Авторы:

Медведев Михаил Геннадиевич – кандидат физико-математических наук, доцент кафедры математической информатики факультета кибернетики Киевского национального университета имени Тараса Шевченко.

Присяжнюк Анатолий Васильевич – учитель-методист высшей категории специализированной школы с углубленным изучением информатики № 17 г. Бердичев Житомирской области.

Жуковский Сергей Станиславович – старший преподаватель кафедры прикладной математики и информатики Житомирского государственного университета имени Ивана Франко, учитель информатики городского лицея №25 имени Н. А. Щорса.

Задачи:

1316 - 1319, 1571 – 1585.

ОГЛАВЛЕНИЕ

ТЕОРЕТИЧЕСКИЙ МАТЕРИАЛ

Вероятностью будем называть меру достоверности случайного события.

Вероятностным пространством называется тройка (Ω, F, P), где

Ω – множество элементарных событий (исходов);

F – сигма-алгебра подмножеств Ω, называемых случайными событиями;

P – вероятностная мера (или вероятность), такая что $P(\Omega) = 1$.

Пример. Рассмотрим эксперимент с бросанием монеты. Вероятностное пространство определим так:

$\Omega = \{0, 1\}$;

$F = \{\{0\}, \{1\}, \{0, 1\}, \varnothing\}$;

$P(\{0\}) = ½, P(\{1\}) = ½, P(\{0, 1\}) = 1, P(\{\varnothing\}) = 0$.

Условная вероятность. Для каждого события A и возможного события B, для которого $P(B) > 0$, условную вероятность события A при условии B определим как

$$P_B(A) = P(AB) / P(B)$$

Формула полной вероятности. Пусть известны:

а) вероятности $P(B_i) = \beta_i$ нескольких исключающих друг друга условий B_i, одно из которых с достоверностью выполняется;

б) условные вероятности $P_{B_i}(A) = a_i$ события A при условии, что выполняется B_i;

Тогда вероятность наступления события A равна

$$P(A) = \sum_{i=1}^{n} P(B_i) \cdot P_{B_i}(A) = \sum_{i=1}^{n} \beta_i a_i$$

Формула Байеса. Пусть известны:

а) вероятности $P(B_i) = \beta_i$ возможных исключающих друг друга предположений B_i;

б) условные вероятности $P_{B_i}(A) = a_i$ события A при условии, что верно предположение B_i;

Тогда условная вероятность того, что верно предположение B_i при условии реализуемости события A равна

$$P_A(B_i) = \frac{P(B_i) \cdot P_{B_i}(A)}{\sum_{j=1}^{n} P(B_j) \cdot P_{B_j}(A)} = \frac{\beta_i a_i}{\sum_{j=1}^{n} \beta_j a_j}$$

Пример. На склад поступило 1000 подшипников. Из них 200 изготовлены на первом заводе, 460 на втором и 340 на третьем. Вероятность того, что подшипник окажется нестандартным, для первого завода равна 0,03, для второго 0,02, для третьего 0,01.

а) Какова вероятность того, что случайно выбранный подшипник является нестандартным?

б) Взятый наудачу подшипник оказался нестандартным. Какова вероятность того, что он изготовлен первым заводом?

Обозначим через H_1, H_2, H_3 вероятности того, что случайно выбранный подшипник изготовлен соответственно первым, вторым или третьим заводом. Они равны

$$P(H_1) = \frac{200}{1000} = 0,2, \quad P(H_2) = \frac{460}{1000} = 0,46, \quad P(H_3) = \frac{340}{1000} = 0,34$$

Пусть A – событие, состоящее в том, что взятый подшипник нестандартный. Из условия задачи следует, что

$$p_1 = P_{H_1}(A) = 0{,}03, \quad p_2 = P_{H_2}(A) = 0{,}02, \quad p_3 = P_{H_3}(A) = 0{,}01$$

По формуле полной вероятности

$$P(A) = P(H_1) * P_{H_1}(A) + P(H_2) * P_{H_2}(A) + P(H_3) * P_{H_3}(A) =$$

$$= 0{,}2 * 0{,}03 + 0{,}46 * 0{,}02 + 0{,}34 * 0{,}01 = 0{,}006 + 0{,}0092 + 0{,}0034 = 0{,}0186$$

Найдем $P_A(H_1)$ – вероятность того, что подшипник, оказавшийся нестандартным, изготовлен первым заводом. По формуле Бейеса имеем

$$P_A(H_1) = \frac{P(H_1)p_1}{P(H_1)p_1 + P(H_2)p_2 + P(H_3)p_3} = \frac{0{,}2 \cdot 0{,}03}{0{,}2 \cdot 0{,}03 + 0{,}46 \cdot 0{,}02 + 0{,}34 \cdot 0{,}01} \approx 0{,}322$$

Таким образом, вероятность гипотезы, что подшипник изготовлен первым заводом, изменилась после того, как стало известно, что он нестандартен.

Пример. В игорном клубе половина игроков честные, половина – шулеры. Вероятность вытащить из колоды короля равна 1/8. Для шулера эта вероятность равна 1. Сидящий перед вами игрок вытаскивает из колоды короля с первого раза. С какой вероятностью перед вами шулер?

Пусть событие A заключается в том, что из колоды вытянут король, B – в том, что игрок шулер. Тогда событие $\overline{B}$ заключается в том, что игрок честный, и $P(A \mid B) = 1$, $P(A \mid \overline{B}) = 1/8$. Если взять первого попавшегося игрока, то он вытянет короля с вероятностью

$$p(A) = p(B) \cdot p(A \mid B) + p(\overline{B}) \cdot p(A \mid \overline{B}) = \frac{1}{2} \cdot 1 + \frac{1}{2} \cdot \frac{1}{8} = \frac{9}{16}$$

Обозначим через X событие, заключающееся в том, что игрок, вытянувший короля, – шулер. Тогда

$$p(X) = \frac{p(B) \cdot p(A \mid B)}{p(A)} = \frac{p(B) \cdot p(A \mid B)}{p(B) \cdot p(A \mid B) + p(\overline{B}) \cdot p(A \mid \overline{B})} = \frac{\frac{1}{2} \cdot 1}{\frac{9}{16}} = \frac{8}{9}$$

Случайной величиной называется измеримая функция, заданная на некотором вероятностном пространстве. Вероятностное поведение случайной величины полностью описывается ее распределением.

