Задача A. Программирование квадрокоптеров

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

Школьники готовятся к участию в соревновании по программированию квадрокоптеров. Квадрокоптер, который используется в соревновании, может выполнять две команды: подняться вверх на 1 метр и опуститься вниз на 1 метр. Команда подъёма обозначается символом «(», а команда спуска — символом «)».

Программа для квадрокоптера представляет собой последовательность команд. Программа считается корректной, если, начав её исполнение на уровне земли и выполнив последовательно все команды, квадрокоптер снова оказывается на уровне земли. При этом в процессе выполнения программы квадрокоптер не должен пытаться опуститься ниже уровня земли.

Например, следующие программы являются корректными: «()(())», «((()))». Программа«(((» не является корректной, поскольку квадрокоптер завершает ее выполнение на высоте 3 метра над уровнем земли, программа «())(» также не является корректной, поскольку при выполнении третьей команды квадрокоптер пытается опуститься ниже уровня земли.

Участник соревнования написал корректную программу для квадрокоптера, состоящую из n команд, пронумерованных от 1 до n. Он загрузил её в память квадрокоптера для демонстрации во время соревнования. К сожалению, после загрузки программы в память квадрокоптера участник случайно удалил её на своём компьютере, а квадрокоптер не позволяет выгрузить программу из своей памяти.

К счастью, квадрокоптер поддерживает специальный режим отладки программы. В этом режиме квадрокоптер с загруженной в него программой может отвечать на специальные запросы. Каждый запрос представляет собой два целых числа: l и r, $1 \leqslant l \leqslant r \leqslant n$. В ответ на запрос квадрокоптер сообщает, является ли фрагмент загруженной в него программы, состоящий из команд с l-й по r-ю включительно, корректной программой для квадрокоптера, либо нет. Участник хочет с помощью режима отладки восстановить загруженную в квадрокоптер программу.

Требуется написать программу-решение, которая взаимодействует с программой жюри, моделирующей режим отладки квадрокоптера, и в итоге восстанавливает загруженную в квадрокоптер программу.

Формат входных данных

Это интерактивная задача.

Сначала на вход подаётся целое число n — количество команд в программе квадрокоптера ($2 \leqslant n \leqslant 50000$).

Для каждого теста жюри зафиксировано число k — максимальное количество запросов. Гарантируется, что k запросов достаточно, чтобы решить задачу. Это число не сообщается программе-решению. Ограничения k в различных подзадачах приведены в таблице системы оценивания. Если программа-решение делает более k запросов к программе жюри, то на этом тесте она получает в качестве результата тестирования «Неверный ответ».

Чтобы сделать запрос, программа-решение должна вывести строку вида «? l r», где l и r — целые положительные числа, задающие фрагмент программы квадрокоптера ($1 \leqslant l \leqslant r \leqslant n$).

В ответ на запрос программы-решения программа жюри подаёт ей на вход либо строку «Yes», либо строку «No», в зависимости от того, является ли запрошенный фрагмент программы квадрокоптера корректной программой.

Если программа-решение определила ответ на задачу, то она должна вывести строку «! c_1 c_2 ... c_n», где символ c_i задаёт i-ю команду в программе квадрокоптера и равен либо «(», либо «)».

После этого программа-решение должна завершиться.

Гарантируется, что в каждом тесте программа в памяти квадрокоптера является фиксированной корректной программой, которая не меняется в зависимости от запросов, произведённых программой-решением.

В тестах $k = 100000$, то есть, разрешается произвести не более 100000 запросов.

