Задача A. Boxes

Имя входного файла:	`boxes.in`
Имя выходного файла:	`boxes.out`
Ограничение по времени:	2 seconds
Ограничение по памяти:	256 mebibytes

Разработанное ещё в СССР устройство для спасения леса с кодовым названием 0xFF стало в результате недавней рекламной кампании очень популярным. Количество заказов на это устройство увеличилось, соответственно, изготовители решили поменять алгоритм упаковки, чтобы соответствовать современным условиям доставки.

При упаковке изделия 0xFF помещают в N коробок. Коробки в каждом наборе располагают в порядке убывания вместимости и нумеруют от 1 до N.

Таким образом самая большая коробка получает номер 1, а самая маленькая, в которой непосредственно содержится агрегат, номер N.

Для упаковки используют следующий алгоритм: за один шаг каждая коробка, находящаяся на чётной позиции, помещается (вместе со всеми внутренними коробками) внутрь ближайшей слева коробки. Эта операция продолжается до тех пор, пока все коробки не будут упакованы в одну. При этом каждая коробка содержит инструкцию по распаковке следующей.

После упаковки достаточно большого количества изделий обнаружилось, что оборудование, печатающее текст на коробках с номерами X и Y, сломалось, и соответствующие коробки оказались неокрашены. Для ускорения доступа к коробкам с этими номерами требуется восстановить часть шагов алгоритма упаковки, выполнявшихся до тех пор, пока одна из них не оказалась внутри другой.

i-ый шаг алгоритма описывается указанием четырёх чисел $X_{Li}, X_{Ri}, Y_{Li}, Y_{Ri}$, обозначающих, что после упаковки коробки с номером X_{Ri} в коробку с номером X_{Li}, коробка X оказалась внутри X_{Li}. Аналогично для коробки Y. Последний шаг алгоритма описывается двумя числами L_F, R_F — после упаковки коробки с номером R_F в коробку с номером L_F оказалось, что коробки X и Y находятся внутри коробки L_F.

Формат входного файла

Входной файл содержит три целых числа — N, X, Y ($2 \leqslant N \leqslant 2^{30}$, $1 \leqslant X < Y \leqslant N$).

Формат выходного файла

Выходной файл должен содержать подробный процесс упаковки коробок:

- Каждая строка, описывающая один шаг алгоритма, кроме последней, должна содержать четыре целых числа — $X_{Li}, X_{Ri}, Y_{Li}, Y_{Ri}$.
- Последняя строчка должна содержать два целых числа — L_F, R_F

В случае, когда на каком-либо шаге в коробку ничего не упаковывается, указать, что коробка помещается сама в себя.

Примеры

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

Задача B. Distribution

Имя входного файла:	`distribution.in`
Имя выходного файла:	`distribution.out`
Ограничение по времени:	2 seconds
Ограничение по памяти:	256 Mebibytes

Изделия 0xFF для физических и юридических лиц продаются по следующей схеме.

- Каждый вновь пришедший покупатель должен взять выставленный ему счёт в бухгалтерии.
- Если покупатель оплачивал по безналичному расчёту, то он сразу отправляется на склад, если по наличному — то сначала в кассу, а затем на склад.

Так как реклама оказалась действенной, то в каждом из мест существует очередь.

Некоторые покупатели за наличный расчёт сразу занимают две очереди, остальные — сначала идут в кассу, а потом занимают очередь на склад.

Если к моменту, когда покупатель, занявший обе очереди, выходит из кассы, его очередь на склад уже прошла, то он становится в конец этой очереди.

Если покупатель выходит из помещения кассы в ту же минуту, что начинается его очередь на склад, то считается, что он её пропустил (см. пример 2).

По заданной последовательности появления покупателей выясните время, которое понадобится каждому из покупателей, чтобы стать владельцем изделия 0xFF.

При этом следует считать, что для любого покупателя длительность обслуживания кассиром равна P минут, длительность обслуживания на складе равна Q минут, а время взятия счёта равно нулю.

Формат входного файла

Входной файл содержит числа N, P, Q, за которыми следуют N наборов чисел ($1 \leqslant N \leqslant 100$, $1 \leqslant P, Q \leqslant 15$).

В каждом наборе первое число означает время прихода покупателя t_i, измеренное в минутах с начала рабочего дня. Далее идёт число, обозначающее способ покупки для i-го покупателя — 1, если за наличный расчёт и 2, если за безналичный ($0 \leqslant t_i \leqslant 1000$, $t_i < t_{i+1}$).

Для покупающих за наличный расчёт, далее идет число 1 или 2, обозначающее, займет ли покупатель место в одной очереди или в двух.

Формат выходного файла

Выходной файл должен содержать N чисел — время обслуживания каждого покупателя в минутах.

Примеры

distribution.in	distribution.out
2 6 12 0 1 1 6 1 1	18 24
2 6 6 0 1 2 6 2	18 6
3 7 14 0 1 2 3 2 5 2	45 14 26
2 6 6 0 1 1 6 2	18 6

Задача C. Repair them all!

