Общая информация по задачам первого тура

Задача	**Тип задачи**	**Ограничения**
1. Распределенные системы	стандартная	1 с, 1024 МБ
2. Расследование в Темерии	стандартная	2 с, 1024 МБ
3. Скобки и деревья	интерактивная, двойной запуск	2 с, 1024 МБ
4. Спортивная тренировка	стандартная	2 с, 1024 МБ

Необходимо считывать данные из стандартного потока ввода. Выходные данные необходимо выводить в стандартный поток вывода.

Баллы за подзадачу начисляются только если все тесты этой и необходимых подзадач пройдены. Решение запускается на тестах для определенной подзадачи, если все тесты всех необходимых подзадач пройдены. В одной из задач можно получить частичные баллы за подзадачу. Для тестирования подзадачи достаточно, чтобы во всех необходимых подзадачах был получен положительный балл.

Во всех подзадачах каждой задачи во время тура вам показываются баллы за подзадачу, если все тесты пройдены, либо первая ошибка и номер теста.

Для некоторых подзадач может также требоваться, чтобы были пройдены все тесты из условия. Для таких подзадач указана дополнительно буква У.

Задача 1. Распределённые системы

Ограничение по времени: 1 секунда
Ограничение по памяти: 1024 мегабайта

В компании n серверов, пронумерованных числами от 1 до n. На i-м сервере запущены a_i сервисов.

Иногда серверы могут отключаться, поэтому для каждого сервера был определён резервный сервер. Для сервера с номером i резервным является сервер с номером p_i. Если у i-го сервера $p_i = i$, то это сервер повышенной надёжности, и он никогда не отключается.

Для любых двух различных серверов i и j номера их резервных серверов p_i и p_j не совпадают. Таким образом, p — это перестановка длины n, то есть каждое число от 1 до n встречается ровно один раз среди значений $p_1, \ldots, p_n$.

Процесс отключения сервера происходит следующим образом. Если сервер i отключается, то все запущенные на нём сервисы перемещаются на сервер с номером p_i, а сервер i заменяется на новый сервер, на котором не запущены никакие сервисы. Номер этого сервера и номер его резервного сервера остаются без изменений. Перенос сервисов и замена сервера очень быстрый процесс, во время него не может произойти новых отключений.

В компании планируется провести тестирование работоспособности системы. Для этого будут отключены не более k серверов. Отключения проводятся последовательно, то есть никакие два сервера не отключаются одновременно. Определите максимальное число сервисов, которые могут оказаться на одном сервере после отключения не более k серверов.

Формат входных данных

В первой строке заданы два целых числа n и k ($1 \leqslant k < n \leqslant 10^5$) — количество серверов, а также максимальное количество серверов, которые могут отключиться.

Во второй строке заданы n целых чисел $a_1, a_2, \ldots, a_n$ ($0 \leqslant a_i \leqslant 10^9$) — количество сервисов, запущенных на серверах.

В третьей строке заданы n целых чисел $p_1, p_2, \ldots, p_n$ ($1 \leqslant p_i \leqslant n$) — номера резервных серверов.

Формат выходных данных

Выведите одно целое число — ответ на задачу.

Система оценивания

Подзадача	Баллы	Дополнительные ограничения		Необх. подзадачи
		n	дополнительно	
1	15	$n \leqslant 1000$	$k = 1$	–
2	27	$n \leqslant 1000$	–	У, 1
3	21	–	$p_i = i \bmod n + 1$	–
4	37	–	–	У, 1, 2, 3

Примеры

стандартный ввод	стандартный вывод
4 2 6 10 7 9 2 3 4 1	26
3 1 1000000000 993 2010 1 3 2	1000000000
11 5 3 5 12 7 5 9 2 6 0 9 4 2 8 9 6 5 11 3 1 10 7 4	23

Замечание

Рассмотрим порядок отключений серверов, который позволяет достичь максимальный ответ в первом примере.

Напомним, как осуществляются переносы сервисов при отключении серверов.

