика и на их верхних гранях выпало по единице, то первый игрок получает также два очка, а второй игрок – двенадцать очков, так как на нижних гранях этих кубиков оказались шестерки.

Матвей рассказал об этой игре своему другу, юному информатику Фоме, и они начали играть в неё через Интернет. Поскольку Фома не видит результат броска и не знает, сколько кубиков бросает Матвей как первый игрок, то о набранных каждым игроком очках он узнает только от Матвея. Чтобы проверить достоверность этой информации, Фома решил узнать, какое минимальное и максимальное количество очков мог получить он, как второй игрок, если известно, сколько очков набрал Матвей.

Требуется написать программу, которая по количеству очков, набранных первым игроком после броска, определяет наименьшее и наибольшее количество очков, которые может получить второй игрок за этот бросок.

Формат входного файла

Первая строка входного файла содержит целое положительное число n — количество очков, которые получил первый игрок ($1 \le n \le 10^{10}$).

Формат выходного файла

Выходной файл должен содержать два разделенных пробелом целых числа: минимальное и максимальное количество очков, соответственно, которые мог набрать второй игрок при таком броске кубиков.

Примеры входных и выходных файлов

dices.in	dices.out
2	5 12
36	6 216

Система оценивания

Правильные решения для тестов, в которых $1 \le n \le 1000$, будут оцениваться из 50 баллов.

Задача 6. «Имена»

Имя входного файла:	`name.in`
Имя выходного файла:	`name.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 Мбайт

На далекой планете Тау Кита есть непонятные нам обычаи. Например, таукитяне очень необычно для землян выбирают имена своим детям. Родители так выбирают имя ребенку, чтобы оно могло быть получено как удалением некоторого набора букв из имени отца, так и удалением некоторого набора букв из имени матери. Например, если отца зовут «`abacaba`», а мать — «`bbccaa`», то их ребенок может носить имена «`a`», «`bba`», «`bcaa`», но не может носить имена «`aaa`», «`ab`» или «`bbc`». Возможно, что имя ребенка совпадает с именем отца и/или матери, если оно может быть получено из имени другого родителя удалением нескольких (возможно, ни одной) букв.

Пусть отец по имени X и мать по имени Y выбирают имя своему новорожденному ребенку. Так как в таукитянских школах учеников часто вызывают к доске в лексикографическом порядке имен учеников, то есть в порядке следования имен в словаре, то они хотят выбрать своему ребенку такое имя, чтобы оно лексикографически следовало как можно позже.

Формально, строка S *лексикографически больше* строки T, если выполняется одно из двух условий:

- строка T получается из S удалением одной или более букв с конца строки S;
- первые (i - 1) символов строк T и S не различаются, а буква в i-й позиции строки T следует в алфавите раньше буквы в i-й позиции строки S.

Требуется написать программу, которая по именам отца и матери находит лексикографически наибольшее имя для их ребенка.

Формат входного файла

Первая строка входного файла содержит имя отца X. Вторая строка входного файла содержит имя матери Y. Каждое имя состоит из строчных букв латинского алфавита, включает хотя бы одну букву и имеет длину не более 10^5 букв.

Формат выходного файла

Выходной файл должен содержать искомое лексикографически наибольшее из возможных имен ребенка. В случае, если подходящего имени для ребенка не существует, выходной файл должен быть пустым.

Пример входных и выходных файлов

name.in	name.out
abcabca abcda	ca
ccba accbbaa	ccba

Пояснения к примеру

В первом примере имя ребенка не может начинаться с буквы большей c, так как имя отца не содержит таких букв. Буква c содержится в обоих именах, следовательно, имя ребенка может начинаться с этой буквы. Единственная буква, которая может идти следом за буквой c в имени ребенка — это буква a.

Система оценивания

Правильные решения для тестов, в которых имена содержат только буквы «a» и «b» и имеют длину не более 1000, будут оцениваться из 20 баллов.

Правильные решения для тестов, в которых имена содержат только буквы «a» и «b» и имеют длину не более 10^5, будут оцениваться из 40 баллов.

Правильные решения для тестов, в которых имена имеют длину не более 1000, будут оцениваться из 40 баллов.

Несмотря на выделение отдельных групп тестов, на окончательную проверку будут приниматься только решения, правильно работающие для всех тестов, приведенных в условии задачи.

Задача 7. «Две окружности»

Имя входного файла:	`circles.in`
Имя выходного файла:	`circles.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 Мбайт

Юный футболист Митя обнаружил на школьном футбольном поле две различные окружности, нарисованные едва заметной белой краской. Вспомнив истории о загадочных кругах на полях, он отметил эти окружности с помощью небольших камушков. Митя разложил на поле n камушков так, чтобы каждый из них находился на одной из окружностей или даже на их пересечении, если эти окружности пересекаются. Получилось так, что на каждой окружности размещался хотя бы один камушек. Обладая великолепным глазомером, Митя расположил камушки на окружностях абсолютно точно, без какой-либо погрешности.

На следующий день пошел дождь, краска стерлась, и нарисованные окружности исчезли, но все камушки остались на своих местах. Теперь Мите очень нужно найти доказательство необычного явления, свидетелем которого он был, то есть, восстановить окружности.

Требуется написать программу, которая по координатам камушков на поле находит вариант размещения их на двух несовпадающих окружностях.

Формат входного файла

Первая строка входного файла содержит целое число n — количество размещенных Митей камушков на поле ($2 \le n \le 2000$). Последующие n строк содержат целочисленные координаты камушков (x_i, y_i) — в каждой строке по одной паре координат, разделенных пробелом ($-10^6 \le x_i, y_i \le 10^6$).