Имя входного файла:	`repair.in`
Имя выходного файла:	`repair.out`
Ограничение по времени:	2 seconds
Ограничение по памяти:	256 mebibytes

В компании «Curvedhand», занимающаяся ремонтом известных агрегатов советской разработки, предназначенных для спасения леса, существует отдел, специализирующийся на ремонте электронных схем.

Любая электронная схема состоит из узлов и соединений между этими узлами. Узлы в схеме могут быть соединены с любым другим узлом, в том числе и с самим собой, кроме того два узла могут иметь несколько соединений, при этом ток от любого узла может дойти до любого другого узла через некоторые соединения. В каждой схеме есть два специальных узла, к которым подключается устройство питания, таким образом, ток через схему бежит от одного из этих узлов к другому. Соединения между узлами на советских секретных заводах паяли очень качественно, поэтому соединения не перегорают, но вот сделанные на болгарских и румынских заводах узлы часто выходят из строя, при этом ток через них перестает проходить.

Диагностируя поломку, инженеры отдела действуют по принципу Оккама: предполагают, что если ток через схему не идёт, то из строя вышло как можно меньше узлов. Зная схему пайки и то, что узлы, подсоединенные к источнику питания, работают исправно, помогите инженерам определить, какое наименьшее количество узлов может выйти из строя так, чтобы ток перестал проходить через схему.

Формат входного файла

В первой строке записаны 4 целых числа N, M, A, B. N, M — количество узлов и соединений в схеме соответственно ($2 \leqslant N \leqslant 10^3$, $1 \leqslant M \leqslant 2 \cdot 10^4$).

A, B — номера узлов, подсоединенных к источнику питания ($1 \leqslant A, B \leqslant N$).

В следующих M строках записано по два целых числа A_i, B_i ($1 \leqslant A_i, B_i \leqslant N$, $1 \leqslant i \leqslant M$) — номера двух узлов соединенных друг с другом.

Формат выходного файла

Минимальное число узлов, которые могут выйти из строя, либо 0, если ток будет проходить через схему независимо от работоспособности узлов.

Примеры

repair.in	repair.out
5 5 1 5 1 2 1 3 2 4 3 4 4 5	1
5 5 1 4 1 2 2 3 3 4 4 5 5 1	2
5 10 1 4 1 2 2 4 4 3 1 3 1 5 2 5 4 5 3 5 1 4 2 3	0

Задача D. Revision

Имя входного файла:	`revision.in`
Имя выходного файла:	`revision.out`
Ограничение по времени:	9 seconds
Ограничение по памяти:	256 mebibytes

Популярный блогер обнаружил среди госзаказов один странный заказ — требовалась поставка агрегатов для спасения леса, разработанных ещё в СССР. Блогер предположил, что популярность и уникальность этих агрегатов искусственно завышена с использованием коррупционных схем, и для предотвращения откатов добился проведения ревизии на производящем эти агрегаты заводе.

Оказалось, что склад для хранения готовых агрегатов полностью автоматизирован — одни роботы занимаются доставкой устройств на склад, другие роботы в определённые моменты времени проводят ревизию.

Для того, чтобы получить расписание событий на складе, необходимо запустить следующую программу, ввести значения параметров расписания (n, q, a, b), и на выходе вы получите список событий в хронологическом порядке.

```
// Решение на C++

int n, q, a, b;

int SHORT_RAND(int &x)
{
    return ((x = x * 214013L + 2531011L) >> 16) & 0x7fff;
}

int RAND(int &x)
{
    return (SHORT_RAND(x) << 16) + SHORT_RAND(x);
}

void GEN()
{
    int t, t1, t2;
    for (int i = 0; i < q; i++)
    {
        t = RAND(a);
        if (t & 0x1ff)
        {
            t1 = RAND(b) % n;
            t2 = SHORT_RAND(b) % 100;
            printf("1 %d %d\n", t1 + 1, t2);
        }
        else
        {
            t1 = RAND(b) % n;
            t2 = t1 + RAND(b) % (n - t1);
            printf("2 %d %d\n", t1 + 1, t2 + 1);
        }
    }
}
```

```
int main()
{
    scanf("%d %d %d %d", &n, &q, &a, &b);
    GEN();
    return 0;
}


// Решение на Delphi

var n, q, a, b: integer;

function SHORT_RAND(var x: integer): integer;
var xx: int64;
begin
    xx := x * int64(214013) + int64(2531011);
    result := (xx shr 16) and $7fff;
    x := xx;
end;

function RAND(var x: integer): integer;
begin
    result := (SHORT_RAND(x) shl 16) + SHORT_RAND(x);
end;

procedure GEN;
var t, t1, t2, i: integer;
begin
    for i := 0 to q - 1 do begin
        t := RAND(a);
        if (t and $1ff) > 0 then begin
            t1 := RAND(b) mod n;
            t2 := SHORT_RAND(b) mod 100;
            writeln('1 ', t1 + 1, ' ', t2);
        end
        else begin
            t1 := RAND(b) mod n;
            t2 := t1 + RAND(b) mod (n - t1);
            writeln('2 ', t1 + 1, ' ', t2 + 1);
        end;
    end;
end;

begin
    read(n, q, a, b);
    GEN;
end.
```

Каждое событие в списке расположено в отдельной строке и описывается тройкой целых чисел X, Y, Z. Если $X = 1$, то в этот момент времени некоторый робот доставляет Z собранных агрегатов и загружает их в контейнер Y. Если $X = 2$, то проверяющий робот проводит ревизию содержимого

контейнеров с Y по Z включительно, результатом ревизии является суммарное количество агрегатов в контейнерах. Изначально все контейнеры пусты.