Случайная величина ζ имеет **распределение Бернулли**, если она принимает всего два значения: 1 и 0 с вероятностями p и $q = 1 - p$ соответственно. Событие $\zeta = 1$ соответствует успеху, а $\zeta = 0$ – неудаче. Таким образом, $P(\zeta = 1) = p$, $P(\zeta = 0) = q$.

Пусть $X_1, X_2, \ldots, X_n$ – конечная последовательность независимых случайных величин с распределением Бернулли, то есть $P(X_i = 1) = p$, $P(X_i = 0) = q$, $i = 1, 2, \ldots, n$.

Построим случайную величину $Y = \sum_{i=1}^{n} X_i$. Тогда число единиц (успехов) в последовательности имеет **биномиальное распределение**, причем

$$P(Y = k) = C_n^k p^k q^{n-k}$$

Если одновременно устремить число опытов n к бесконечности, а вероятность p к нулю, причем их произведение сохраняет постоянное значение $np = \lambda$, то предельное свойство биномиального распределения можно записать в виде

$$\lim_{n \to \infty} C_n^k p^k q^{n-k} = \frac{\lambda^k}{k!} e^{-\lambda}$$

Распределение Пуассона моделирует случайную величину, представляющую собой число событий, произошедших за фиксированное время, при условии, что данные события происходят с некоторой фиксированной средней интенсивностью и независимо друг от друга. Если через λ обозначить интенсивность событий случайной величины Y, то

$$P(Y = k) = \frac{\lambda^k}{k!} e^{-\lambda}$$

Пример. Устройство состоит из 1000 элементов, работающих независимо друг от друга. Вероятность отказа любого элемента в течение времени T равна 0,002. Найти вероятность того, что за время T откажут ровно три элемента.

Так как по условию $n = 1000$ достаточно велико, а $p = 0{,}002$ мало, то можно воспользоваться распределением Пуассона:

$$P_n(k) = \frac{\lambda^k}{k!} e^{-\lambda}, \text{ где } \lambda = np = 1000 * 0{,}002 = 2$$

Имеем:

$$P_{1000}(3) = \frac{2^3}{3!} e^{-2} \approx \frac{8}{6} \cdot 0{,}13534 \approx 0{,}18$$

Геометрическое распределение. Пусть $X_1, X_2, \ldots, X_n$ – конечная последовательность независимых случайных величин с распределением Бернулли. Построим случайную величину $Y = \min_{1 \le i \le n} \{X_i = 1\} - 1$, обозначающее количество неудач до первого успеха. Случайная величина Y имеет геометрическое распределение с вероятностью успеха p, причем

$$P(Y = n) = q^n \cdot p$$

Пример. Пусть игральный кубик выбрасывается до выпадения первой шестерки. Тогда вероятность того, что потребуется не больше трех бросаний, равна

$$P(Y \le 2) = P(Y = 0) + P(Y = 1) + P(Y = 2) =$$

$$= \left(\frac{5}{6}\right)^0 \cdot \left(\frac{1}{6}\right) + \left(\frac{5}{6}\right)^1 \cdot \left(\frac{1}{6}\right) + \left(\frac{5}{6}\right)^2 \cdot \left(\frac{1}{6}\right) = \frac{1}{6} \cdot \left(1 + \frac{5}{6} + \frac{25}{36}\right) = \frac{1}{6} \cdot \frac{91}{36} \approx 0{,}42$$

Гипергеометрическое распределение. В партии из N изделий имеется M (M < N) доброкачественных и N – M дефектных изделий. Если случайным образом из всей партии выбрать контрольную партию из n изделий, то число доброкачественных изделий в этой партии будет случайной величиной, которую обозначим Y. Ее распределение имеет вид:

$$P(Y = k) = \frac{C_M^k C_{N-M}^{n-k}}{C_N^n}$$

Пример. Пусть в урне находится 5 белых и 45 черных шаров. Вы закрываете глаза и вытаскиваете 10 шаров. Какова вероятность вытянуть ровно 4 белых шара?

В наших обозначениях N = 50, M = 5, $n = 10$, $k = 4$. Искомая вероятность равна

$$P(Y = 4) = \frac{C_5^4 C_{45}^6}{C_{50}^{10}} \approx 0{,}003964583$$

Функция распределения случайной величины $F_\zeta(x) = P(\zeta \le x)$ удовлетворяет трем свойствам:

- $F_\zeta(x)$ является неубывающей функцией;
- $\lim\limits_{x\to-\infty} F_\zeta(x) = 0$, $\lim\limits_{x\to+\infty} F_\zeta(x) = 1$;
- $F_\zeta(x)$ является непрерывной справа.

Пусть случайная величина X распределена **равномерно на отрезке** $[a; b]$. Ее функция распределения имеет вид:

$$F(X) = \begin{cases} 0, x < a \\ \dfrac{x-a}{b-a}, a \le x \le b \\ 1, x > b \end{cases}$$

Цепь Маркова. Пусть $\{E_1, E_2, ..., E_k\}$ – множество состояний некоторой физической системы. В любой момент времени система может находиться в одном состоянии и меняет свое состояние только в моменты $t_1, t_2, ..., t_n, ...$ Для однородных цепей Маркова вероятность p_{ij} перехода системы из состояния E_i в состояние E_j за один шаг зависит только от того, из какого состояния в какое осуществлялся переход.

Вероятности перехода p_{ij} удобно располагать в виде матрицы. Обозначим ее

$$P = \begin{bmatrix} p_{11} & p_{12} & \cdots & p_{1k} \\ p_{21} & p_{22} & \cdots & p_{2k} \\ \cdots & \cdots & \cdots & \cdots \\ p_{k1} & p_{k2} & \cdots & p_{kk} \end{bmatrix}$$

и будем называть ***матрицей перехода*** однородной цепи Маркова за один шаг. Матрица P обладает следующими свойствами:

а) $0 \le p_{ij} \le 1$ б) $\sum\limits_{j=1}^{k} p_{ij} = 1$ $(i = 1, 2, ..., k)$,

т.е. сумма элементов каждой строки матрицы перехода равна единице. Квадратные матрицы, для которых выполняются условия а) и б), называются ***стохастическими***.

Вектор $a = (a_1, a_2, ..., a_k)$, где $a_i = P(E_i)$ – вероятность появления состояния E_i $(i = 1, 2, ..., k)$ в начальном испытании, называется ***вектором начальных вероятностей***.

Матрица перехода $P^{(n)}$ за n шагов находится как P^n.

