Задача A. Поле математических чудес

Имя входного файла:	`money.in`
Имя выходного файла:	`money.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	64 мегабайта

Недавно на известной телеигре «Поле математических чудес» игроку предложили поиграть в следующую игру.

Исходно у игрока n рублей. Каждый ход происходит следующее. Если n четно, то игрок отдает ведущему половину своих денег. Иначе ведущий дает игроку $2n+1$ рублей (всего у игрока становится $3n+1$ рубль). Перед каждым ходом игрок может решить, что достаточно, и забрать деньги. Также игра заканчивается, если у игрока остается один рубль.

По заданному n выясните, какое максимальное количество денег мог забрать себе игрок.

Формат входного файла

Входной файл содержит одно число n ($1 \le n \le 100\,000$).

Формат выходного файла

Выведите в выходной файл одно число — максимальное количество денег, которое мог забрать себе игрок.

Пример

money.in	money.out
11	52
27	9232

Задача B. Строй новобранцев

Имя входного файла:	`formation.in`
Имя выходного файла:	`formation.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	64 мегабайта

—Напрааааа-во! Налееее-во! Крууууу-гом! Шаааагом марш! — орал, срывая голос, товарищ сержант на толпу новобранцев. В конце концов, он понял, что теперешние новобранцы — не те, что были во времена его молодости, даже «лево» и «право» различить не могут. Очевидно, умные юноши в свое время участвовали в интернет-олимпиадах и поэтому поступили в университеты.

Поэтому товарищ сержант решил немного подучить ребят самостоятельно. В первую очередь он научит их различать «право» и «лево». Делать это он будет так.

N новобранцев, пронумерованных от 1 до N, разделены на два множества: *строй* и *толпа*. Вначале *строй* состоит из новобранца номер 1, все остальные составляют *толпу*. В любой момент времени *строй* стоит в один ряд по прямой.

Товарищ сержант может использовать четыре команды. Вот они.

- «I, встать в строй слева от J.» Эта команда заставляет новобранца номер I, находящегося в *толпе*, встать слева от новобранца номер J, находящегося в *строю*.
- «I, встать в строй справа от J.» Эта команда действует аналогично предыдущей, за исключением того, что I встает справа от J.
- «I, выйти из строя.» Эта команда заставляет выйти из строя новобранца номер I. После этого он присоединяется к *толпе*.
- «I, назвать соседей.» Эта команда заставляет глубоко задуматься новобранца номер I, стоящего в *строю*, и назвать номера своих соседей по *строю*, сначала левого, потом правого. Если кто-то из них отсутствует (новобранец находится на краю ряда), то вместо соответствующего номера он должен назвать 0.

Известно, что ни в каком случае строй не остается пустым.

Иногда строй становится слишком большим, и товарищ сержант уже не может проверять сам, правильно ли отвечает новобранец. Поэтому он попросил вас написать программу, которая помогает ему в нелегком деле обучения молодежи и выдает правильные ответы для его команд.

Формат входного файла

В первой строке находятся два числа N ($1 \le N \le 50000$) и M ($1 \le M \le 50000$) — количество новобранцев и команд соответственно. Следующие M строк содержат команды, одна команда на строку.

Каждая команда — одна из следующих:

- `left I J` — соответствует команде «I, встать в строй слева от J.»
- `right I J` — «I, встать в строй справа от J.»
- `leave I` — «I, выйти из строя.»
- `name I` — «I, назвать соседей.»

Гарантируется, что все команды корректны, например, `leave I` не будет заставлять выйти из строя новобранца, стоящего в *толпе*. Также гарантируется, что *строй* никогда не будет пустым.

Формат выходного файла

Для каждой строки, содержащей `name I`, выведите в отдельной строке два числа — номера левого и правого соседа новобранца номер I. Если кто-то из соседей отсутствует, выведите ноль вместо его номера.

Пример

formation.in	formation.out
3 3 left 2 1 right 3 1 name 1	2 3
3 4 left 2 1 right 3 1 leave 1 name 2	0 3

Задача C. Круглый дом

Имя входного файла:	`house.in`
Имя выходного файла:	`house.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	64 мегабайта

Представьте себе, что вы находитесь в круглом доме. В нем N комнат, и все комнаты расположены на окружности. Точное число комнат N вам неизвестно, однако вы знаете, что оно положительно и не превышает M. Вы можете переходить в следующую или предыдущую по кругу комнату, определять, включен ли свет в текущей комнате, а также включать и выключать свет в текущей комнате. В начальный момент времени вы находитесь в некоторой комнате, и в каждой комнате лампочка может быть как включенной, так и выключенной. Необходимо определить, сколько комнат в круглом доме.

Вам дается число M. В выходной файл вы должны вывести программу на упрощенном варианте языка Паскаль, которая определяет число комнат в доме.