Сервер	1	2	3	4
Резерв	2	3	4	1

Первым отключается второй сервер, его сервисы переходят на третий сервер, то есть теперь на третьем сервере $10 + 7 = 17$ сервисов.

Вторым отключается третий сервер, его сервисы переходят на четвёртый сервер, после этого на четвёртом сервере $9 + 17 = 26$ сервисов.

Для лучшего понимания смотрите таблицу, в которой записано количество сервисов на каждом из серверов в ходе процесса, описанного выше.

Стадия	a_1	a_2	a_3	a_4
До первого отключения	6	10	7	9
После отключения сервера 2	6	0	17	9
После отключения сервера 3	6	0	0	26

Если бы первым отключился третий сервер, а вторым — второй, то процесс выглядел бы так.

Стадия	a_1	a_2	a_3	a_4
До первого отключения	6	10	7	9
После отключения сервера 3	6	10	0	16
После отключения сервера 2	6	0	10	16

При этом максимальное количество сервисов на сервере было бы равно 16, что не является оптимальным ответом.

Во втором примере один из возможных вариантов — ни один сервер не отключится. Тогда на первом сервере 1000000000 сервисов, что и является ответом на задачу. Если отключится сервер 2 или 3, то максимальное число сервисов также будет на первом сервере.

Задача 2. Расследование в Темерии

Ограничение по времени:	2 секунды
Ограничение по памяти:	1024 мегабайта

Темерия — одно из самых могущественных королевств Севера со столицей в городе Вызима. Чародейка Трисс, которая живет в городе Вызима, обнаружила сильные магические аномалии и решила исследовать королевство Темерия, чтобы найти их источник.

В Темерии располагаются n городов, пронумерованных числами от 1 до n, столица Вызима имеет номер 1. Города соединены $n-1$ двусторонними дорогами, i-я дорога соединяет города с номерами u_i и v_i и имеет длину w_i. Гарантируется, что Трисс может добраться из любого города в любой другой, пользуясь только этими дорогами.

Трисс планирует начать и закончить свой путь в Вызиме, побывав во всех n городах. Трисс может ходить по дорогам, но это медленно. У неё есть k кристаллов телепортации, которыми она может воспользоваться для мгновенного перемещения между городами.

В любой момент Трисс может оставить кристалл в городе, где она находится. В дальнейшем чародейка может воспользоваться ранее оставленным кристаллом и мгновенно вернуться по кратчайшему пути в город, в котором она ранее оставила кристалл. После использования кристалл разрушается. Трисс может оставлять и использовать кристаллы в произвольном порядке. К сожалению, телепортация не проходит бесследно. А именно, если Трисс, находясь в городе a, воспользовалась кристаллом и попала в город b, то во всех городах, лежащих на кратчайшем пути от a до b, включая a и b, остается магический след и в дальнейшем через такие города другой маршрут телепортации проходить не может.

Помогите Трисс. Для каждого j от 1 до k включительно, определите, какое минимальное расстояние надо пройти чародейке, чтобы обойти все города королевства и вернуться в Вызиму, потратив при этом не более чем j кристаллов.

Формат входных данных

В первой строке ввода находятся два целых числа n и k ($2 \leqslant n \leqslant 500\,000$; $1 \leqslant k \leqslant n$) — количество городов и количество кристаллов телепортации, которые есть у Трисс.

В следующих $(n-1)$ строках находятся описания дорог: целые числа u_i, v_i и w_i ($1 \leqslant u_i, v_i \leqslant n$; $1 \leqslant w_i \leqslant 10^9$) — номера городов, соединенных i-й дорогой, и длина этой дороги, соответственно.

Формат выходных данных

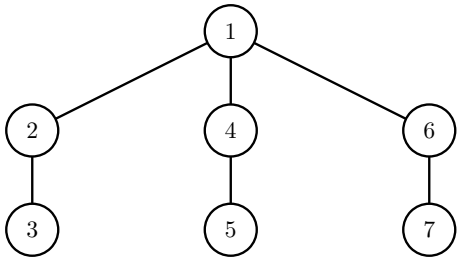
Выведите k чисел, где j-е число — минимальное расстояние, которое требуется пройти чародейке, чтобы побывать во всех городах и вернуться в Вызиму, потратив не более j кристаллов.