Если из состояния E_i система может перейти в состояние E_j с положительной вероятностью за конечное число шагов, то говорят, что E_j достижимо из E_i. Состояние E_i называется ***существенным***, если для каждого состояния E_j, достижимого из E_i, E_i достижимо из E_j. В противном случае E_i называется ***несущественным*** состоянием.

УСЛОВИЯ ЗАДАЧ

1316. Одинаковые шары

Корабельная роща. 1898. При сравнении колорита «Корабельной рощи», позднего произведения Шишкина, с колоритом ранних его работ, ясно видно, что художник-пейзажист, подобно своим современникам-жанристам, перешел от локального понимания цвета к скупой, но целостной цветовой гамме. В основе этой гаммы лежит передача объединяющей изображение светотени.

В этой картине Шишкин нашел в цветовом объединении серовато-коричневых стволов елей и зелени мха на первом плане новое для себя тональное понимание цвета.

Картина интересна также новым способом передачи пространства леса. Деревья изображаются не целиком, а как бы срезаются рамой. Ели даются видимыми в непосредственной близости, но когда зритель смотрит на них, он не способен охватить всю картину в целом.

В непрозрачной закрытой урне с небольшим отверстием в крышке находится n шаров (n четно), ровно половина из которых черные, остальные – белые. У Вас есть симметрическая монета, которую Вы начинаете подбрасывать. Если выпадет орел, то Вы извлекаете белый шар, если решка – то черный. Когда в урне остается в точности два шара, то они оказываются одинакового цвета и смысла дальше бросать монету нет (до этого момента в урне обязательно присутствуют хотя бы один белый и хотя бы один черный шар). Какая вероятность того, что произойдет описанная ситуация?

Вход. Каждая строка является отдельным тестом и содержит количество шаров в урне n ($0 < n \le 100000$, n четно).

Выход. Для каждого теста в отдельной строке вывести вероятность того, что произойдет описанная в условии задачи ситуация. Вероятность выводить с 8 знаками после десятичной запятой.

Пример входа

```
6
10
256
```

Пример выхода

```
0.62500000
0.72656250
0.94998552
```

1317. Дни рождения

Рожь. 1878. *Пейзаж стал классическим и программным произведением, как для самого художника, так и для художественной критики его времени. На одном из эскизов к картине И.И. Шишкин сделал запись, которую можно считать его творческим кредо: «Раздолье, простор, угодье, рожь, Божья благодать, русское богатство».*

Лирическое отношение к отечественной природе, воспетое И.И. Шишкиным, было ярко выражено в стихах Н.А. Некрасова:

Все рожь кругом, как степь живая,
Ни замков, ни морей, ни гор.
Спасибо, сторона родная,
За твой врачующий простор.

Известно, что в группе из 23 или более человек вероятность того, что хотя бы у двух из них дни рождения (число и месяц) совпадут, превышает 50%. Этот факт может показаться противоречащим здравому смыслу, так как вероятность одному родиться в определённый день года довольно мала, а вероятность того, что двое родились в конкретный день – ещё меньше, но является верным в соответствии с теорией вероятностей. Таким образом, факт не является парадоксом в строгом научном смысле – логического противоречия в нём нет, а парадокс заключается лишь в различиях между интуитивным восприятием ситуации человеком и результатами математического расчёта.

Для заданного количества людей вычислить вероятность того, что двое из них родились в один день года. Год считать равным 365 дням.

Вход. Каждая строка является отдельным тестом и содержит количество людей n ($1 < n < 400$).

Выход. Для каждого значения n в отдельной строке вывести вероятность того, что хотя бы у двух из n людей дни рождения (число и месяц) совпадают. Искомую вероятность выводить в процентах и округлять до 8 знаков после запятой как указано в примере.

Пример входа

```
2
10
23
366
```

Пример выхода

```
0.27397260%
11.69481777%
50.72972343%
100.00000000%
```

1318. Толстая монета

Вид в окрестностях Дюссельдорфа. 1865. *Окончив в 1860 году Петербургскую Академию художеств с большой золотой медалью, Иван Шишкин получил право на поездку за границу. В 1861—1865 годах молодой художник странствовал по Европе, где посещал мастерские известных художников.*

По заказу Н.Д. Быкова он написал в 1865 году картину «Вид окрестностей Дюссельдорфа», за которую получил звание академика.

В городе N люди настолько полюбили игру с монетой, что двух исходов (орел, решка) им оказалось мало. Поэтому решено было создать толстую монету, одним из исходов бросания которой было бы падение на ребро.

Какую наименьшую толщину должна иметь монета радиуса r, чтобы вероятность ее падения на ребро равнялась $1/n$? Считать, что монета имеет вид прямого кругового цилиндра, а поверхность, на которую она бросается, является клейкой (монета, коснувшись поверхности, падает на ребро или на одно из оснований).

Вход. Каждая строка является отдельным тестом и содержит целочисленный радиус монеты r ($0 < r < 100000$) и целое значение n ($1 < n < 100$).

Выход. Для каждого теста в отдельной строке вывести искомую наименьшую толщину монеты. Результат выводить с точностью до 6 десятичных знаков.

Пример входа

```
1 3
100 20
1000 50
```

Пример выхода

```
0.707107
10.012523
40.008002
```

1319. Дуэлянты

«Среди долины ровныя...». 1883. *«Среди долины ровныя...» — классическое произведение Шишкина. Художник порой увлекался передачей поэтических мотивов, которые он видел в природе. Так, следуя некогда популярной песне А.Ф. Мерзлякова, в этой картине он пытался выразить величие могучего одинокого дуба, возвышавшегося среди широты полей.*

Среди долины ровныя,
На гладкой высоте,
Цветет, растет высокий дуб
В могучей красоте.

Высокий дуб развесистый,
Один у всех в глазах;
Один, один, бедняжечка,
Как рекрут на часах!

Взойдет ли красно солнышко -
Кого под сень принять?
Ударит ли погодушка -
Кто будет защищать?
Ах, скучно одинокому
И дереву расти!
Ах, горько, горько молодцу
Без милой жизнь вести!

Среди долины ровныя Петр Иванович и Илья Васильевич решили выяснить отношения на дуэли. Но никому из них не хотелось умирать. Поэтому они договорились, что каждый из них прибывает к дубу (место встречи) в случайный момент времени между 5 и 6 часами утра и, прождав соперника n минут, удаляется. В случае прибытия последнего в эти n минут дуэль состоится. Вычислить вероятность того, что дуэль закончится поединком.

Вход. Каждая строка является отдельным тестом и содержит целочисленное значение n ($0 < n \le 60$).

