Задача 1. 23 февраля

Имя входного файла:	`23feb.in`
Имя выходного файла:	`23feb.out`
Ограничение по времени:	0.5 секунд
Ограничение по памяти:	256 мегабайт

Сегодня 23 Февраля, поэтому Малыш решил устроить парад своих солдатиков. Он уже достал их из коробки, расположил в одну линию и теперь хочет построить их по росту в порядке невозрастания слева направо.

В честь праздника Малыш решил, что будет менять местами тех и только тех солдатиков, между которыми стоят ровно 2 или ровно 3 других солдатика.

Малышу совершенно не обязательно построить солдатиков за минимально возможное количество перестановок, но ему обязательно нужно это сделать не более чем за 23 000 перестановок, иначе он не успеет до сна.

Помогите Малышу справиться с этой ответственной задачей.

Формат входных данных

В первой строке входного файла указано целое число N ($2 \leqslant N \leqslant 100$) — количество солдатиков у Малыша. Во второй строке через пробел указаны N положительных целых чисел, каждое из которых не превосходит 2000. K-ое число во второй строке задает рост K-ого солдатика в исходном построении. Солдатики нумеруются числами от 1 до N слева направо.

Формат выходных данных

Если расположить солдатиков требуемым образом невозможно, то в выходной файл требуется вывести единственное слово «`NO`» без кавычек.

Если построение осуществимо, то в первой строке выведите единственное слово «`YES`» без кавычек, а во второй строке выведите требуемое количество действий K ($0 \leqslant K \leqslant 23\,000$). Далее нужно вывести K строк, каждая из которых должна содержать два числа X и Y, означающих, что Малышу нужно поменять местами солдатиков, стоящих сейчас на X-ом и Y-ом местах.

Если есть несколько решений, выведите любое. Обратите внимание, что минимизировать число действий не требуется, главное — чтобы их было не больше 23 000.

Примеры

23feb.in	23feb.out
10 10 9 4 7 6 5 1 3 2 8	YES 2 10 7 3 7
2 1500 1700	NO

Примечание

Решения, работающие при $N \leqslant 8$, будут набирать 50 баллов.

Задача 2. Оптимальное расписание

Имя входного файла:	`buses.in`
Имя выходного файла:	`buses.out`
Ограничение по времени:	0.5 секунд
Ограничение по памяти:	256 мегабайт

Как вы помните, ваш друг работает главным диспетчером в одной из компаний, владеющей сетью маршрутных такси. Недавно директор поставил перед ним задачу оптимизировать расписание маршруток на некотором маршруте.

Экономический отдел предоставил вашему другу прогноз пассажиропотока на следующие n часов. Теперь необходимо составить расписание маршруток, а именно для каждого часа определить, выйдет маршрутка на линию в этот час или нет. Известно, что если в i-ый час на маршрут выйдет маршрутка, то она принесет прибыль a_i рублей, при этом a_i может быть как положительным, так и нулевым или отрицательным. Также, в связи с ограниченным размером автопарка и в соответствии с требованиями департамента транспорта, маршрутки должны работать примерно $2/3$ всего времени. Более формально, для любого i ($1 \leqslant i \leqslant n$) величина

$$b = \frac{t_{work}}{2} - t_{skip}$$

должна лежать в пределах от $-k$ до k включительно. Здесь t_{work} — количество часов до i-ого включительно, в которые маршрутки выходили на линию, а t_{skip} — количество часов до i-ого включительно, в которые маршрутки не выходили на линию.

Помогите вашему другу найти максимальную суммарную прибыль маршрута с учетом всех требований.

Формат входных данных

В первой строке входных данных находится количество часов n, на которые необходимо составить расписание, и число k ($1 \leqslant n \leqslant 10^5$, $1 \leqslant k \leqslant 10$). На второй строке находятся n чисел: прогнозируемая прибыль для каждого часа a_i ($|a_i| \leqslant 10^9$).