Система оценивания

Подзадача	Баллы	Дополнительные ограничения		Необх. подзадачи
		n, k	Дополнительно	
1	9	$n \leqslant 150\,000$; $k = 1$		
2	5	$n \leqslant 100$		У
3	10	$n \leqslant 5\,000$		У, 2
4	9	$n \leqslant 150\,000$; $k \leqslant 300$		У, 1 – 2
5	11	$n \leqslant 150\,000$	Полное двоичное дерево*, $w_i = 1$	
6	11	$n \leqslant 150\,000$	$w_i = 1$	5
7	15	$n \leqslant 150\,000$	Специальный граф**	
8	12	$n \leqslant 150\,000$	Из каждого города выходят не более 10 дорог	
9	11	$n \leqslant 150\,000$		У, 1 – 8
10	4	$n \leqslant 300\,000$		У, 1 – 9
11	3	$n \leqslant 500\,000$		У, 1 – 10

**Полное двоичное дерево* в подзадаче 5 — это дерево, состоящее из $2^s - 1$ вершин ($n = 2^s - 1$), в котором для каждого i от 1 до $2^{s-1} - 1$ существует пара рёбер $(i, 2i)$ и $(i, 2i + 1)$.

***Специальный граф* в подзадаче 7 — это дерево, состоящее из нечётного числа вершин n, в котором для каждого i от 1 до $\frac{n-1}{2}$ существует пара рёбер $(1, 2i)$ и $(2i, 2i + 1)$.



Примеры

стандартный ввод	стандартный вывод
5 1 1 2 1 1 3 1 3 4 1 3 5 1	6
10 2 1 2 10 2 3 6 3 4 8 4 6 5 6 10 7 4 8 6 3 7 6 1 5 4 1 9 9	86 85

Замечание

В первом примере оптимальный маршрут для Трисс выглядит следующим образом.

- Трисс оставляет кристалл в городе 1, затем следует по маршруту $1 \to 2 \to 1 \to 3 \to 4 \to 3 \to 5$, после чего используется кристалл, и она мгновенно возвращается в город 1.

Во втором примере оптимальные маршруты выглядят так:

- Трисс следует по маршруту $1 \to 5 \to 1$, после чего оставляет кристалл в городе 1, затем следует по маршруту $1 \to 9 \to 1 \to 2 \to 3 \to 7 \to 3 \to 4 \to 8 \to 4 \to 6 \to 10$ и использует кристалл в городе 1. Длина такого маршрута равна 86, причём Трисс использует ровно один кристалл.

В другом маршруте Трисс потребуется использовать два кристалла, обозначим их x и y.

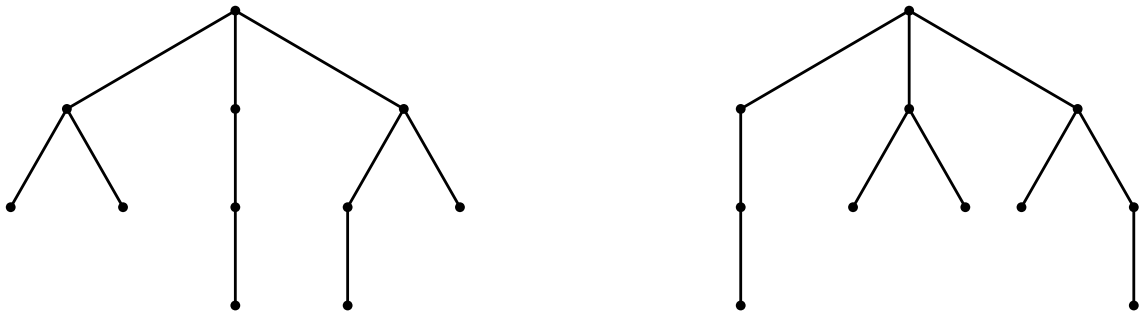
- Трисс оставляет кристалл x в городе 1;
- Затем следует по маршруту $1 \to 5 \to 1 \to 9 \to 1 \to 2 \to 3 \to 7 \to 3 \to 4 \to 6$;
- В городе 6 Трисс оставляет кристалл y;
- Затем переходит $6 \to 10$ и использует кристалл y, возвращаясь в город 6;
- Идет по маршруту $6 \to 4 \to 8$;
- И заканчивает свой путь использованием кристалла x.