Выход. Для каждого теста в отдельной строке вывести вероятность того, что дуэль закончится поединком. Результат выводить с точностью до 6 десятичных знаков.

Пример входа
```
5
50
60
```

Пример выхода
```
0.159722
0.972222
1.000000
```

1571. Какая вероятность?

Вероятность всегда была неотъемлемой частью компьютерных алгоритмов. Там, где детерминированные алгоритмы не могли решить задачу за разумное время, использовались вероятностные алгоритмы. В этой задаче Вам следует определить вероятность выигрыша определенного игрока.

Рассмотрим игру, в которой бросается некоторая вещь (например, кубик), которая имеет несколько исходов. Если у некоторого игрока случается некоторый наперед установленный выигрышный исход (например, выпала цифра 3, или сверху выпал зеленый цвет, или еще что-нибудь), то он объявляется победителем и игра останавливается. Всего имеется n игроков. Вещь подбрасывается игроками последовательно: сначала первым, потом вторым и так далее. Если у n - го игрока выигрышный исход не выпал, то подбрасывание снова совершается первым игроком, потом вторым и так далее по очереди. Необходимо установить вероятность выигрыша i - го игрока.

Вход. Первая строка содержит количество тестов s ($s \le 1000$). Каждая следующая строка является отдельным тестом и содержит три числа: количество игроков n ($n \le 1000$), действительное число p, равное вероятности наступления победного события и номер игрока i ($i \le n$), вероятность выигрыша которого следует подсчитать (игроки пронумерованы числами от 1 до n). Входные данные корректны.

Выход. Для каждого теста в отдельной строке вывести вероятность выигрыша i-го игрока с четырьмя десятичными знаками.

Пример входа
```
2
2 0.166666 1
2 0.166666 2
```

Пример выхода
```
0.5455
0.4545
```

1572. Купоны

Имеется n разнотипных купонов, пронумерованных от 1 до n, и бесконечное количество закрытых коробок. В каждой коробке лежит один купон некоторого типа. Из каждой коробки с равной вероятностью можно извлечь купон любого типа. Какое ожидаемое количество коробок необходимо открыть, чтобы иметь хотя бы по одному купону каждого типа?

Вход. Каждая строка содержит натуральное число n, $1 \le n \le 33$, количество типов купонов.

Выход. Для каждого значения n вывести ожидаемое число коробок, которое надо открыть, чтобы иметь купоны всех типов. Если искомое число коробок целое, то вывести его. Если результат не целый, то вывести его целую часть, пробел, и дробную часть как показано в примере. Дробную часть результата представлять несократимой дробью. Лишних пробелов в конце строк выводить не следует.

Пример входа
```
2
5
17
```

Пример выхода
```
3
   5
11 --
   12
   340463
58 ------
   720720
```

1573. Коровы и машины

В телевизионных конкурсах участников часто просят открыть одну или несколько дверей из заданного множества, за которыми находятся разные призы. В этой задаче мы будем иметь дело с одним из таких конкурсов. Итак, ведущий предложил Вам следующую игру:

Перед Вами находятся три двери. За двумя из них спрятаны коровы, а за третьей приз - автомобиль. После того как Вы выберете дверь, но еще не откроете ее, я дам Вам подсказку, открыв одну из дверей, за которой спрятана корова (я никогда не буду открывать дверь, выбранную Вами, даже если за ней находится корова). Вам следует принять решение - оставить Ваш выбор, или изменить его, выбрав одну из закрытых дверей. Вы выиграете то, что спрятано за дверьми.

Сложно поверить, но в этом примере вероятность выиграть автомобиль равна 2 / 3, если вы всегда будете менять свой выбор, когда ведущий будет давать возможность это сделать (после того как он покажет Вам дверь с коровой). Ответ 2 / 3 кроется в том, что если вы выбрали одну из двух коров, то после смены двери однозначно перейдете к машине, так как ведущий откроет Вам другую корову. Если вы изначально выбрали автомобиль, то перейдете к оставшейся корове и потеряете приз. Таким образом, в двух случаях из трех Вы выберете машину. Если вы решили не менять первоначальный выбор, то вероятность выигрыша, очевидно, составит только 1 / 3.

В этой задаче Вам следует вычислить вероятность выигрыша автомобиля, если входные данные немножко обобщены:

- Задается количество коров
- Задается количество автомобилей (число коров + число машин = общему количеству дверей)
- Задается количество дверей, открываемых ведущим (несколько дверей могут быть открытыми когда Вы решаете менять или не менять Ваш выбор)

Будем считать, что Вы всегда меняете свой выбор после того как ведущий откроет несколько дверей с коровами.

Вход. Содержит несколько тестов. Каждый тест состоит из одной строки, содержащей три целых числа: *ncows* ($1 \le ncows \le 10000$) – количество коров, *ncars* ($1 \le ncars \le 10000$) – количество дверей с машинами и *nshow* ($0 \le nshow < ncows$) – количество дверей, которое открывает ведущий перед тем как Вы будете менять свой выбор.

Выход. Для каждого теста вывести в отдельной строке вероятность выигрыша игрока, если после предложения ведущего он сменит свое мнение. Ответ выводить с 5 десятичными знаками.

Пример входа

```
2 1 1
5 3 2
2000 2700 900
```

Пример выхода

```
0.66667
0.52500
0.71056
```

1574. Бросание кубиков

Бросается n одинаковых игральных кубиков. Найти вероятность того, что сумма чисел на всех кубиках будет как минимум x.

Вход. Состоит из нескольких тестов. Каждый тест состоит из двух целых чисел n ($1 \le n \le 24$) и x ($0 \le x < 150$), смысл которых описан в условии задачи. Последний тест содержит $n = 0$, $x = 0$ и не обрабатывается.

Выход. Для каждого теста в отдельной строке вывести искомую вероятность в виде обыкновенной несократимой дроби в формате, указанном в примере. Все выводимые числа помещаются в беззнаковое 64-битовое целое.

Пример входа

```
3 9
1 7
24 24
15 76
24 56
24 143
23 81
7 38
0 0
```

Пример выхода

```
20/27
0
1
11703055/78364164096
789532654692658645/789730223053602816
25/4738381338321616896
1/2
55/46656
```

1575. Боже! Спаси меня

В комнате имеется n дверей. Если Вы откроете дверь с номером i, то либо через xi часов попадете в безопасное место, либо через x_i часов снова вернетесь в эту же комнату. Вычислить ожидаемое время P (в часах), через которое можно выбраться из комнаты в безопасное место.