Формат выходных данных

Примеры

стандартный ввод	стандартный вывод
4 <ожидание ответа> Yes <ожидание ответа> No <ожидание ответа> Yes <ожидание ответа> Yes <ожидание ответа>	<ожидание ввода> ? 1 4 <ожидание ввода> ? 1 3 <ожидание ввода> ? 1 2 <ожидание ввода> ? 3 4 <ожидание ввода> ! ()()
6 <ожидание ответа> Yes <ожидание ответа> No <ожидание ответа> Yes <ожидание ответа>	<ожидание ввода> ? 3 4 <ожидание ввода> ? 1 2 <ожидание ввода> ? 2 5 <ожидание ввода> ! ((()))

Задача B. Новогодний и прямоугольный

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	3 секунды
Ограничение по памяти:	256 мегабайт

Это **интерактивная** задача.

На Новый год Дед Мороз подарил Глебу то, о чём он уже давно мечтал — клетчатый квадрат размером $n \times n$. Подарок этот не простой, а с сюрпризом — внутри квадрата Дед Мороз выбрал некоторый непустой прямоугольник, и в каждую клетку этого прямоугольника он положил по мандарину.

Теперь, чтобы получить желанный подарок, Глебу нужно сыграть с Дедом Морозом в очень интересную игру. Глеб должен отгадать, в каком именно прямоугольнике находятся все мандаринки, подаренные Дедом Морозом. Будем считать, что строки и столбцы занумерованы числами от 1 до n снизу вверх и слева направо. Глеб может производить два типа запросов:

- ? $x_1\, y_1\, x_2\, y_2$ ($1 \leqslant x_1 \leqslant x_2 \leqslant n$, $1 \leqslant y_1 \leqslant y_2 \leqslant n$) — в ответ на этот запрос Дед Мороз говорит, сколько мандаринок находится в прямоугольнике, левым нижним углом которого является клетка (x_1, y_1), а правым верхним — клетка (x_2, y_2);
- ! $x_1\, y_1\, x_2\, y_2$ ($1 \leqslant x_1 \leqslant x_2 \leqslant n$, $1 \leqslant y_1 \leqslant y_2 \leqslant n$) — когда Глеб уверен, что он точно знает, где находятся мандаринки, он должен сделать запрос такого вида, чтобы сообщить свой ответ. При этом (x_1, y_1) соответствует предполагаемому расположению левого нижнего угла, а (x_2, y_2) — правого верхнего.

Формат входных данных

При запуске решения на вход вашей программе подается одно число n ($1 \leqslant n \leqslant 2 \cdot 10^9$) — размер квадрата.

Затем на каждый запрос типа "?" вам будет выдаваться количество мандаринок, находящихся в указанном вами прямоугольнике.

Формат выходных данных

Вы должны выводить корректные запросы в формате, описанном выше. Последним должен следовать единственный запрос вида "`!`", после чего ваша программа должна немедленно завершиться. Ваша программа должна произвести не больше q (параметр зависит от номера группы) запросов типа "?". Обратите внимание, что последний запрос, выводящий ответ, не входит в данные q запросов.

В точности соблюдайте формат выходных данных. После вывода каждой строки сбрасывайте буфер вывода — для этого используйте команды `flush(output)` на языке Паскаль или Delphi, `fflush(stdout)` или `cout.flush()` в C/C++, `sys.stdout.flush()` на языке Python, `System.out.flush()` на языке Java.

Пример

стандартный ввод	стандартный вывод
4 \medskip 6 \medskip 6 \medskip 4	\medskip ? 1 1 4 4 \medskip ? 1 3 4 4 \medskip ? 2 3 4 4 \medskip ! 1 3 3 4

Замечание

Пример в условии иллюстрирует взаимодействие с проверяющей программой. Для прохождения первого теста не обязательно производить такие же запросы, как в примере.

Тесты к этой задаче состоят из шести групп. Баллы за каждую группу ставятся только при прохождении всех тестов группы и всех тестов **предыдущих** групп.

