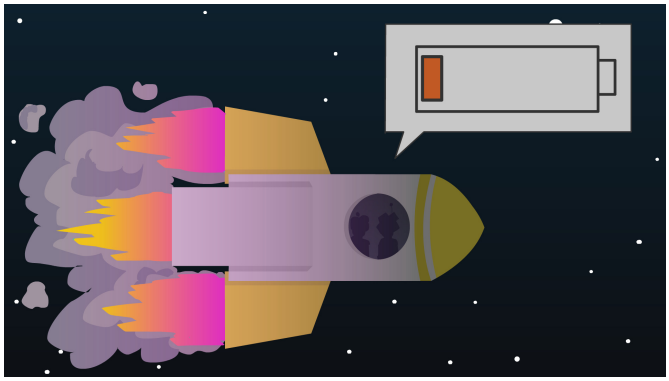
Задача A. Утечка энергии

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

Петя и его друг, робот Petya++, отправились на ракете собственной сборки на Марс. К сожалению, ребята забыли расчитать, сколько энергии нужно для полета, и теперь не уверены, хватит ли им заряда.



Известно следующее. В течение первого часа ракета потратила одну единицу энергии. В течение каждого следующего часа она тратила $x \bmod 10$, единиц энергии, где x — суммарное количество израсходованной энергии за все предыдущие часы, а mod — остаток от деления.

Таким образом, во второй час будет израсходована $1 \bmod 10 = 1$ единица энергии, в третий — $2 \bmod 10 = 2$ единицы энергии ($x = 2$, поскольку в первые два часа суммарно было израсходовано две единицы энергии), и т. д.

Петя хочет быть уверенным, что они смогут долететь до Марса. Поэтому он просит вас рассчитать количество энергии, которое ракета потратит за все время полета.

Формат входных данных

В первой строке входных данных находится целое число t ($1 \leqslant t \leqslant 2 \cdot 10^5$) — количество тестовых примеров.

В каждой из следующих t строк расположено целое число x_i ($1 \leqslant x_i \leqslant 10^{18}$) — длительность полета в i-м тестовом примере.

Формат выходных данных

Выведите t строк, по одной строке на каждый тестовый пример. В i-й строке выведите одно целое число — количество энергии, потраченное за x_i часов полета.

Система оценки

№	Дополнительные ограничения	Баллы за подзадачу	Необходимые подзадачи
1	$x_i \leqslant 10$	7	
2	$x_i \leqslant 10^6$	36	1
3	$x_i \leqslant 10^8$	15	1 - 2
4	Нет дополнительных ограничений	42	1 - 3

Пример

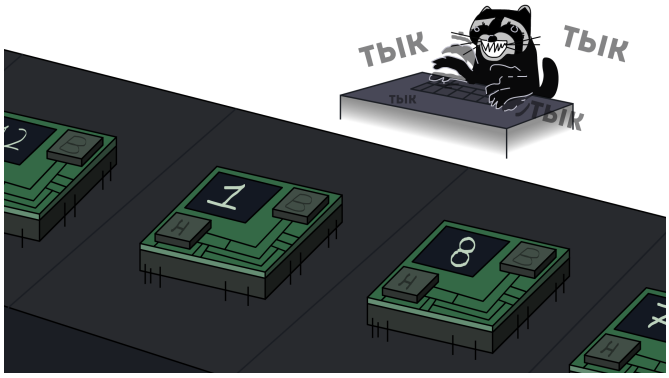
стандартный ввод	стандартный вывод
5 1 2 4 6 9	1 2 8 22 36

Задача B. Енотовидное хулиганство

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

Еноты — странные существа. А марсианские еноты — тем более, поскольку они копаются не в мусоре, а в компьютерных комплектующих.

Один марсианский енот только что проник на автоматическую фабрику. Фабрика собрала n микросхем. Микросхемы, согласно заказу, должны иметь одинаковую мощность, однако енот-хулиган, случайно нажимая на кнопки, изменил настройки фабрики и теперь это условие не выполняется! А именно, i-я микросхема теперь имеет мощность a_i.



Заметив это утром, вы осознали, что у вас не так много времени, чтобы исправить микросхемы, к тому же, вы уже потратили некоторое время на поимку енота.

За одну минуту вы можете выбрать некоторую микросхему и сделать с ней следующую операцию:

- разделить её мощность на некоторое простое число p (при условии, что деление будет без остатка).
- умножить её мощность на некоторое простое число p;

Какое минимальное время вам понадобится, чтобы сделать мощность микросхем снова равной?