Вход. Первая строка содержит количество тестов. Первая строка каждого теста содержит количество дверей n ($1 < n < 100$). Каждая из следующих n строк содержит числа x_i, $0 < |x_i| < 25$ (если x_i положительно, то оно обозначает время, за которое можно попасть в безопасное место; если отрицательно, то $|x_i|$ – это время, через которое Вы снова окажетесь в комнате) и p_i – вероятность открыть i - ую дверь. Сумма всех p_i равна 1.

Выход. Для каждого теста вывести его номер, двоеточие, пробел, а затем фразу "God! Save me", если выбраться из комнаты невозможно или ожидаемое время P (с двумя десятичными знаками), через которое можно выбраться из комнаты в безопасное место.

Пример входа

```
2
3
2 0.33
-3 0.33
-5 0.34
3
2 0.34
-3 0.33
-5 0.33
```

Пример выхода

```
Case 1: 10.15
Case 2: 9.76
```

1576. Потерянный подарок

Стэн шокирован, так как потерял часть подарка Оли. Драгоценным подарком были две коробки, наполненные хрустальными шариками. Шарики в одной коробке были красными, а в другой черными. Когда Оли дарил подарок, он произнес следующие слова с шаловливой улыбкой: "Стэн! Если ты смешаешь шарики из двух коробок и наугад выберешь два, то вероятность того что они будут одного цвета равна ½.". Стэн весело вернулся домой с двумя коробками, но тут неожиданно уронил на пол коробку с черными шариками, которые раскатились по полу. Стэн бросился их собирать, и в результате смог собрать не меньше 70% черных шариков. По количеству красных шариков и собранных черных Вам следует определить количество потерянных черных шариков.

Вход. Каждая строка содержит два натуральных числа r – количество красных шариков и b – количество черных шариков, найденных Стэном. Последняя строка содержит два ноля и не обрабатывается.

Выход. Для каждого теста в отдельной строке вывести ответ. Если количество красных шариков r не соответствует выше описанной истории, то вывести "No. of red balls invalid". Если количество красных шариков верно, но количество найденных черных шариков не корректно, то вывести строку "No. of black balls invalid". Если оба числа r и b корректны, то вывести одно или два целых числа в одной строке. Эти числа указывают на возможное количество потерянных черных шариков. Выводимые числа следует разделять одним пробелом и отсортировать по возрастанию.

Пример входа

```
10 5
11 1
0 0
```

Пример выхода

```
1
No. of red balls invalid
```

1577. Так Вы хотите стать 2^n-эром?

У игрока имеется $1, и ему предстоит последовательно ответить на n вопросов. Перед каждым вопросом он может:

- остановить игру и забрать имеющиеся у него деньги.
- ответить на вопрос. Если ответ неправильный, он покидает игру ни с чем. Если ответ правильный, то денежная сумма удваивается, и игра переходит к следующему вопросу.

Ответив на последний вопрос, игрок забирает деньги. Игрок желает максимизировать ожидаемую сумму выигрыша.

На каждый заданный вопрос игрок может ответить правильно с вероятностью p. Считайте, что вероятность p равномерно распределена на отрезке t .. 1.

Вход. Каждая строка является отдельным тестом и содержит два числа: целое значение n ($1 \le n \le 30$) и действительное t ($0 \le t \le 1$). Последняя строка содержит 0 0 и не обрабатывается.

Выход. Для каждой пары чисел n и t вывести в отдельной строке максимальную ожидаемую сумму выигрыша, если известно, что игрок придерживается наилучшей стратегии. Результат следует выводить с тремя десятичными знаками.

Пример входа

```
1 0.5
1 0.3
2 0.6
24 0.25
0 0
```

Пример выхода

```
1.500
1.357
2.560
230.138
```

1578. Заданная вероятность

n друзей собрались за покупками в супермаркет. Вероятность купить что-либо составляет $p_1, p_2, \ldots, p_n$ соответственно для каждого друга. После посещения магазина оказалось, что в точности r друзей совершили покупки (остальные ничего не купили). Определить вероятность покупательской способности каждого друга при выполнении этого условия.

Вход. Содержит не более 50 тестов. Первая строка каждого теста содержит два числа n ($1 \le n \le 20$) и r ($0 \le r \le n$). Каждая из следующих n строк содержит вероятность покупки i - го друга p_i ($0.1 \le p_i \le 1$). Все вероятности содержат как минимум два знака после десятичной точки. Последний тест содержит $n = r = 0$ и не обрабатывается.

Выход. Для каждого теста вывести его номер, а также n строк. i - ая строка должна содержать вероятность покупательной способности i - го друга при условии, что в точности r друзей совершили покупки. Вероятности следует выводить с 6 знаками после десятичной точки.

Пример входа

```
3 2
0.10
0.20
0.30
5 1
0.10
0.10
0.10
0.10
0.10
0 0
```

Пример выхода

```
Case 1:
0.413043
0.739130
0.847826
```

```
Case 2:
0.200000
0.200000
0.200000
0.200000
0.200000
```

1579. Тройной прыжок

Тройной прыжок совершается следующим образом. Прыгун разгоняется, добегает до определенной отметки и совершает три последовательных прыжка. Победителем является тот, чья суммарная длина прыжков наибольшая.

Вы принимаете участие в соревновании и прыгаете последним. Все Ваши соперники уже совершили прыжки, поэтому их результаты известны.

Первый свой прыжок Вы уже совершили, его длина равна *first*. Длина каждого из оставшихся прыжков может с одинаковой вероятностью принимать любое значение из отрезка [*lower*, *upper*], и не обязательно быть целым. Вам необходимо вычислить вероятность того, что Вы займете i - ое место. Место, занятое Вами, равняется единице плюс количество соперников, которые прыгнули дальше Вас.

Вход. Состоит из нескольких тестов. Первая строка каждого теста содержит значения *lower*, *upper*, *first* ($1 \le lower \le 1000$, $lower \le upper \le 1000$, $lower \le first \le upper$) и количество Ваших соперников n ($1 \le n \le 50$). Вторая строка теста содержит n целых чисел от 1 до 3000 – длины тройных прыжков всех Ваших соперников.

Выход. Для каждого теста в отдельной строке вывести $n + 1$ действительных чисел – соответственно вероятности того что Вы займете первое, второе, третье, ..., последнее место. Все вероятности следует выводить с 4 десятичными знаками.