Все числа во входном файле целые.

Формат выходных данных

Выведите одно число — максимальную суммарную прибыль в рублях.

Примеры

buses.in	buses.out
5 1 2 1 3 4 -5	9
5 2 2 1 3 4 -5	10
5 1 5 5 -10 5 5	20

Примечание на следующей странице

Примечание

В первом примере оптимально выпустить маршрутки на линию в первый, третий и четвертый часы, таким образом, прибыль составит $2+3+4=9$ рублей. Величины t_{work}, t_{skip} и b после каждого часа представлены в таблице:

№ часа	1	2	3	4	5
t_{work}	1	1	2	3	3
t_{skip}	0	1	1	1	2
b	0.5	-0.5	0	0.5	-0.5

Решение, в котором маршрутки выходят на линию в первый, второй, третий и четвертый часы, неправильно, потому что в таком случае уже после третьего часа $t_{work} = 3$, $t_{skip} = 0$, а значит, $b = 1.5$, что недопустимо по условию задачи.

Однако, для второго примера это решение допустимо, так как максимальное значение b равно 2, что удовлетворяет ограничению. Это решение и является оптимальным.

В третьем примере необходимо выпустить маршрутки на линию во все часы, кроме третьего.

Решения, правильно работающие при $n \leqslant 20$, оцениваются не менее чем в 20 баллов.

Решения, правильно работающие при $n \leqslant 1000$, оцениваются не менее чем в 50 баллов.

Задача 3. Часы с кукушкой

Имя входного файла:	`cuckoo.in`
Имя выходного файла:	`cuckoo.out`
Ограничение по времени:	0.5 секунд
Ограничение по памяти:	64 мегабайта

У Васи на кухне висят часы с кукушкой. Часы устроены так: в каждый ровный час кукушка кукует столько раз, сколько сейчас часов (от 1 до 12), например, ровно в 7:00 кукушка кукует 7 раз. Кроме того, в 30 минут каждого часа (в 0:30, 1:30, 2:30 и т.д.) кукушка кукует ровно один раз.

Васе очень нравится смотреть на то, как кукует кукушка, он любит считать, сколько раз она прокуковала. Но, к сожалению, сегодня мама отправила Васю в магазин за покупками, и поэтому он пропустил несколько моментов, в которые куковала кукушка. Определите, сколько раз всего куковала кукушка за время отсутствия Васи.

Считайте, что кукушка кукует очень быстро. Например, даже в 11:00 она успевает прокуковать 11 раз быстрее, чем за одну минуту, т.е. к моменту 11:01 кукушка уже закончила куковать.

Формат входных данных

Входные данные содержат четыре целых числа H_1, M_1, H_2 и M_2 — время ухода (H_1 часов M_1 минут) и время возвращения (H_2 часов M_2 минут) Васи. Гарантируется, что $0 \leqslant H_{1,2} < 12$ и что $0 < M_{1,2} < 60$. Гарантируется, что момент ухода Васи следует до момента его возвращения (т.е. или $H_1 < H_2$, или $H_1 = H_2$, но $M_1 < M_2$), и что кукушка не кукует ни в момент ухода, ни в момент возвращения (т.е. что $M_{1,2} \neq 0$ и $M_{1,2} \neq 30$).

Вася уходил и приходил в одной и той же половине суток, т.е. между его моментом ухода и моментом прихода не было ни полудня, ни полуночи.

Формат выходных данных

Выведите одно число — сколько раз кукушка куковала за время отсутствия Васи.

Примеры

cuckoo.in	cuckoo.out
2 20 2 40	1
2 31 4 1	8
8 10 8 20	0

Примечание

В первом примере кукушка кукует один раз — в момент времени 2:30.

Во втором примере кукушка кукует три раза в момент времени 3:00, один раз в момент времени 3:30, и еще четыре раза в момент времени 4:00 — итого 8 раз.

В третьем примере кукушка не куковала ни разу.