Формат входных данных

В первой строке входных данных находится целое число n ($1 \leqslant n \leqslant 5 \cdot 10^5$) — количество микросхем.

Во второй строке входных данных находится n целых чисел a_i ($1 \leqslant a_i \leqslant 10^7$) — мощность i-ой микросхемы.

Формат выходных данных

Выведите одно целое число — минимальное количество минут, необходимое для того, чтобы все мощности стали равными.

Система оценки

№	Дополнительные ограничения	Баллы за подзадачу	Необходимые подзадачи
1	$n \leqslant 2$	9	
2	$a_i \leqslant 2$	10	
3	$a_i \leqslant 3$	6	2
4	$n \leqslant 100$, $a_i \leqslant 100$	13	
5	$n \leqslant 2000$, $a_i \leqslant 2000$	8	4
6	$a_i \leqslant 2000$	9	2 - 5
7	Все a_i — степени двойки	19	2
8	$n \leqslant 10^5$	12	1, 4 - 5
9	Нет дополнительных ограничений	14	1 - 8

Пример

стандартный ввод	стандартный вывод
6 1 4 2 8 10 7	7

Замечание

Одна из оптимальных последовательностей действий в первом примере выглядит следующим образом:

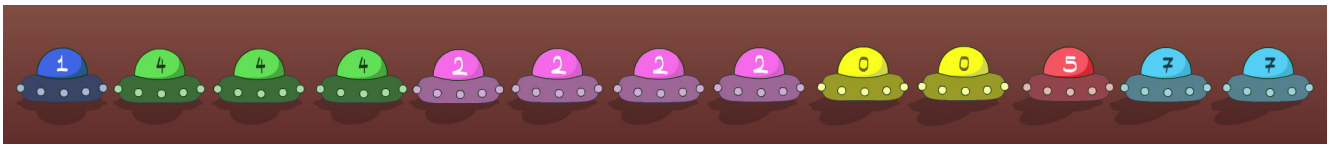
1. Умножить a_1 на 2. Получится $a_1 = 2$.
2. Разделить a_2 на 2. Получится $a_2 = 2$.
3. Разделить a_4 на 2. Получится $a_4 = 4$.
4. Разделить a_4 на 2. Получится $a_4 = 2$.
5. Разделить a_5 на 5. Получится $a_5 = 2$.
6. Разделить a_6 на 7. Получится $a_6 = 1$.
7. Умножить a_6 на 2. Получится $a_6 = 2$.

После этого мощности всех микросхем станут одинаковы и равны 2, что и требовалось сделать.

Задача C. Парковка у космомаркета

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

Иногда приятно просто посидеть и подумать о чем-то простом... Например, сейчас вы сидите в кафе напротив торгового центра «Марс». На парковке неподалеку в ряд стоит n космических машин, i-я из которых имеет цвет a_i. Будем считать, что каждый цвет описывается некоторым целым числом.



Просто сидеть и смотреть на машины скучно, поэтому вы решили посчитать их *случайноцветность*. Эта величина равна минимальному количеству непрерывных подотрезков, на которые можно разбить последовательность машин, чтобы в каждом подотрезке все машины были одного цвета. Например, случайноцветность последовательности 1 1 1 2 1 1 3 3 2 2 равна 5, т. к. ее можно разбить на подотрезки 1 1 1, 2, 1 1, 3 3 и 2 2.

Но машины не стоят стоят на парковке вечно, поэтому в каждый момент происходит одно из следующих событий:

- `ask` l r: вы хотите вычислить случайноцветность на подотрезке $a_l, a_{l+1}, \ldots, a_r$.
- `and` l r x: заменить a_i на a_i and x для всех $l \leqslant i \leqslant r$. Здесь and обозначает операцию побитового И;
- `or` l r x: заменить a_i на a_i or x для всех $l \leqslant i \leqslant r$. Здесь or обозначает операцию побитового ИЛИ;
- `xor` l r x: заменить a_i на a_i xor x для всех $l \leqslant i \leqslant r$. Здесь xor обозначает операцию побитового исключающего ИЛИ;

Поскольку операций бывает много, а ваш кофе уже остывает, вы решили написать программу, которая возьмет и посчитает все за вас.

Формат входных данных

В первой строке входных данных находится два целых числа n и q ($1 \leqslant n, q \leqslant 2 \cdot 10^5$) — количество машин и количество операций.

Во второй строке входных данных находится n целых чисел a_i ($0 \leqslant a_i \leqslant 7$) — цвет i-й космической машины.