Пример входа

```
1 2 1 4
1 2 3 4
1 10 5 8
1 2 3 5 10 11 12 19
```

Пример выхода

```
0.5000 0.5000 0.0000 0.0000 0.0000
0.2222 0.6235 0.0556 0.0432 0.0556 0.0000 0.0000 0.0000 0.0000
```

1580. Простой футбол

Вы смотрите футбольный матч и хотите вычислить вероятность того, что хотя бы одна из команд забьет такое количество голов, которое является простым числом. Игра длится 90 минут. Весь матч разделим на 18 пятиминутных интервалов: от 0 до 5 минуты, от 5 до 10 минуты и так далее. Процентная вероятность забить гол первой командой за любую пятиминутку равна *skillOfTeamA*, а второй *skillOfTeamB*. В течение каждой пятиминутки каждая из команд может забить не более одного гола.

Вам необходимо вычислить вероятность того, что хотя бы одна из команд забьет такое количество голов, которое является простым числом.

Вход. Каждая строка содержит два целых числа *skillOfTeamA* и *skillOfTeamB* ($0 \le skillOfTeamA, skillOfTeamB \le 100$), задающие вероятность в процентах забить гол соответственно первой и второй командой за одну пятиминутку.

Выход. Для каждого теста в отдельной строке вывести вероятность того, что хотя бы одна из команд забьет такое количество голов, которое является простым числом. Вероятность следует выводить с 4 цифрами после десятичной точки.

Пример входа

```
50 50
100 100
12 89
```

Пример выхода

```
0.5266
0.0000
0.6772
```

1581. Игра с кубиком

У Джона есть стандартный игральный кубик с 6 гранями, пронумерованных от 1 до 6. Каждый раз после его бросания Джон получает от мамы столько конфет, сколько выпадет косточек на верхней грани. Мальчик хочет узнать, сколько раз ему необходимо бросать кубик, чтобы суммарно получить от мамы не менее *candies* конфет.

Вход. Каждая строка содержит одно число – желаемое количество конфет candies ($1 \le candies \le 10^6$).

Выход. Для каждого значения *candies* в отдельной строке вывести ожидаемое количество бросков кубика. Каждое число следует выводить с 4 цифрами после десятичной точки.

Пример входа

```
1
2
7
47
```

Пример выхода

```
1.0000
1.1667
2.5216
13.9048
```

1582. Стрельба из лазера

Лазерная пушка расположена в точке (0, 0) на плоскости. Целями являются вертикальные отрезки с координатами концов (x[i], y1[i]) – (x[i], y2[i]). Выбирается произвольный угол от $-\pi$ / 2 до π / 2 и производится выстрел. Выстрел под углом $-\pi$ / 2 производится вертикально вниз, 0 – горизонтально вправо, π / 2 – вертикально вверх. Выстрелом является бесконечный луч, исходящий из начала координат. Выстрел попадает в цель, если луч и отрезок цели имеют общую точку.

Вычислить ожидаемое количество целей, которое может быть поражено одним выстрелом. Попадание в цель не меняет движение луча.

Вход. Состоит из нескольких тестов. Первая строка каждого теста содержит количество целей n ($1 \le n \le 50$). Следующие три строки задают координаты целей. i-ое число второй строки каждого теста содержит значение x_i, i-ое число третьей строки - значение $y1_i$, i-ое число четвертой строки каждого теста - значение $y2_i$. Известно, что все координаты целые, значения x_i разные, $1 \le x_i \le 1000$, $-1000 \le y1_i, y2_i \le 1000$.

Выход. Для каждого теста в отдельной строке вывести с 4 цифрами после десятичной точки ожидаемое количество целей, которое может быть поражено одним выстрелом.

Пример входа

```
1
1
-1
1
4
134 298 151 942
-753 -76 19 568
440 689 -39 672
```

Пример выхода

```
0.5000
1.4442
```

1583. Тир

Вы поспорили с другом, что он не попадет в мишень, сделав n выстрелов. Вероятность попадания друга в мишень равна *accuracy*.

Найти максимальное количество выстрелов, при котором Вам есть смысл спорить. Вам есть смысл спорить, если вероятность того, что Ваш друг попадет в цель хотя бы одним из n выстрелов, строго меньше 50%.

Вход. Каждая строка содержит целое число *accuracy* ($1 \le accuracy \le 100$) – вероятность попадания друга в мишень.

Выход. Для каждого теста в отдельной строке вывести максимальное количество выстрелов, при котором Вам есть смысл спорить.

Пример входа

```
40
20
50
1
```

Пример выхода

```
1
3
0
68
```

1584. Произвольное тасование

Произвольное тасование чисел массива A производится согласно следующего алгоритма:

```
n = длина массива A
for i=1 to n
  сгенерировать произвольное число r между 1 и n включительно
  поменять местами A[i] и A[r]
```

Вычислить вероятность того, что заданный массив чисел получен в результате выполнения операции произвольного тасования набора $\{1, 2, \ldots, n\}$. Здесь n равно количеству элементов входного массива.

Вход. Каждая строка является отдельным тестом и содержит значение n ($1 \le n \le 10$), за которым следуют элементы массива A – перестановка чисел от 1 до n.

Выход. Для каждого теста в отдельной строке вывести с 4 цифрами после десятичной точки вероятность того, что входной массив получен в результате выполнения операции произвольного тасования набора $\{1, 2, \ldots, n\}$.

Пример входа

```
1 1
2 1 2
5 4 2 5 1 3
```

Пример выхода

```
1.0000
0.5000
0.0067
```

1585. Игра в лотерею

Возвращаясь домой, вы прочитали объявление: "Выберите m разных чисел между 1 и n включительно. Мы выберем m разных чисел между 1 и n произвольным образом. Если хотя бы k чисел совпадут, то Вы выиграли".

В задаче требуется вычислить вероятность Вашего выигрыша.

Вход. Каждая строка является отдельным тестом и содержит три целых числа n, m и k. Известно, что $2 \le n \le 8$, $1 \le m \le n - 1$, $1 \le k \le m$.

Выход. Для каждого теста в отдельной строке вывести вероятность Вашего выигрыша с 4 цифрами после десятичной точки.

Пример входа

```
3 2 1
8 2 1
8 4 2
```

Пример выхода

```
1.0000
0.4643
0.7571
```

АНАЛИЗ ЗАДАЧ

1316. Одинаковые шары

Вычислим вероятность противоположного события X – того, что последними будут вынуты шары разного цвета. Для этого среди первых $n - 2$ вынутых шаров ровно половина должна быть белыми, половина – черными.