Зная параметры алгоритма генерации расписания событий на складе, определите результаты всех ревизий.

Формат входного файла

Четыре целых числа n, q, a, b ($1 \leqslant n \leqslant 10^5$, $1 \leqslant q \leqslant 4 \cdot 10^7$, $1 \leqslant a, b \leqslant 10^5$).

Формат выходного файла

Результаты всех ревизий в хронологическом порядке по одному в строке.

Пример

revision.in	revision.out
20 55 3534 15	110

Задача E. Special Dates

Имя входного файла:	`special.in`
Имя выходного файла:	`special.out`
Ограничение по времени:	2 seconds
Ограничение по памяти:	256 mebibytes

Во время ревизии на заводе, производящем известные агрегаты советской разработки, предназначенных для спасения леса, был замечен странный факт: особо крупные партии отгружались в «запоминающиеся» даты. Судя по всему, это указывает на внешнее вмешательство в автоматизированный процесс обработки — для человека естественно выбирать запоминающуся дату.

У сотрудников завода дата записывается в виде «число.месяц.год». При этом существует много способов записи даты в данном виде: «число» и «месяц» могут записываться с ведущим нулём, если значение соответствующего параметра не превышает 9, «год» может быть записан либо полностью, либо в виде последних двух цифр.

Например, дата 1 января 2012 года может быть записана как `01.01.2012`, `1.01.12`, `01.1.2012` и т.д.

Назовём дату «запоминающейся», если найдётся такой вариант записи даты, что после удаления точек из записи получившаяся строка удовлетворяет хотя бы одному из следующих условий:

- строка является палиндромом, т.е. одинакова при чтении слева направо и справа налево, например 11 февраля 2011 — `11.02.2011` или 7 ноября 1917 — `7.11.17`;
- строка разбивается на подстроки равной длины, равные между собой, например 20 июля 2007 — `20.07.2007` или 9 августа 1998 — `9.8.98` для разбиения на 2 подстроки, 1 сентября 1919 — `1.9.1919` в случае разбиения на три подстроки.

Требуется написать программу, которая по заданной дате определит ближайшую следующую (включая её саму) «запоминающуюся» дату. Гарантируется, что заданная дата корректна, правила для определения високосности не отличаются от общепринятых.

Формат входного файла

Во входном файле заданы три целых числа — D, M, Y, обозначающих дату, где D — день, M — месяц, Y — год ($1 \leqslant D \leqslant 31$, $1 \leqslant M \leqslant 12$, $1600 \leqslant Y \leqslant 9999$).

Формат выходного файла

Выведите 3 целых числа — D_z, M_z, Y_z, задающих ближайшую «запоминающуюся» дату, где D_z — день, M_z — месяц, Y_z — год, записанные без ведущих нулей.

Примеры

special.in	special.out
11 11 2011	11 11 2011
30 10 1810	1 11 1810
26 11 2152	25 12 2152

Задача F. Total Overprice

Имя входного файла:	`total.in`
Имя выходного файла:	`total.out`
Ограничение по времени:	2 seconds
Ограничение по памяти:	256 mebibytes

Как оказалось, используя недавнюю рекламную кампанию и монопольное положение, владельцы завода искусственно завышали цены на изделие 0xFF.

По сути, цена определялась случайным способом. Например, чтобы определить цену изделия для заказчика из некоторой страны, директор по маркетингу записывает N чисел $a_1, a_2, ..., a_N$ по кругу (по часовой стрелке).

Директор начинает жеребьёвку с числа a_1. Начальная квота региона равняется a_1. Затем директор кидает игральный кубик (на разных гранях кубика нарисовано 1, 2, 3, 4, 5 и 6 точек). Если на кубике выпало число x, то директор отсчитывает x чисел по часовой стрелке. Например, если выпало число 3, то директор перейдёт от числа a_1 к числу a_4. Число, к которому перешёл директор, прибавляется к цене. Директор бросает кубик M раз и каждый раз прибавляет очередное число к цене.

У читателей популярного блога, в котором были описаны эти схемы, возник вопрос — какая наибольшая цена может получиться при такой жеребьёвке. Ваша задача — написать программу, вычисляющую наибольшую цену.

Формат входного файла

Входной файл содержит целые числа NM, за которыми следуют N целых чисел $a_1a_2...a_N$ ($1 \leqslant N \leqslant 1000$, $0 \leqslant M \leqslant 5000$, $0 \leqslant a_i \leqslant 1000$).

Формат выходного файла

Требуется вывести в выходной файл целое число — максимальную цену.

Примеры

total.in	total.out
7 1 1 2 3 4 5 6 7	8
7 2 1 2 3 4 5 6 7	14
5 2 5 0 10 1 6	25