Группа	Тесты	Баллы	Дополнительные ограничения		Комментарий
			n	q	
0	1 – 1	0	$n = 4$	$q = 1000$	Тест из условия.
1	2 – 12	10	$n \leqslant 10$	$q = 10\,000$	
2	13 – 23	20	$n \leqslant 100$	$q = 10\,000$	
3	24 – 34	20	$n \leqslant 10\,000$	$q = 20\,000$	
4	35 – 45	25	$n \leqslant 2 \cdot 10^9$	$q = 128$	
5	46 – ∞	25	$n \leqslant 2 \cdot 10^9$	$q = 64$	

Задача C. 1.5G — модем

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

Алиса должна передать Бобу целое неотрицательное число X, меньшее 10^{10}. Единственным способом связи у Алисы является старый 1.5G модем, который при передаче произвольным образом переставляет цифры в числе.

Алиса может отправить по модему число из не более, чем 2021 цифр (ведущие нули допустимы).

Боб получает от Алисы отправленное число с переставленными произвольным образом цифрами. Задача Боба — восстановить по полученному сообщению число X.

Формат входных данных

В первой строке входных данных задаётся одна строка — «`Alice`», если данные предназначены для Алисы, и «`Bob`», если данные предназначены для Боба.

В случае, когда данные предназначены для Алисы, вторая строка содержит целое неотрицательное число X, меньшее 10^{10}, заданное без ведущих нулей.

В случае, когда данные предназначены для Боба, вторая строка содержит строку s из цифр — отправленное Алисой число с переставленными цифрами. Строка не может быть пустой или состоять из более чем 2021 цифры.

Формат выходных данных

Если входные данные предназначены для Алисы, выведите непустую строку, состоящую из не более, чем 2021 цифр.

Если входные данные предназначены для Боба, выведите одно число — предполагаемое значение X.

Протокол взаимодействия

Ваше решение будет запущено два раза — в первый раз на данных, предназначенных для Алисы, во второй раз — на сгенерированных первым запуском данных, предназначенных для Боба. Передача каких-либо ещё данных между запусками запрещена.

Примеры

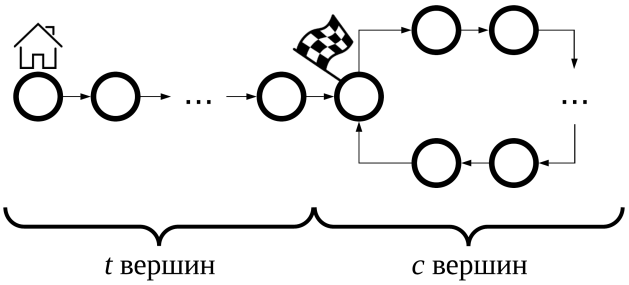
стандартный ввод	стандартный вывод
`Alice` `2021`	`66889`
`Bob` `68986`	`2021`

Задача D. Двое в лесу

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	1 секунда
Ограничение по памяти:	512 мегабайт

Это интерактивная задача.

Игровое поле для одной малоизвестной кабо-вердианской игры представляет из себя **ориентированный** граф, состоящий из двух частей. Одна из этих частей — цикл из c вершин. Вторая часть — цепь из t вершин, причём из последней вершины этой цепи проведено ребро в одну из вершин цикла — назовём её интересной. Игровое поле держат от вас в секрете, так что никто кроме ведущего не знает ни t, ни c.



В начале игры игровые фишки всех десяти игроков, для удобства пронумерованных цифрами от 0 до 9, располагаются в начале цепи, в вершине у домика (см. рис.). Затем не более q раз ведущий будет одновременно передвигать в следующие вершины фишки игроков, высказавших такое желание на этом ходу, а после этого объявлять, фишки каких игроков находятся в одной вершине, а каких — в разных. Обратите внимание, что у каждой вершины игрового поля есть только одна следующая, при этом каждая кроме стартовой и интересной является следующей только для одной вершины. Стартовая вершина не является следующей ни для какой вершины, а интересная вершина является следующей для последней вершины цепочки и для одной из вершин цикла. Ведущему так понравилась идея не сообщать игрокам t и c, что и q он решил тоже не объявлять.