Рассмотрим схему испытаний Бернулли, где вероятность успеха (например, появления белого шара) равна $p = 1/2$, а неуспеха $q = 1 - p = 1/2$. Вероятность того, что среди $n - 2$ вынутых шаров будет ровно $(n - 2) / 2$ белых, равна

$$P(X) = C_{n-2}^{\frac{n}{2}-1} p^{\frac{n}{2}-1} q^{\frac{n}{2}-1} = \frac{(n-2)!}{\left(\left(\frac{n}{2}-1\right)!\right)^2}\left(\frac{1}{2}\right)^{n-2}$$

Остается для каждого входного n вычислить значение P(X) и вывести 1 – P(X).

Хотя P(X) лежит в промежутке от 0 до 1 включительно, возможны проблемы с его непосредственным вычислением. Прологарифмируем равенство:

$$\ln P(X) = \ln(n-2)! - 2 \cdot \ln\left(\frac{n}{2}-1\right)! - (n-2) \cdot \ln 2$$

После вычисления правой части равенства, возведем e в него, получив P(X). Логарифм факториала вычислим как сумму логарифмов:

$$\ln n! = \ln 1 + \ln 2 + \ln 3 + \ldots + \ln n$$

В связи с многократным входом, все ответы следует предвычислить и занести в массив.

Пример. Для n = 6 вероятность того, что последними будут вынуты шары разного цвета, равна

$$C_4^2 p^2 q^2 = 6 \cdot \left(\frac{1}{2}\right)^4 = \frac{6}{16} = \frac{3}{8}$$

Вероятность того, что последними будут вынуты шары одного цвета, равна $1 - \frac{3}{8} = \frac{5}{8} =$ 0,625.

1317. Дни рождения

Рассчитаем сначала, какова вероятность $\overline{p(n)}$ того, что в группе из n человек дни рождения всех людей будут различными. Если $n > 365$, то в силу принципа Дирихле вероятность равна нулю. Если же $n \le 365$, то будем рассуждать следующим образом. Возьмём наугад одного человека из группы и запомним его день рождения. Затем возьмём наугад второго человека, при этом вероятность того, что у него день рождения не совпадёт с днем рождения первого человека, равна 1 – 1/365. Затем возьмём третьего человека, при этом вероятность того, что его день рождения не совпадёт с днями рождения первых двух, равна 1 – 2/365. Рассуждая по аналогии, мы дойдём до последнего человека, для которого вероятность несовпадения его дня рождения со всеми предыдущими будет равна 1 – $(n - 1) / 365$. Перемножая все эти вероятности, получаем вероятность того, что все дни рождения в группе будут различными:

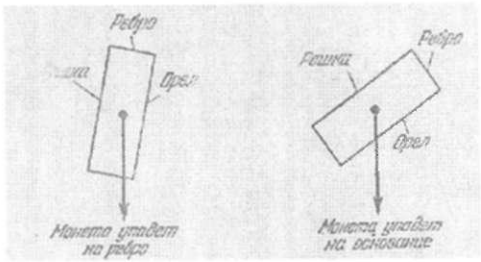
$$\overline{p(n)} = 1 \cdot \left(1 - \frac{1}{365}\right) \cdot \left(1 - \frac{2}{365}\right) \cdot \ldots \cdot \left(1 - \frac{n-1}{365}\right)$$

Тогда вероятность того, что хотя бы у двух человек из n дни рождения совпадут, равна

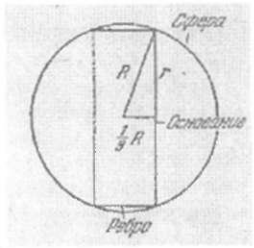
$$p(n) = 1 - \overline{p(n)}$$

1318. Толстая монета

Рассмотрим монету как вписанную в сферу, центр которой совпадает с центром тяжести монеты. Считаем, что поверхность, на которую бросается монета, является клейкой.



Лемма. Поверхность куска сферы, заключенного между двумя параллельными плоскостями, пропорциональна расстоянию между этими плоскостями. Поэтому толщина нашей монеты должна составлять $1 / n$ диаметра сферы (на рисунке показан случай $n = 3$).



Пусть R – радиус сферы, а r – радиус монеты. По теореме Пифагора

$$R^2 = r^2 + \frac{1}{n^2} R^2$$

или

$$\left(1 - \frac{1}{n^2}\right) R^2 = r^2, \; R^2 = \frac{r^2 n^2}{n^2 - 1}, \; R = \frac{rn}{\sqrt{n^2 - 1}}$$

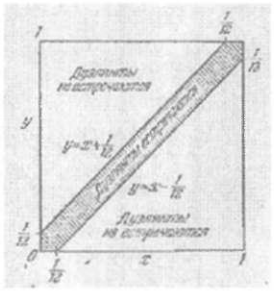
Откуда искомая толщина монеты должна быть как минимум

$$2 * \frac{1}{n} R = \frac{2r}{\sqrt{n^2 - 1}}$$

1319. Дуэлянты

Пусть x и y обозначают время прибытия первого и второго дуэлянтов соответственно, измеренное в долях часа, начиная с 5 часов. Заштрихованная площадь квадрата отвечает случаю, когда дуэлянты встречаются для случая $n = 5$ минут = 1/12 часа. Вероятность того, что дуэлянты не встретятся, равна

$$\left(1-\frac{n}{60}\right)^2 \text{ (сумма площадей двух треугольников)}$$



Соответственно вероятность поединка составляет $1-\left(1-\frac{n}{60}\right)^2$.

1571. Какая вероятность?

Вероятность того, что i - ый игрок на первом же своем броске выиграет, равна $p * (1-p)^{i-1}$. Вероятность того, что i - ый игрок выиграет при втором своем броске, равна $p * (1-p)^{i-1} * (1-p)^n$: для этого необходимо чтобы первый бросок каждого игрока потерпел неудачу (вероятность $(1-p)^n$), далее первые $i-1$ игроков не получили выигрышный исход (вероятность $(1-p)^{i-1}$), и наконец, i - ый игрок выиграл, совершив свой бросок с вероятностью p.