Задача 3. Скобки и деревья

Ограничение по времени:	2 секунды
Ограничение по памяти:	1024 мегабайта

Это интерактивная задача с двойным запуском.

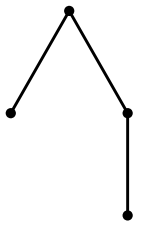
В этой задаче рассматриваются *корневые деревья без порядка на детях*. Корневое дерево без порядка на детях состоит из корня, у которого может быть ноль или более детей. Каждый ребенок в свою очередь является корневым деревом без порядка на детях. При этом, как и следует из названия дерева, порядок, в котором перечисляются дети, не важен, то есть деревья, изображенные на рисунке ниже являются одним и тем же деревом без порядка на детях. Далее будем называть корневые деревья без порядка на детях просто *деревьями*.



Любое дерево можно закодировать в виде *правильной скобочной последовательности* (далее — ПСП) следующим образом:

- Дерево, состоящее из одной вершины, кодируется как «`()`».
- Пусть после удаления корня дерево распадается на поддеревья $t_1, t_2, \ldots, t_k$, где k — количество детей корня исходного дерева. Положим, что $s_1, \ldots, s_k$ — строки, которые кодируют деревья $t_1, \ldots, t_k$. Тогда для любой перестановки $a = [a_1, a_2, ..., a_k]$ чисел от 1 до k исходное дерево может быть закодировано ПСП «`(`$s_{a_1} s_{a_2} \ldots s_{a_k}$`)`».

Обратите внимание, что одно и то же дерево может быть закодировано различными ПСП. Например, дерево, изображенное на рисунке ниже, может быть закодировано с помощью ПСП «`(()(()))`» или «`((())())`».



Вам требуется научиться кодировать произвольную последовательность деревьев $u_1, \ldots, u_n$ в виде одного корневого дерева w. Чтобы проверить, что ваш способ кодирования корректный, ваше решение будет запущено два раза.

Первый запуск

При первом запуске вашей программе на вход подаются n ПСП, каждая из которых представляет собой код s_i некоторого корневого дерева. В ответ ваша программа должна вывести ПСП, которая кодирует произвольное корневое дерево w. В различных подзадачах накладываются различные ограничения на количество вершин в дереве w в зависимости от суммарного количества вершин в исходных деревьях.

Второй запуск

На втором запуске вашей программе подаётся единственная ПСП, которая кодирует дерево w, которое ваша программа вывела при первом запуске. При этом на вход может быть подана любая

подходящая ПСП, которая кодирует дерево w, не обязательно та, которая была выведена вашей программой при первом запуске.

В ответ ваша программа должна вывести ПСП, которые кодируют те же деревья, которые были поданы при первом запуске, в том же порядке. Для каждого дерева вы можете вывести любую кодирующую его ПСП, но порядок самих деревьев в последовательности должен быть таким же, как при первом запуске программы.

Протокол взаимодействия

В начале каждого запуска ваша программа должна прочитать одно число t, равное 1 или 2 — номер запуска.

Первый запуск

При первом запуске необходимо обработать несколько наборов входных данных. Каждый набор подаётся на стандартный поток ввода интерактивно, то есть перед тем, как считать очередной набор, ваша программа должна вывести ответ для всех предыдущих наборов входных данных и сбросить буфер стандартного потока вывода.

В первой строке каждого набора входных данных записано одно число n — количество деревьев, которое нужно закодировать. Если n равно 0, то это означает, что все наборы входных данных обработаны и программа должна завершить работу. Иначе в следующих n строках следуют описания деревьев.