Цель игры — перевести игровые фишки всех игроков в интересную вершину, то есть в ту, которая отмечена финишным флагом (см. рис.). Игроки не представляют, как можно выиграть в такой игре не зная ни c, ни t, ни q, но, к счастью для них, они ещё и ваши друзья. Помогите им: скоординируйте их действия так, чтобы победить.

Протокол взаимодействия

Для удобства пронумеруем игроков целыми числами от 0 до 9 включительно.

Для того, чтобы отдать команды перемещения игрокам, выведите «`next`» а затем через пробел номера игроков, которым необходимо отдать команды. Например, чтобы отдать команду перемещения игрокам с номерами 0, 2, 5 и 9 выведите «`next 0 2 5 9`». На каждом ходу обязательно перемещать хотя бы одного из игроков.

В качестве ответа проверяющая программа выдаст сначала число k, а потом 10 цифр, разбитых на k групп, разделённых пробелами. Игроки, соответствующие цифрам, находящимся в одной группе, находятся в одной вершине. Игроки, соответствующие цифрам, находящимся в разных группах, находятся в разных вершинах. Цифры в каждой группе следуют в порядке возрастания.

Например ответ проверяющей программы «`2 05 12346789`» означает, что игроки с номерами 0 и 5 находятся в одной вершине, а все остальные игроки в одной и той же, но другой вершине. Ответ проверяющей программы «`4 01 567 234 89`» означает, что игроки находятся в четырёх различных вершинах: в первой находятся игроки с номерами 0 и 1, во второй — игроки с номерами 5, 6 и 7, в третьей — игроки с номерами 2, 3 и 4, а в четвёртой — игроки с номерами 8 и 9.

Если вы превысите лимит на количество ходов или нарушите протокол взаимодействия, то вместо информации о местоположении игроков вашей программе на вход будет передана строка «`stop`».

Считав такую строку, ваша программа должна немедленно завершить работу, иначе она может получить вердикт тестирования, отличный от настоящего.

После того, как все игроки соберутся в интересной вершине необходимо вывести «`done`» и завершить работу программы.

После вывода каждой строки сбрасывайте буфер вывода — для этого используйте `flush(output)` на языке Паскаль или Delphi, `fflush(stdout)` или `cout.flush()` в C/C++, `sys.stdout.flush()` на языке Python, `System.out.flush()` на языке Java.

Тесты	Дополнительные ограничения	
	$(t+c)$	q
1	$(t+c) \leqslant 65$	$6600 \leqslant q \leqslant 10\,000$
2–10	$(t+c) \leqslant 65$	$6600 \leqslant q \leqslant 10\,000$
11–20	$(t+c) \leqslant 100$	$(t+c)^2 \leqslant q \leqslant 10\,000$
21–61	$(t+c) \leqslant 1000$	$4 \cdot (t+c) \leqslant q \leqslant 10\,000$
62–100	$(t+c) \leqslant 1000$	$3 \cdot (t+c) \leqslant q \leqslant 10\,000$

Пример

стандартный ввод	стандартный вывод
2 3 10000	next 0 1 2 23456789 01 next 0 3 1 23456789 0 next 0 1 3 23456789 0 1 next 0 3 0 23456789 1 next 0 1 3 23456789 1 0 next 0 2 23456789 01 next 0 1 2 3 4 5 6 7 8 9 2 01 23456789 next 0 1 2 3 4 5 6 7 8 9 1 0123456789 done

Замечание

В условии в примере взаимодействия вводимые и выводимые данные расположены для удобства восприятия в хронологическом порядке, при реальном взаимодействии никакие «лишние» переводы строк возникать не должны.

Пример

стандартный ввод	стандартный вывод
2 3 10000	next 0 1 2 23456789 01 next 0 3 1 23456789 0 next 0 1 3 23456789 0 1 next 0 3 0 23456789 1 next 0 1 3 23456789 1 0 next 0 2 23456789 01 next 0 1 2 3 4 5 6 7 8 9 2 01 23456789 next 0 1 2 3 4 5 6 7 8 9 1 0123456789 done

Задача E. Четные и нечетные сочетания

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	2 секунды
Ограничение по памяти:	512 мегабайт

Это задача с двойным запуском.