В каждой из следующих q строк входных данных находится описание i-го события. Сначала записана строка s, а также два целых числа l, r ($s \in \{\texttt{and}, \texttt{or}, \texttt{xor}, \texttt{ask}\}$, $1 \leqslant l \leqslant r \leqslant n$) — название события, а также подотрезок, на котором оно применяется. Если событие имеет тип `and`, `or` или `xor`, то в конце строки записано еще одно целое число x ($0 \leqslant x \leqslant 7$).

Формат выходных данных

Для каждого события типа `ask` выведите по одному числу в отдельной строке — случайноцветность на заданном подотрезке.

Система оценки

№	Дополнительные ограничения	Баллы за подзадачу	Необходимые подзадачи
1	$n, q \leqslant 2000$	13	
2	$a_i \leqslant 1$	18	
3	$a_i \leqslant 3$	14	2
4	$s \in \{\texttt{and}, \texttt{ask}\}$	20	
5	$s \in \{\texttt{xor}, \texttt{ask}\}$	19	
6	Нет дополнительных ограничений	16	1 - 5

Пример

стандартный ввод	стандартный вывод
13 8 1 4 4 4 2 2 2 2 0 0 5 7 7 ask 1 13 ask 3 9 and 3 9 4 ask 1 13 or 11 13 3 ask 1 13 xor 3 6 4 ask 1 13	6 3 5 4 6

Замечание

Побитовые операция И, ИЛИ, исключающее ИЛИ двух чисел x и y заключается в следующем. Сначала числа переводят в двоичную систему, а затем поразрядно применяют соответствующую логическую операцию. Например, $6 \ \& \ 3 = 2$, поскольку $6 = 110_2$, $3 = 011_2$, а, применив операцию И поразрядно ($1 \ \& \ 0 = 0$, $1 \ \& \ 1 = 1$, $0 \ \& \ 1 = 0$), мы получим $2 = 010_2$.

Логические операции принимают два бита и возвращают один бит в качестве результата. Они устроены следующим образом:

- Операция И возвращает 1, если оба входных бита равны 0, иначе возвращает 1.
- Операция ИЛИ возвращает 1, если хотя бы один входной бит равен 1, иначе возвращает 0.
- Операция исключающее ИЛИ возвращает 1, если входные биты различаются, иначе возвращает 0.

В языке программирования Pascal побитовое операции для чисел a и b обозначается соответственно как `a and b`, `a or b` и `a xor b`, а в языках Python и C++ — `a & b`, `a | b` и `a ^ b`.

Теперь рассмотрим пример.

В первом запросе можно разбить последовательность 1 4 4 4 2 2 2 2 0 0 5 7 7 на шесть отрезков 1, 4 4 4, 2 2 2 2, 0 0, 5 и 7 7, поэтому ответ равен 6.

После второго события случайноцветность последовательности 4 4 2 2 2 2 0 равно трем.

После третьего события последовательность становится равной 1 4 **4 4 0 0 0 0 0** 0 5 7 7.

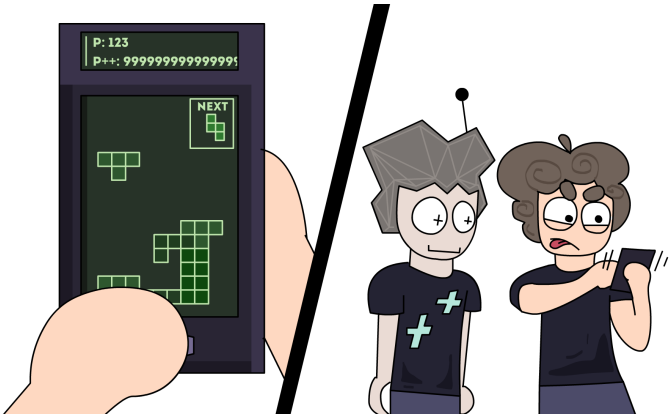
После пятого события последовательность становится равной 1 4 4 4 0 0 0 0 0 0 **7 7 7**.

После седьмого события последовательность становится равной 1 4 **0 0 4 4** 0 0 0 0 7 7 7.