Каждое дерево задается одной строкой s_i, которая состоит из символов «(» и «)» — ПСП, которая кодирует i-е дерево описанным в условии образом. Гарантируется, что s_i задает корректное дерево.

Для данного набора входных данных программа должна вывести ПСП, которая кодирует некоторое дерево w. После вывода дерева необходимо вывести символ конца строки и сбросить буфер потока вывода.

В данной задаче работа программы жюри при первом запуске является адаптивной. Это означает, что программа жюри на первом запуске может использовать выведенные вами в предыдущих наборах входных данных текущего теста деревья w при генерации нового набора входных данных.

Второй запуск

На втором запуске необходимо обработать несколько наборов входных данных. Каждый из наборов входных данных задаётся строкой s. Если строка s равна «0», то вы обработали все наборы входных данных, и программа должна завершить работу. Иначе s содержит некоторую ПСП, кодирующую какое-то дерево w, которое программа построила при первом запуске.

Для каждого такого дерева необходимо вывести в отдельной строке одно число n — количество декодированных деревьев.

В следующей строке требуется вывести n ПСП, которые кодируют в соответствующем порядке те же деревья, что кодировали строки $s_1, \ldots, s_n$, поданные при первом запуске, в одну строку, разделяя их символом «+». Например, если нужно вывести ПСП «(())» и «((()))» в таком порядке, то вывод должен быть таким: в первой строке «2», а во второй строке «(())+((()))».

После вывода числа n и вывода строки с описанием деревьев необходимо перевести строку и сбросить буфер поток вывода.

В каждом из наборов данных во втором запуске вашей программе на вход может подаваться любое из деревьев, полученных при первом запуске.

Замечание

Не забывайте переводить строку после каждого вывода. Обратитесь к памятке участника, чтобы узнать, как правильно сбрасывать поток вывода в интерактивных задачах.

Система оценивания

Обозначим за s суммарную длину ПСП в одном наборе входных данных, а за m — размер вывода вашей программы на первом запуске для этого набора входных данных. Для каждой подзадачи определена функция $f(x)$. Подзадача считается пройденной, если для каждого набора входных данных выполняется $m \leqslant f(s)$, а также если все деревья были корректно восстановлены.

Обозначим за t_i количество вершин в i-м дереве. Тогда длина строки s_i равна $2t_i$.

Также гарантируется, что сумма s по всем наборам входных данных одного теста не превосходит 10^6, а количество наборов входных данных в каждом тесте не превосходит 100.

Подзадача	Баллы	$f(x)$	Дополнительные ограничения		Необх. подз.
			s	Дополнительно	
1	13	$f(x) = x + 2000$	$s \leqslant 200\,000$	При втором запуске даются ПСП, точно совпадающие с выведенными вашим решением при первом запуске.	
2	7	$f(x) = x + 2000$	$s \leqslant 200\,000$	$t_1 < t_2 < \ldots < t_n$	
3	6	$f(x) = x + 2000$	$s \leqslant 200\,000$	$n = 2$	
4	до 34	$f(x) = 4 \cdot x + 2000$	$s \leqslant 200\,000$		
5	до 11	$f(x) = x + 2000$		$t_1 = t_2 = \ldots = t_n > 1$	
6	до 9	$f(x) = x + 2000$		$t_i > 1$	5
7	до 20	$f(x) = x + 2000$			1 – 6

Четвертая подзадача оценивается по следующей формуле. Обозначим $k = max\left(0, \frac{m - 2000}{s}\right)$ для каждого набора входных данных. Введем функцию score(k) следующим образом:

k	score(k)
$\leqslant 1{,}5$	34
2	20
3	10
4	5
> 4	0

Для промежуточных значений k функция вычисляется линейно между соседними строками таблицы и округляется до ближайшего целого числа.

Балл за тест равняется минимуму score(k) по всем наборам входных данных в тесте. Балл за подзадачу равняется минимуму из баллов по тестам этой подзадачи.