Задача 4. Угадайка

Имя входного файла:	`module.in`
Имя выходного файла:	`module.out`
Ограничение по времени:	0.5 секунд
Ограничение по памяти:	256 мегабайт

Вася прошел на уроке математики в школе операцию деления с остатком. Чтобы закрепить пройденный материал, он решил потренироваться в выполнении этого действия. Придя домой, Вася попросил маму назвать N произвольных натуральных чисел X_i ($1 \leqslant i \leqslant N$), не превосходящих 10^9. Затем он придумал некоторое натуральное число M, не превосходящее наибольшего из X_i, и вычислил остатки Y_i от деления всех чисел на него.

Вечером к Васе зашел его друг Леша и увидел результаты вычислений. Подумав несколько минут, он радостно сообщил товарищу, что придумал еще одно значение M, удовлетворяющее всем соотношениям. Вася был шокирован. Он не мог поверить, что это возможно, и обратился к Вам с просьбой о помощи.

Помогите Васе найти все возможные значения M, удовлетворяющие всем соотношениям.

Несмотря на то, что и Вася, и его друг Леша — прилежные и умные ученики, они могли ошибиться в вычислениях, и ни одного подходящего M может не существовать.

Формат входных данных

В первой строке находится одно натуральное число N — количество чисел, с которыми Вася проводил операцию деления ($1 \leqslant N \leqslant 1000$).

Далее следуют N строк, в i-ой из которых находятся два целых числа X_i, Y_i, разделенные пробелом ($1 \leqslant X_i \leqslant 10^9$, $0 \leqslant Y_i \leqslant X_i$).

Формат выходных данных

В первой строке должно находиться целое число K — количество различных M, удовлетворяющих всем соотношениям.

Во второй строке должны находиться K чисел, разделенные пробелами — все возможные значения M. Числа можно выводить в произвольном порядке.

Примеры

module.in	module.out
2 9 4 6 1	1 5
3 14 2 7 1 12 0	2 6 3

Примечание

В первом примере существует единственное число $M = 5$. Остаток от деления 9 на 5 есть 4, а остаток от деления 6 на 5 есть 1.

Во втором примере в качестве M можно взять 3 и 6.

Решения, работающие при $X_i \leqslant 200\,000$, будут набирать 30 баллов.

Задача 5. Миша против древних берляндцев

Имя входного файла:	`string.in`
Имя выходного файла:	`string.out`
Ограничение по времени:	0.5 секунд
Ограничение по памяти:	256 мегабайт

Миша изучает работы по расшифровке древнеберляндских текстов. Древние берляндцы пользовались иероглифичной письменностью. Поскольку число иероглифов очень велико, в трудах, изучаемых Мишей, каждый иероглиф записывается в виде последовательности строчных латинских символов. Древнеберляднское слово представляет собой последовательность из одного или нескольких иероглифов, записанных один за другим слева направо без разделителей. Известно, что древние берляндцы очень любили палиндромы, поэтому все древнеберляндские слова являются палиндромами: первый иероглиф слова совпадает с последним, второй иероглиф — с предпоследним и так далее.

К сожалению, Миша не нашел правила сопоставления иероглифов последовательностям латинских символов, поэтому чтение трудов дается ему очень тяжело. Отчаявшись найти эти правила, Миша решил попробовать сам угадать разбиение слов на иероглифы, для чего он придумал следующий алгоритм: он выписывает каждое отдельное слово и пытается разбить его на наибольшую по длине последовательность иероглифов так, чтобы эта последовательность при объединении давала исходное слово и при этом являлась палиндромом. Но поиск такого разбиения оказался слишком сложным для Миши, поэтому он обратился за помощью к Вам.

Формат входных данных

В первой и единственной строке записано слово, состоящее из не более чем $4 \cdot 10^5$ строчных латинских букв — древнеберляндское слово, не разбитое на иероглифы.