Сочетанием из n элементов по k будем называть k-элементное подмножество n-элементного множества $\{1, 2, \ldots, n\}$. Чтобы записать сочетание, перечислим его элементы в порядке возрастания. Например, сочетания из 3 элементов по 2 выглядят так: $\{1, 2\}$, $\{1, 3\}$, $\{2, 3\}$.

Будем называть сочетание *четным*, если количество элементов в нём — четное число, и *нечетным* в противном случае. Зафиксируем $n > 0$ и рассмотрим два множества: A_n, множество всех чётных сочетаний из n элементов, и B_n, множество всех нечетных сочетаний из n элементов. Можно доказать, что A_n и B_n содержат одинаковое количество сочетаний.

Для каждого $n = 1, 2, \ldots, 50$ задача такова. Постройте любую биекцию (взаимно однозначное соответствие) между множествами A_n и B_n. После этого по данному элементу одного из этих множеств выводите соответствующий ему элемент другого множества.

Формат входных данных

При первом запуске в первой строке записано целое число t — количество тестовых случаев ($1 \leqslant t \leqslant 1000$). Далее следуют их описания.

Каждый тестовый случай задает сочетание и занимает две строки.

В первой из них записаны через пробел два целых числа n и k ($1 \leqslant n \leqslant 50$, $0 \leqslant k \leqslant n$).

Во второй записаны через пробел k целых чисел $a_1, a_2, \ldots, a_k$ — элементы сочетания ($1 \leqslant a_1 < a_2 < \ldots < a_k \leqslant n$). Если $k = 0$, то вторая строка пуста.

При втором запуске формат ввода точно такой же, как при первом запуске. Но в каждом тестовом случае дано не исходное сочетание, а то, которое было выведено при первом запуске.

Формат выходных данных

При первом запуске в ответ на каждый тестовый случай выведите сочетание из другого множества, соответствующее заданному. Формат сочетания в выводе — точно такой же, как во вводе. У выведенного сочетания n должно совпадать с заданным, а k должно иметь другую четность. Других ограничений на выбор соответствия нет.

При втором запуске в ответ на каждый тестовый случай, как и при первом запуске, выведите сочетание из другого множества, соответствующее заданному. Формат сочетания в выводе — точно такой же, как во вводе. Поскольку соответствие должно быть взаимно однозначным, выведенное при втором запуске сочетание должно совпадать с тем, которое было дано при первом запуске. Это и будет проверять программа жюри.

Протокол взаимодействия

Для проверки того, что вы действительно построили биекцию, в этой задаче ваше решение будет запущено на каждом тесте два раза. В конце каждой строки входных данных следует символ перевода строки.

Пример

стандартный ввод	стандартный вывод
6 3 0 2 1 1 3 3 1 2 3 3 1 1 3 1 2 3 1 3	6 3 0 2 1 1 3 3 1 2 3 3 1 1 3 1 2 3 1 3

Задача F. Сообщение из шума

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	2 секунды
Ограничение по памяти:	512 мегабайт

Это задача с двойным запуском.

Алиса хочет передать Еве по *числовому проводу* сообщение — одно английское слово.

К сожалению, сейчас числовой провод передаёт только шум: случайные целые числа от 0 до $10^9 - 1$ включительно. Алисе известна последовательность из следующих 10 000 чисел, которые будут переданы.

К счастью, у Алисы есть суперспособность: стереть из этой последовательности любое количество элементов на любых позициях. Порядок оставшихся чисел не поменяется.

К сожалению, после этого примерно половина чисел потеряется при передаче: каждое передаваемое число с вероятностью $\frac{1}{2}$ исчезнет. Порядок оставшихся чисел опять не поменяется.