Подзадачи 5, 6 и 7 также оцениваются по формуле. Обозначим $c = \max(0, m - s)$ для каждого набора входных данных. Введем функцию score(c) следующим образом:

c	score(c), подз. 5	score(c), подз. 6	score(c), подз. 7
$\leqslant 30$	11	9	20
100	7	7	14
200	4	4	8
2000	2	2	4
> 2000	0	0	0

Для промежуточных значений c функция вычисляется линейно между соседними строками таблицы и округляется до ближайшего целого числа.

Балл за тест равняется минимуму score(c) по всем наборам входных данных в тесте. Балл за подзадачу равняется минимуму из баллов по тестам этой подзадачи.

Пример


стандартный поток ввода
стандартный поток вывода
первый запуск
1
3
()
(())
(()())
(((()(()())))
1
((())())
(((((((((())))))))))
0
второй запуск
2
(((((((((())))))))))
1
(()(()))
((((()())()))
3
()+(())+(()())
0

Задача 4. Спортивная тренировка

Ограничение по времени:	2 секунды
Ограничение по памяти:	1024 мегабайта

Несколько школьников занимаются в спортивной секции. В начале тренировки в зале присутствуют n человек, а затем в течение занятия к ним по одному присоединяются еще q человек. Рост всех $n + q$ школьников различен, пронумеруем школьников от 1 до $n + q$ по возрастанию роста.

На тренировке школьники выполняют упражнения с мячом. Школьники выстраиваются в ряд слева направо в некотором порядке. В зависимости от порядка, в котором они выстроились, некоторые пары школьников образуют *допустимые пары*.

Пара школьников, стоящих на позициях i и j, где $i < j$, образует допустимую пару, если выполнено одно из двух условий:

- школьник на i-й позиции является самым левым школьником из тех, которые ниже школьника на j-й позиции и стоят левее него;
- школьник на j-й позиции является самым правым школьником из тех, которые ниже школьника на i-й позиции и стоят правее него.

Например, если в ряд стоят школьники с номерами $[6, 7, 3, 5, 1, 2]$, то допустимыми являются пары школьников с номерами $(6, 2)$, $(6, 7)$, $(7, 2)$, $(3, 2)$, $(3, 5)$, $(5, 2)$, $(1, 2)$.

У упражнения есть два уровня сложности, на каждом из которых есть свои *допустимые броски*. При выполнении упражнения на любом уровне сложности запрещается бросать мяч школьнику, у которого он уже был во время выполнения этого упражнения.

На первом уровне сложности школьник может бросить мяч любому школьнику, с которым он образует допустимую пару и который ниже его. Например, если в ряд стоят школьники с номерами $[6, 7, 3, 5, 1, 2]$, то школьник с номером 3 может бросить мяч только школьнику с номером 2, школьник с номером 5 — школьникам с номерами 3 и 2, школьник с номером 1 не может бросить мяч никому.

На втором уровне сложности школьник может бросить мяч любому школьнику, с которым он образует допустимую пару. Например, если в ряд стоят школьники с номерами $[6, 7, 3, 5, 1, 2]$, то школьник с номером 3 может бросить мяч школьникам с номерами 2 и 5, школьник с номером 5 — школьникам с номерами 3 и 2, школьник с номером 1 может бросить мяч школьнику с номером 2.

Упражнение выполняется следующим образом. Тренер выбирает уровень сложности упражнения t. Один из школьников берёт мяч и совершает допустимый бросок. Школьник, получивший мяч, снова совершает допустимый бросок, и т.д. Броски выполняются, пока это возможно. Если допустимых бросков несколько, можно выбрать любой из них, но запрещается бросать мяч тому из школьников, у кого уже был мяч во время выполнения этого упражнения. Участники, находящиеся на тренировке, выполняют допустимые для этого уровня сложности броски таким образом, чтобы было произведено максимальное число бросков.

Затем q раз к тренирующимся присоединяется еще один школьник. Он встаёт справа или слева от уже выполнявших упражнение. После этого упражнение выполняется заново на том же уровне сложности.