Формат выходных данных

В первой строке вывести число N — количество иероглифов, в последующих N строках вывести последовательности латинских символов, соответствующие иероглифам исходного слова. При этом объединение всех иероглифов должно давать исходное слово, а последовательность иероглифов должна быть палиндромом. Количество N должно быть максимально возможным для всех корректных разбиений.

Примеры

string.in	string.out
abacaba	7 a b a c a b a
nnoi	1 nnoi
abcab	3 ab c ab

Примечание

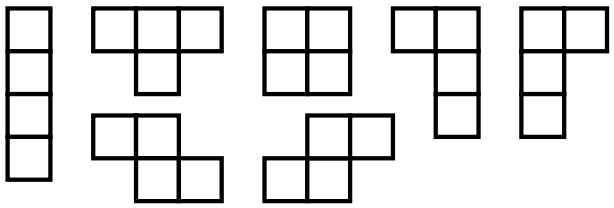
Гарантируется, что решение, работающее для слов, состоящих из не более чем 150 000 латинских букв, набирает 60 баллов.

Задача 6. Площадь Пажитнова

Имя входного файла:	`tetris.in`
Имя выходного файла:	`tetris.out`
Ограничение по времени:	0.5 секунд
Ограничение по памяти:	256 мегабайт

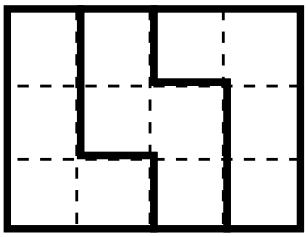
В прошлом году всемирно известной игре Тетрис исполнилось 30 лет. В Берляндии очень любят Тетрис, и поэтому решили назвать одну из городских площадей в столице в честь изобретателя игры, Алексея Леонидовича Пажитнова. Дизайн площади, разумеется, должен напоминать всем о Тетрисе. Поэтому она будет замощена плитками в виде фигурок игры.

Как известно, фигурки Тетриса состоят из четырех «клеток». Берляндский завод тротуарной плитки может выпускать плитку в виде любых фигурок Тетриса. Размер каждой «клетки» такой плитки составляет 1×1 метр. Все возможные фигурки Тетриса представлены на рисунке ниже.



Площадь Пажитнова имеет вид прямоугольника длиной N и шириной M метров. Для упрощения замощения она вся расчерчена на «клетки» размером 1×1 метр, то есть представляет собой сетку. При замощении «клетки» плитки должны совпадать с «клетками» площади. Плитки можно использовать только целиком, они не должны перекрываться или выступать за пределы площади, но их можно поворачивать на углы, кратные 90°. Разумеется, плитки должны покрывать площадь полностью. Размерами промежутков между плитками, покрывающими соседние «клетки», можно пренебречь.

Например, площадь длиной 3 и шириной 4 метра можно покрыть тремя плитками, как показано на рисунке ниже.



Напишите программу, которая поможет берляндцам найти подходящее покрытие.

Продолжение условия на следующей странице

Формат входных данных

В первой и единственной строке записаны два натуральных числа N и M ($1 \leqslant N, M \leqslant 100$), разделенные пробелом — длина и ширина площади соответственно.

Формат выходных данных

В первой строке выведите число K — минимальное количество плиток, нужное для того, чтобы замостить площадь. Если площадь замостить невозможно, то выведите одно число `-1`.

Если замощение существует, то выведите его, начиная со второй строки. А именно, считайте, что плитки, используемые в замощении, занумерованы натуральными числами от 1 до K. Выведите N строк по M чисел, разделенных пробелами, в каждой. Эти числа должны соответствовать номерам плиток, покрывающим соответствующие «клетки» площади.

Если существует несколько замощений с минимальным количеством плиток, то выведите любое из них.

Примеры

tetris.in	tetris.out
3 4	3 1 2 3 3 1 2 2 3 1 1 2 3
3 3	-1

Примечание

Первый пример соответствует картинке из условия.