"A" Простая задача

Ограничение по времени	2 секунды
Ограничение по памяти	256 мегабайт

Жюри Russian Code Cup приготовило большое количество различных интересных задач. Но все они не нравятся председателю жюри. Он говорит, что все они слишком сложные. А для того, чтобы больше участников решило хотя бы одну задачу, нужна простая задача.

Председатель жюри называет задачу *простой*, если у нее существует решение, использующее не более одного оператора ветвления и не более двух операторов цикла, либо решение, использующее не более двух операторов ветвления и не более одного оператора цикла. Члены жюри подготовили n различных задач с решениями и представили их на рассмотрение председателю жюри. Он подсчитал количество операторов ветвления и операторов циклов в решении каждой задачи и теперь хочет понять, какие из преложенных ему задач являются простыми.

Вам дано n описаний задач, каждое описание представляет собой два числа: количество операторов ветвления и количество операторов цикла, использующихся в решении задачи. Необходимо для каждой задачи выяснить, является ли она простой.

Формат входных данных	Первая строка содержит одно целое число n ($1 \le n \le 121$) — количество задач. Далее, в n строках задано по два целых числа i и f ($0 \le i, f \le 10$) — количество операторов ветвления и количество операторов цикла, использующихся в решении соответствующей задачи.
Формат выходных данных	Для каждой задачи выведите в отдельной строке «`Yes`», если задача является простой, и «`No`» в противном случае.

	Входные данные	Выходные данные
Примеры	2 2 1 2 2	Yes No

"B" Ключ к шифру

Ограничение по времени	2 секунды
Ограничение по памяти	256 мегабайт

Все те сотни лет, в течении которых существуют государства, границы и периодически возникающие конфликты между странами, существует и система шпионажа. Агент, нелегально или легально проживающий в одной стране, отправляет донесения, содержащие в себе государственную тайну, в другую страну. Способы передачи донесений непрерывно меняются. Сто лет назад это были бумажные письма, пятьдесят лет назад — радиограммы, сейчас же есть возможность отправить донесение даже по электронной почте. Однако, одна характерная черта всех донесений не изменилась и не изменится никогда — любое информационное сообщение можно перехватить. Чтобы застраховаться от такой возможности, донесения шифруются так, чтобы прочитать их мог только тот, кому они предназначены. При этом обычно используется некоторый *ключ* — сравнительно небольшая строка, часто не имеющая смысла, с помощью которой

донесение можно расшифровать. А чтобы, в случае поимки агента, он не мог сообщить ключ заинтересованным лицам, каждому донесению соответствует свой ключ, который отправляется вместе с донесением, тоже зашифрованный, но не очень сложным способом. Вам предстоит научиться получать этот ключ по его зашифрованной версии для некоторого алгоритма шифрования.

Чтобы зашифровать непустой ключ *s*, агент сначала выбирает такую строку *t*, что строка *s* является префиксом строки *t*, а развернутая строка *s* — суффиксом строки *t*. При этом в строке *t* могут быть символы, не имеющие отношения к строке *s*. После этого слева к строке *t* дописывается некоторое случайное (возможно, нулевое) количество случайных символов, справа же дописывается точно такое же количество, возможно других, случайных символов. Теперь строка *t* представляет из себя зашифрованный ключ *s*.

Понятно, что при попытках восстановить сам ключ, то есть исходную строку *s*, вариантов может получиться несколько. Поэтому было принято решение, что ключ должен быть самой длинной строкой из всех возможных вариантов, а в случае, если и таких строк несколько — то такой строкой, что количество случайно дописанных слева и справа символов было минимально, то есть строка *t* имеет максимальную длину. Вам необходимо реализовать алгоритм восстановления ключа *s* по его зашифрованной версии.

Формат входных данных	Первая строка содержит одно целое число *n* — количество ключей, которые вам необходимо расшифровать. Следующие *n* строк содержат зашифрованные версии искомых ключей по одной на строке. Каждая зашифрованная версия содержит только строчные буквы латинского алфавита. Суммарная длина всех зашифрованных ключей не превышает 100000 символов. Гарантируется, что для каждой зашифрованной версии существует хотя бы один подходящий непустой ключ.
Формат выходных данных	Выведите *n* расшифрованных ключей, по одному на строке.

Примеры

Входные данные	Выходные данные
3 ababc ababa cxbaydzabxe	bab ababa xba

"C" Тетраэдр

Ограничение по времени	2 секунды
Ограничение по памяти	256 мегабайт

Вопреки известной поговорке «спички детям не игрушка», один мальчик все еще очень любит играть со спичками. Но он не балуется ими, не разжигает огонь, а решает различные головоломки. Например, он умеет приравнивать число девять к числу одиннадцать, переложив только одну спичку.

Недавно родители этого мальчика подарили ему несколько наборов, каждый из которых состоит из шести спичек. Мальчик начал собирать из них различные трехмерные

геометрические фигуры. Он уже собрал много различных фигур, но теперь ему стало интересно: из каких наборов возможно склеить каркас тетраэдра ненулевого объема при помощи шести спичек из набора и клея? Ломать спички нельзя и ни одна из спичек не должна выступать за каркас.