Как должны действовать Алиса и Ева, чтобы передать заданное слово?

Формат входных данных

В этой задаче ваше решение будет запущено на каждом тесте два раза. При вводе и выводе числа в одной строке отделяются друг от друга пробелами. В конце каждой строки входных данных следует символ перевода строки.

Формат выходных данных

При первом запуске решение действует за Алису. В первой строке записано имя «`Alisa`». Вторая строка содержит одно слово из английского словаря, имеющее длину от 2 до 15 букв и состоящее из маленьких букв. В третьей строке записано целое число n — размер последовательности (в этой задаче n всегда равно 10 000). Четвертая строка содержит n чисел от 0 до $10^9 - 1$ включительно — исходную последовательность. Числа выбраны заранее генератором псевдослучайных чисел, все целые числа из диапазона равновероятны.

При втором запуске решение действует за Еву. В первой строке записано имя «`Eva`». Во второй строке записано целое число k — количество чисел, оставшихся в последовательности. Третья строка содержит k чисел от 0 до $10^9 - 1$ включительно — то, что осталось от последовательности: каждое число, которое Алиса решила оставить в первом запуске, с вероятностью $\frac{1}{2}$ оказалось здесь и с вероятностью $\frac{1}{2}$ исчезло. Как именно исчезают числа из последовательности — зафиксировано заранее в каждом тесте, то есть, если решения ведут себя одинаково при первом запуске, то они получат одинаковые последовательности при втором запуске.

Протокол взаимодействия

При первом запуске вывести следует те числа, которые Алиса решила **оставить** в последовательности. В первой строке выведите целое число m — количество оставшихся чисел. Во второй строке выведите сами эти числа в том же порядке, в котором они следуют в исходной последовательности.

При втором запуске выведите одно английское слово — то, которое Алиса должна была передать Еве.

Примеры

стандартный ввод	стандартный вывод
Alisa spark 10000 833080662 16249270 933346436 811379468 <...> 13286897 459644281	3900 933346436 811379468 877083772 408973036 <...> 583178591 13286897
Eva 1955 811379468 408973036 585189166 111199534 <...> 226510051 829146141	spark

Задача G. Абстракционист и треугольники

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	15 секунд
Ограничение по памяти:	1024 мегабайта

Это задача с двойным запуском

Требуется закодировать треугольник ненулевой площади с целыми сторонами и периметром p, никакие две стороны которого не равны, треугольником ненулевой площади и периметром $p - 6$, и затем по закодированному треугольнику восстановить исходный.

Формат входных данных

Первая строка входных файлов содержит режим работы: `encode`, если требуется закодировать треугольники или `decode`, если требуется восстановить исходные. Вторая строка содержит одно целре число t ($1 \leqslant t \leqslant 25\,000$) — количество тестовых примеров.

Далее следуют тестовые примеры по одному на строку.

В режиме кодирования каждый тестовый пример содержит три целых числа a, b и c — стороны исходного треугольника. Гарантируется, что $3 \leqslant a + b + c \leqslant 1000$, что a, b и c попарно не равны и что треугольник со сторонами a, b и c существует и имеет ненулевую площадь.

В режиме декодирования каждый тестовый пример содержит три целых числа a', b' и c' — переставленные в каком-то порядке стороны, выведенные вашей программой в режиме кодирования.

Формат выходных данных

В режиме кодирования выведите для каждого тестового примера в произвольном порядке три целых числа a', b', c', такие, что треугольник со сторонами a', b', c' существует и имеет ненулевую площадь, и $a' + b' + c' = a + b + c - 6$.

В режиме декодировыния для каждого тестового примера выведите три стороны исходного треугольника. Порядок вывода сторон внутри одного тестового примера произвольный.

Примеры

стандартный ввод	стандартный вывод
encode 2 20 24 11 25 36 49	11 20 24 25 36 49
encode 1 1000 999 998	998 999 1000