Вашей программе предоставляются следующие процедуры и функции:

- `procedure next` перемещает вас в следующую по часовой стрелке комнату.
- `procedure prev` перемещает вас в следующую против часовой стрелки комнату.
- `procedure switch` изменяет состояние лампочки в текущей комнате на противоположное.
- `procedure answer(numberOfRooms: integer)`— вызывая эту процедуру с параметром `numberOfRooms`, вы утверждаете, что в доме ровно `numberOfRooms` комнат. Процедура не возвращает управление программе.
- `function check: boolean` возвращает `true`, если в текущей комнате горит свет, и `false`, если не горит.

Структура программы описывается следующим образом:

⟨program⟩ ⟶ `begin` ⟨list-of-statements⟩ `end.`
⟨list-of-statements⟩ ⟶ ε | ⟨statement⟩ `;` ⟨list-of-statements⟩
⟨statement⟩ ⟶ ⟨next⟩ | ⟨prev⟩ | ⟨answer⟩ | ⟨switch⟩ | ⟨if⟩
⟨next⟩ ⟶ `next`
⟨prev⟩ ⟶ `prev`
⟨switch⟩ ⟶ `switch`
⟨answer⟩ ⟶ `answer(` ⟨integer-number⟩ `)`
⟨if⟩ ⟶ `if check then` ⟨compound⟩ | `if check then` ⟨compound⟩ `else` ⟨compound⟩
⟨compound⟩ ⟶ ⟨statement⟩ | `begin` ⟨list-of-statements⟩ `end`
⟨integer-number⟩ ⟶ *any positive integer number*

В частности, вы не можете использовать переменные, функции и процедуры, кроме данных вам процедур и функций, а также циклы.

Как и в языке Паскаль, любые две подряд идущие смысловые единицы могут быть разделены произвольным числом пробелов, переводов строки и табуляций. Регистр символов не важен.

Ваша программа должна произвести суммарно не более 200 вызовов процедур `next`, `prev`, `answer`, `switch` и функции `check` для определения числа комнат.

Формат входного файла

В первой строке входного файла находится число M ($1 \le M \le 22$). Известно, что количество комнат в круглом доме положительно и не превышает M.

Формат выходного файла

Выведите программу на описанном варианте языка Паскаль, определяющую количество комнат в круглом доме. Объем программы не должен превышать 100 килобайт.

Проверка вывода будет происходить следующим образом. Если программа не соответствует описанному выше синтаксису, то результатом проверки будет `Presentation Error`. Иначе, ваша программа будет проверена на некотором наборе тестов, соответствующем входному файлу. Если для каждого из них ответ будет правильным, а число вызовов процедур не превысит 200, то результатом проверки будет `Accepted`, иначе `Wrong Answer`.

Примеры

house.in	house.out

1	`begin`
	`  answer(1);`
	`end.`

2	`begin`
	`  if check then switch;`
	`  next;`
	`  if check then answer(2) else begin`
	`  switch;`
	`  next;`
	`  if check then answer(1) else answer(2);`
	`  end;`
	`end.`

Задача D. Двоичная новогодняя елка

Имя входного файла:	`tree.in`
Имя выходного файла:	`tree.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	64 мегабайта

Народ Байтландии празднует новый год. В отличие от всех остальных стран, байтландцы украшают к новому году не обычные елки, а двоичные елки. Двоичная елка представляет собой дерево с корнем, каждая вершина которого имеет не более двух детей.

Отличная двоичная елка с n вершинами была приготовлена для установки на главной площади Байтландской столицы. Однако сначала ее надо перевезти в столицу. Для транспортировки планируется использовать железную дорогу. Но, к сожалению, стандартный железнодорожный байтландский вагон может перевезти только дерево, имеющее не больше k вершин.

Для транспортировки решено было разделить елку на части. Для этого некоторые ребра решено было временно распилить, а после перевозки соединить обратно. Каждый вагон будет перевозить ровно одну часть дерева.

Конечно хотелось бы минимизировать количество используемых вагонов. Однако министру транспорта Байтландии эта задача показалась слишком сложной. Поэтому решено было распилить елку на несколько частей таким образом, чтобы количество частей не превышало $\lceil 2n/k \rceil$ (именно такое количество вагонов было запрошено для транспортировки).

Но в министерстве не могут решить даже такую задачу. Помогите им! По заданному двоичному дереву найдите способ распилить некоторые его ребра так, чтобы каждая из его частей имела не более k вершин, а общее количество частей не превышало $\lceil 2n/k \rceil$.

Формат входного файла

Первая строка входного файла содержит n — количеств вершин дерева, и k — максимальное количество вершин, которое может быть у дерева, чтобы его можно было перевезти в вагоне ($1 \le n \le 100\,000$, $1 \le k \le n$). Следующие $n-1$ строк описывают ребра дерево. Каждое ребро описывается двумя номерами вершин: номером родителя и номером ребенка. Гарантируется, что дерево является двоичным. Вершины пронумерованы от 1 до n, корень имеет номер 1.

Формат выходного файла

Первая строка выходного файла должная содержать число l — количество ребер, которые следует перепилить. Вторая строк должна содержать l целых чисел — номера ребер, которые следует перепилить. Ребра пронумерованы от 1 до $n-1$ в том порядке, в котором они заданы во входном файле.

Если дерево распилить указанным образом невозможно, выведите $l = -1$.

Пример

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