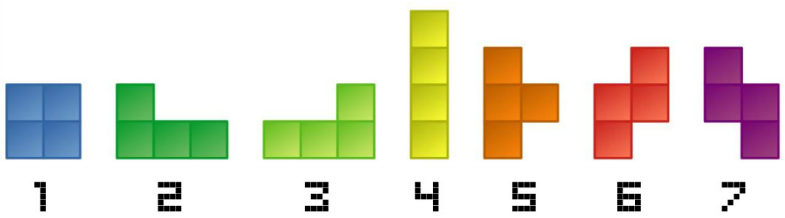
Задача D. Славные времена с космотетрисом

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	4 секунды
Ограничение по памяти:	256 мегабайт

Петя и его друг, робот Petya++, играют в космотетрис.

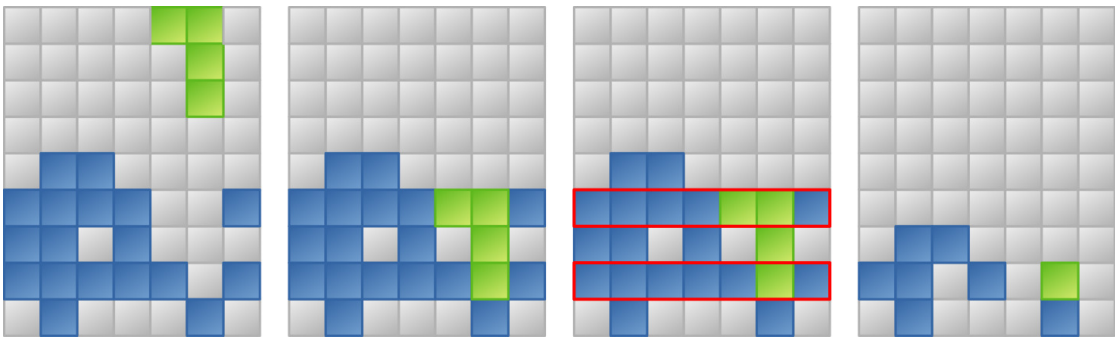


Правила тетриса таковы: Игровое поле представляет собой квадратную сетку, содержащую m столбцов по ширине и $n+4$ столбца по высоте. Каждая ячейка этой сетки либо содержит блок, либо пуста. В игре есть семь типов фигурок, состоящих из блоков:



Каждый ход игроку сообщается, какая фигурка появляется. После чего игрок может выбрать, начиная с какого столбца он может ее разместить, а также повернуть ее на 90, 180 или 270 градусов. При этом фигурка не должна вылезать за пределы поля. Затем фигурка появляется в заданной ориентации и в заданном столбце и начинает падать вниз до тех пор, пока не упрется в уже поставленные блоки или в низ игрового поля. После этого, если на поле оказываются полностью заполненные блоками строки, то они удаляются, а остальные блоки смещаются вниз.

Если после описанных выше действий какой-либо блок оказывается выше, чем нижние n строк игрового поля, то засчитывается поражение. Рассмотрим пример.



Здесь видно, что мы хотим разместить фигурку номер 3 в пятом столбце, повернутую на 270 градусов по часовой стрелке. Далее фигурка падает до тех пор, пока не упрется в блок снизу от нее. После чего мы обнаруживаем, что две строки были заполнены, и удаляем их содержимое.

Обычный тетрис — игра бесконечная, но в космотетрисе есть дополнительные правила. Игра состоит из q уровней, i-й уровень характеризуется количеством фигурок c_i, а также числами $a_1, a_2, \ldots, a_7$, которые задают шанс выпадения фигурок. При этом шанс выпадения i-й фигурки равен $\frac{a_i}{a_1+a_2+\cdots+a_7}$. Игра начинается с первого уровня, после окончания фигурок на i-м уровне начнется уровень номер $i+1$, а после окончания последнего уровня игра заканчивается.

Игрок получает одно очко за каждую выставленную фигурку, кроме той, которая привела к проигрышу. Таким образом, максимальное количество очков равно суммарному количеству фигурок на всех уровнях.

Petya++ является роботом и поэтому умеет проходить космотетрис очень хорошо. Но Петя не унывает и хочет написать стратегию, которая бы играла в космотетрис за него. К сожалению, он пока не представляет, как такую программу написать, поэтому мальчик обратился за вашей помощью.

Формат входных данных

Это интерактивная задача с открытыми тестами.

Всего в задаче 10 тестов. Описание тестов расположено в файлах `input1.txt`, `input2.txt`, ..., `input10.txt`.

Описание теста выглядит следующим образом.

В первой строке находится четыре целых числа t, q, n и m ($0 \leqslant t \leqslant 10$, $t = 0$ для примера из условия) — номер теста, количество уровней и размеры игрового поля.