Для начального состава участников тренировки и после добавления каждого нового школьника необходимо определить, какое максимальное количество бросков смогут сделать участники тренировки.

Формат входных данных

Первая строка содержит одно целое число t ($1 \leqslant t \leqslant 2$) — уровень сложности упражнения.

Вторая строка содержит два целых числа n и q ($1 \leqslant n \leqslant 10^5$, $0 \leqslant q \leqslant 2 \cdot 10^5$) — начальное количество участников упражнения и количество участников, которые к нему присоединятся.

Третья строка содержит n целых чисел $a_1, a_2, \ldots, a_n$ ($1 \leqslant a_i \leqslant n + q$) — номера участников, первоначально стоящих в ряду в порядке слева направо. Гарантируется, что все номера различны.

Следующие q строк содержат номера участников, присоединяющихся к упражнению. Очередная строка содержит символ «`L`» или «`R`» и целое число x через пробел ($1 \leqslant x \leqslant n + q$). Буква «`L`» означает, что школьник номер x встает в ряд слева, а «`R`» — справа.

Гарантируется, что после каждого добавления все номера участников различны.

Формат выходных данных

В первой строке выведите одно число — ответ на задачу для исходных n участников и упражнения сложности t.

В следующих q строках выведите по одному целому числу — ответ на задачу после добавления очередного из q участников и выполнения упражнения той же сложности.

Примеры

стандартный ввод	стандартный вывод
1 6 2 6 7 3 5 1 2 L 8 R 4	3 3 5
2 6 2 6 7 3 5 1 2 L 8 R 4	4 4 6
1 5 4 4 3 1 6 2 R 7 L 8 R 9 L 5	3 3 4 5 4
2 5 4 9 4 6 8 2 R 1 L 7 R 5 R 3	4 4 5 7 6

Пояснение к примеру

В первом примере упражнение оптимально начинать, например, участнику с номером 5. Первым броском можно отдать мяч участнику с номером 3, вторым — участнику с номером 2, третьим — с номером 1. Добавление слева участника с номером 8 не увеличивает максимальное количество бросков. А добавление справа участника с номером 4 позволяет, начиная с участника с номером 7, последовательно бросать мяч участникам с номерами 6, 4, 3, 2 и 1.

Во втором примере тоже можно начать с участника с номером 5 и получить четыре допустимых броска участникам с номерами 3, 2, 7 и 6. Добавление слева участника с номером 8 не меняет максимальное количество бросков, а добавление справа участника с номером 4 позволяет, начиная, например, с номера 7, последовательно бросать мяч участникам с номерами 6, 4, 5, 3, 2 и 1.

Система оценивания

Подзадача	Баллы	Доп. ограничения			Необх. подзадачи
		t	n, q	дополнительно	
1	6	$t = 1$	$n + q \leqslant 16$	–	–
2	4		$n, q \leqslant 100$	–	1
3	3		$n \leqslant 1000$, $q = 0$	–	–
4	5		$n, q \leqslant 1000$	–	1–3
5	3		$q = 0$	–	3
6	10		$n = 1$	$a_1 = 1$ Школьники добавляются в порядке возрастания номеров	–
7	6		–	Гарантируется, что начальный набор участников, их порядок, очерёдность добавления оставшихся и сторона добавления случайны	–
8	5		$n, q \leqslant 50\,000$	–	1–4
9	8		–	–	1–8
10	4	$t = 2$	$n + q \leqslant 16$	–	–
11	6		$n, q \leqslant 100$	–	10
12	5		$n \leqslant 1000$, $q = 0$	–	–
13	9		$n, q \leqslant 1000$	–	10–12
14	3		$q = 0$	–	12
15	6		$n = 1$	$a_1 = 1$ Школьники добавляются в порядке возрастания номеров	–
16	6		–	Гарантируется, что начальный набор участников, их порядок, очерёдность добавления оставшихся и сторона добавления случайны	–
17	7		$n, q \leqslant 50\,000$	–	10–13
18	4		–	–	10–17