Никакие два камушка не размещаются в одной точке. Гарантируется, что ответ для заданного набора камушков существует.

Формат выходного файла

Выходной файл должен содержать две строки. Первая строка должна содержать последовательность номеров *всех* камушков, которые принадлежат первой окружности, вторая строка — последовательность номеров *всех* камушков, которые принадлежат второй окружности. Считается, что камушки пронумерованы от 1 до n в порядке их следования во входных данных.

Каждый камушек должен встречаться хотя бы в одной из двух последовательностей. Если камушек встречается в обеих последовательностях, то это обозначает, что он находится на пересечении окружностей.

Нумерация окружностей не имеет значения, то есть, выводить две последовательности можно в любом порядке. Числа в последовательностях можно также выводить в произвольном порядке. Каждая из последовательностей должна содержать не менее одного числа. Все числа в строках должны быть разделены пробелами.

Если вариантов расположения окружностей несколько, можно выбрать любой из них.

Пример входных и выходных файлов

circles.in	circles.out
7 1 -1 0 0 1 1 3 1 3 -1 2 0 4 0	6 1 2 3 4 7 5 6
5 -1000000 0 0 1000000 1000000 0 0 -1000000 0 0	1 2 3 4 1 5

Пояснения к примеру

В первом примере одна из искомых окружностей имеет центр в точке с координатами (1, 0), а вторая — в точке с координатами (3, 0). Обе окружности имеют радиус равный 1.

Во втором примере центр первой окружности совпадает с началом координат, а радиус равен 10^6. Вторая окружность — любая, проходящая через точки (-10^6, 0) и (0, 0).

В обоих примерах возможны и другие правильные ответы.

Система оценивания

Правильные решения для тестов, в которых $2 \le n \le 50$, будут оцениваться из 50 баллов.

Задача 8. «Столицы»

Имя входного файла:	`capitals.in`
Имя выходного файла:	`capitals.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	512 Мбайт

В стране Триландии близятся выборы новых столиц. Столицы в Триландии необычные, поскольку ими являются одновременно сразу три различных города. Такая идея размещения столиц основана на исследованиях эффективности управления страной, выполненных ведущими экономистами Триландии.

Всего в Триландии n городов, из которых некоторые пары городов соединены дорогами, и по каждой из них можно проехать в обе стороны. Время проезда по каждой дороге в одну сторону равно одному часу. При этом все города соединены дорогами таким образом, что из каждого города можно добраться в любой другой, причем это можно сделать единственным способом, если по каждой дороге проезжать не более одного раза и только в одну сторону.

Как показали результаты проведенных триландскими экономистами исследований, управление страной будет наиболее эффективным, если три столицы будут выбраны так, что время проезда по кратчайшему пути между каждой парой столиц составит ровно d часов. Перед проведением выборов необходимо знать, сколько существует различных троек городов, удовлетворяющих описанным выше свойствам. Две тройки городов считаются различными, если в первой тройке есть хотя бы один город, которого нет во второй тройке, и наоборот.

Требуется написать программу, которая по количеству городов в Триландии и описанию дорог находит количество троек городов, которые могут быть столицами.

Формат входного файла

Первая строка входного файла содержит два разделенных пробелом целых числа: количество городов в Триландии n и требуемое время в пути между столицами d ($3 \le n \le 10^5$, $1 \le d < n$). Каждая из последующих $(n - 1)$ строк содержит описание одной дороги: пару разделенных пробелом различных целых чисел a_i и b_i — номера городов, которые соединены двусторонней дорогой ($1 \le a_i \le \mathrm{n}$, $1 \le b_i \le \mathrm{n}$, $a_i \ne \mathrm{b}_i$). Каждая пара городов соединена не более чем одной дорогой.

Формат выходного файла

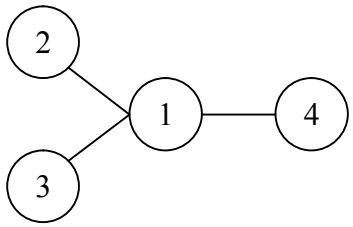
Выходной файл должен содержать одно целое число — количество подходящих троек городов, которые могут быть выбраны столицами. В случае, если нужных троек городов не окажется, выходной файл должен содержать ноль.

Пример входных и выходных файлов

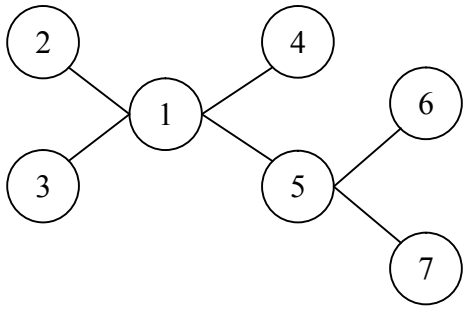
capitals.in	capitals.out
4 2 1 2 1 3 1 4	1
7 2 1 2 1 3 1 4 5 1 5 6 5 7	5

Пояснения к примерам

В первом примере существует единственный способ выбрать три столицы: города с номерами 2, 3 и 4. Рисунок, соответствующий первому примеру, приведен ниже.



Во втором примере существует четыре варианта выбора трёх столиц из четверки городов: 2, 3, 4 и 5. Можно также выбрать столицами города с номерами 1, 6 и 7. Рисунок, соответствующий второму примеру, приведен ниже.



Система оценивания

Правильные решения для тестов, в которых $3 \le n \le 50$, будут оцениваться из 20 баллов.

Правильные решения для тестов, в которых $3 \le n \le 500$, будут оцениваться из 40 баллов.

Правильные решения для тестов, в которых $3 \le n \le 5000$, будут оцениваться из 60 баллов.