В каждой из следующих q строк находится восемь целых чисел $c_i, a_1, a_2, \ldots, a_7$ — описание i-го уровня.

Протокол взаимодействия

Сначала на вход подается описание теста в формате, описанном выше (см. раздел «Формат входных данных»). Гарантируется, что описание теста совпадает с содержимым файла `input`t`.txt`, где t — номер теста.

Затем происходит один или несколько ходов, в следующем формате.

Затем вход подается целое число x ($-1 \leqslant x \leqslant 7$) — номер очередной фигурки. При $x \leqslant 0$ ваша программа должна немедленно завершиться, при этом $x = 0$ означает, что больше фигурок нет, а $x = -1$ означает, что вы проиграли.

После этого вы должны принять решение, где разместить фигурку, и вывести два целых числа r и a ($1 \leqslant r \leqslant m$, $a \in \{0, 90, 180, 270\}$) — столбец, начиная с которого размещается фигурка, а также ее угол поворота по часовой стрелке.

Затем наступает следующий ход, и описанный выше процесс повторяется с ввода числа x.

Гарантируется, что очередная фигурка всегда выбирается случайно и равновероятно в соответствии с вероятностями на текущем уровне. При этом генерация случайных чисел может некоторым образом зависеть от ваших предыдущих действий, но при совершении одних и тех же действий на одном тесте вы получите одну и ту же последовательность фигурок.

После каждого действия Вашей программы Вы должны делать перевод строки и сбрасывать поток вывода. Если Вы используете «`writeln`» в Pascal, «`cout << .. << endl;`» в C++, «`print`» в Python, то сброс потока вывода происходит автоматически, ничего дополнительно делать не требуется. Если Вы используете другой способ вывода, рекомендуется делать сброс потока вывода. Обратите внимание, что перевод строки надо выводить в любом случае. Для сброса потока вывода можно использовать «`fflush(stdout)`» в C и C++, «`flush(output)`» в Pascal, «`sys.stdout.flush()`» в Python.

Если Вы получаете вердикт «`Idleness Limit Exceeded`» (`IL`), то это обозначает, что Ваша программа ожидает ввода, но данных в стандартном потоке ввода нет. Это может произойти, например, когда Ваша программа ошибочно ожидает ввода, а она должна вывести информацию для программы жюри, либо завершиться. Если Вы забыли вывести перевод строки либо сбросить поток вывода, то Вы также можете получить этот вердикт.

Система оценки

Если протокол взаимодействия не соответствует условию, то вы получите 0 баллов за тест.
Иначе Ваш балл за тест равен

$$S = 10 \cdot \sqrt{\frac{s}{s_{max}}},$$

где s — набранное вами количество очков, а s_{max} — максимально возможное количество очков (т. е. суммарное количество фигурок на всех уровнях).

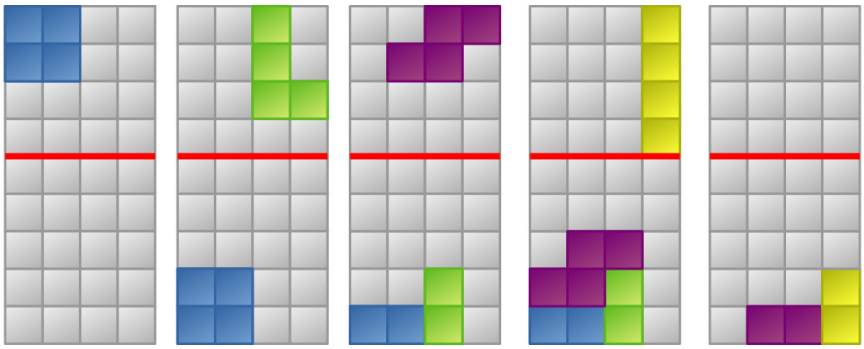
Баллы за каждый тест округляются вверх до сотых и суммируются. Правила округления таковы, что, например, при округлении числа 10.112 вверх до сотых получаем число 10.12.

Пример

стандартный ввод	стандартный вывод
0 1 5 4 4 1 1 1 1 1 1 1 1 3 7 4 0	 1 0 3 90 2 270 4 180

Замечание

Игра в примере выше происходит следующим образом:



Поскольку мы поставили все четыре фигурки и не проиграли, то балл за тест равен 10. Если бы мы проиграли после того, как поставили третью фигурку, то мы набрали бы два очка, и балл за тест был бы равен $10 \cdot \sqrt{\frac{2}{4}} = 7.08$.