Ваша задача состоит в том, чтобы по известным длинам спичек для каждого набора проверить, можно ли из них склеить каркас тетраэдра.

Формат входных данных	Первая строка содержит целое число n ($1 \le n \le 1000$) — количество наборов, которые необходимо проверить. Далее, в n строках задано по шесть целых чисел, лежащих в диапозоне от 1 до 1000 — длины спичек в i-м наборе.
Формат выходных данных	Выведите n строк, где в i-й строке требуется вывести «Yes», если из i-го набора можно собрать тетраэдр не нулевого объема, или «No» — в противном случае.

Примеры

Входные данные	Выходные данные
4 1 1 1 1 1 1 1 2 3 1 2 3 1 2 1 2 1 2 1 2 2 1 2 2	Yes No Yes Yes

"D" Задачи

Ограничение по времени	2 секунды
Ограничение по памяти	256 мегабайт

При подготовке соревнований по программированию одна из проблем, встающих перед организаторами — выбор задач, которые будут в этом соревновании. Чтобы соревнование было интересным, необходимо, чтобы задачи в нем были разной сложности. Также, необходимо, чтобы задачи были на разные темы. Ведь кому интересно решать одинаковые задачи? К счастью, у организаторов есть множество задач, и им есть, из чего выбирать.

У вас есть несколько задач. Для каждой задачи известна ее уровень сложости — целое число, и то, какие темы она затрагивает. Вам нужно выбрать такое подмножество задач, чтобы выполнялись следующие требования:

- Выбрана ровно одна задача каждого уровня сложности
- Каждая тема представлена
- Никакие две задачи не пересекаются по темам

Сформируйте из имеющихся у вас задач набор задач на соревнование, удовлетворяющий написанным выше требованиям.

Формат входных данных	Первая строка содержит три целых числа n ($1 \le n \le 300$) — количество задач, d ($1 \le d \le 50$) — количество уровней сложности задач и m ($1 \le m \le 18$) — количество различных тем. Далее, в n строках задается

описание задач.

Каждая задача задается в следующем формате: c_i k_i t_{i1} t_{i2} ... t_{iki},
где c_i ($1 \le c_i \le d$) — сложность i-й задачи, k_i ($0 \le k_i \le m$) — количество тем, которые затрагивает данная задача, t_{ij} ($1 \le t_{ij} \le m$) — темы, которые затрагивает i-я задача. Внутри одной задачи все t_{ij} различны.

Формат выходных данных

Выведите в первой строке «OK», если требуемый набор задач найден, и «Impossible» в противном случае. Если ответ есть, выведите в следующей строке d целых чисел — номера задач, которые входят в требуемый набор.

Примеры

Входные данные	Выходные данные
4 2 3 1 1 1 1 2 1 2 2 2 1 2 2 2 2 3	OK 1 4
1 2 1 1 1 1	Impossible

"E" Текстовый редактор

Ограничение по времени: 2 секунды

Ограничение по памяти: 256 мегабайт

Маленький Иван недавно решил написать свой собственный текстовый редактор. Этот текстовый редактор он собирается использовать для написания сложных программ, поэтому ему очень важна функция выделения парных скобок.

На ранней стадии текстовый редактор должен поддерживать всего одну операцию — изменение символа на позиции i. При этом редактор не разрешает иметь текст длиной более n символов. Каждый раз, когда новый символ является открывающей или закрывающей круглой скобкой, редактор должен подсвечивать парную ей скобку, если таковая имеется.

Определим понятие парной скобки. Пусть на позиции i в тексте располагается открывающая скобка. Тогда парной к ней будет закрывающая скобка на позиции j, для которой верны следующие свойства:

- $i < j$;
- если рассмотреть текст с позиции i до позиции j включительно и удалить из него все символы, не являющиеся круглыми скобками, то получится правильная скобочная последовательность;
- j минимально.

Аналогичным образом определяется парная скобка для закрывающей скобки.

К сожалению, Ивану не удалось реализовать желаемую функциональность. Ваша задача состоит в том, чтобы по заданным операциям, для каждой операции изменения символа на круглую скобку находить к ней парную.

Формат входных данных

Первая строка содержит целые числа n ($1 \le n \le 100000$) — максимальная длина текста, и m ($1 \le m \le 100000$) — число операций модификации символа.

Далее, в m строках описаны изменения операции изменения символа в виде целого числа i — позиции, в которой изменяется символ, и нового символа c ($1 \le i \le n$, c является строчной буквой латинского алфавита, открывающей или закрывающей круглой скобкой). Изначально считается, что текст состоит из n строчных латинских букв “a”.

Формат выходных данных

Для каждой операции изменения символа на скобку, необходимо на отдельной строке вывести позицию парной ей скобки. Если парной скобки не существует, в соответствующей строке выведите число -1.

Примеры

Входные данные	Выходные данные
3 4 1 (3) 2) 3)	-1 1 1 -1
6 6 2 m 3 a 4 i 5 l 6) 1 (	-1 6