Соответственно, i - ый игрок выиграет на k - ом своем броске с вероятностью

$$p * (1-p)^{i-1} * (1-p)^{kn}.$$

Просуммируем вероятности выигрыша i - ого игрока. Искомая вероятность его выигрыша равна

$$p * (1-p)^{i-1} + p * (1-p)^{i-1} * (1-p)^n + p * (1-p)^{i-1} * (1-p)^{2n} + \ldots + p * (1-p)^{i-1} * (1-p)^{kn} + \ldots =$$
$$= p * (1-p)^{i-1} (1 + (1-p)^n + (1-p)^{2n} + .. + (1-p)^{kn} + \ldots)$$

и образует бесконечную геометрическую прогрессию, сумма которой равна

$$p * (1-p)^{i-1} \frac{1}{1-(1-p)^n} = \frac{p*(1-p)^{i-1}}{1-(1-p)^n}$$

Отдельно следует обработать случай, когда $p = 0$. В таком случае ответом будет 0.

1572. Купоны

Предположим, что у Вас уже имеется $n - k$ разных купонов. Обозначим через a_k ожидаемое количество коробок, которое следует открыть для того чтобы собрать недостающие k разных купонов. С вероятностью $(n-k) / n$ купон в следующей коробке будет бесполезным, а с вероятностью k / n он будет того типа, которого у Вас еще нет. Имеем соотношение:

$$a_k = (1 + a_k) * \frac{n-k}{n} + (1 + a_{k-1}) * \frac{k}{n},$$
$$a_0 = 0$$

Раскроем скобки и выразим a_k через a_{k-1}:

$$a_k = \frac{n}{k} + a_{k-1}$$

Рекуррентность можно расписать в виде:

$$a_k = \frac{n}{k} + a_{k-1} = \frac{n}{k} + \frac{n}{k-1} + a_{k-2} = \frac{n}{k} + \frac{n}{k-1} + \frac{n}{k-2} + a_{k-3} = \ldots =$$
$$= \frac{n}{k} + \frac{n}{k-1} + \frac{n}{k-2} + \ldots + \frac{n}{1} + a_0 = n * \sum_{i=1}^{k} \frac{1}{i}$$

Ответом задачи будет значение a_n – ожидаемое количество коробок, которое следует открыть для того чтобы собрать недостающие n разных купонов:

$$a_n = n * \sum_{i=1}^{n} \frac{1}{i}$$

Для вывода результата в требуемом формате остается реализовать суммирование при помощи рациональных чисел. Для сокращения дробей будем использовать функцию gcd, вычисляющую наибольший общий делитель.

Пример. Вычислим значения a_2 и a_3.

$$a_2 = 2 * (1 + \frac{1}{2}) = 3,$$
$$a_3 = 3 * (1 + \frac{1}{2} + \frac{1}{3}) = 3 + \frac{3}{2} + 1 = 5\frac{1}{2}.$$

1573. Коровы и машины

Для решения задачи следует записать формулу условной вероятности.

Число дверей равно $doors = cows + cars$. Вероятность сначала угадать корову составляет $cows / doors$, машину – $cars / doors$. После смены мнения и открытия дверей с $shown$ коровами игрок может выбирать приз среди $doors - shown - 1$ закрытых дверей. Пусть вначале игрок указал на корову. Тогда среди закрытых дверей имеется $cars$ машин, и вероятность ее выигрыша составляет

$$cars / (doors - shown - 1)$$

Если вначале игрок указал на машину, то среди закрытых дверей осталось $cars - 1$ машин и вероятность ее выигрыша составляет

$$(cars - 1) / (doors - shown - 1)$$

Результирующая вероятность выиграть машину равна

$$(cows / doors) * (cars / (doors - shown - 1)) + (cars / doors) * ((cars - 1) / (doors - shown - 1))$$
$$= (cows * cars + cars * (cars - 1)) / (doors * (doors - shown - 1)).$$

Пример. Рассмотрим первый тест. В игре имеются 3 двери. Если сначала игрок укажет на дверь с призом (вероятность 1/3), то после смены мнения он проиграет. Если сначала будет выбрана дверь с коровой (вероятность 2/3), то после открытия двери с коровой и смены мнения игрок непременно попадет на дверь с призом и выиграет. Таким образом, вероятность выигрыша составит 1/3 * 0 + 2/3 * 1 = 2/3.

1574. Бросание кубиков

Учитывая ограничения на n и x, вычислим в ячейках массива res[i][j] вероятность того, что при бросании в точности i кубиков выпадет сумма j. Очевидно, что при бросании одного кубика вероятность выпадения любого значения от 1 до 6 одинакова и равна 1/6:

$$res[1][i] = \begin{cases} 1/6, 1 \le i \le 6 \\ 0, \text{иначе} \end{cases}$$

При бросании i кубиков может выпасть сумма j, если при бросании первых $i - 1$ кубиков выпадет сумма $j - k$, а при бросании i - ого кубика сумма k, $k = 1, 2, \ldots, 6$. Таким образом

$$res[i][j] = res[i-1][j-6] * res[1][6] +$$
$$+ res[i-1][j-5] * res[1][5] + \ldots + res[i-1][j-1] * res[1][1] =$$
$$= \frac{1}{6}\sum_{k=1}^{6} res[i-1][j-k],$$

учитывая что res[1][6] = res[1][5] = … = res[1][1] = 1/6.

Вероятность того, что сумма очков при бросании n кубиков будет как минимум x, равна

$$s = \sum_{i=x}^{6n} res[n][i]$$

Пример. Рассмотрим пример заполнения массива res для одного, двух и трех кубиков. В пустых клетках вероятность равна 0.

i/*j*	1	2	3	4	5	6	7	8	9	10	11	12
1	1/6	1/6	1/6	1/6	1/6	1/6						
2		1/36	2/36	3/36	4/36	5/36	6/36	5/36	4/36	3/36	2/36	1/36
3			1/216	3/216	6/216	10/216	15/216	21/216	25/216	27/216	27/216	25/216

Обозначим через P($n = i$, $s = j$) вероятность того, что при бросании n кубиков выпадет сумма j. Рассмотрим вычисление значений некоторых ячеек.

а) На двух кубиках сумма 5 может получиться при следующих исходах: 1 + 4, 2 + 3, 3 + 2, 4 + 1. То есть

$$P(n=2, s=5) = P(n=1, s=1) * P(n=1, s=4) + P(n=1, s=2) * P(n=1, s=3) +$$
$$+ P(n=1, s=3) * P(n=1, s=2) + P(n=1, s=4) * P(n=1, s=1) =$$
$$= 1/6 * 1/6 + 1/6 * 1/6 + 1/6 * 1/6 + 1/6 * 1/6 = 4/36$$

б) На трех кубиках может получиться сумма 10, если на первых двух выпало 4, на третьем – 6, или на первых двух 5, на третьем 5 и так далее. При этом значения вероятностей для двух кубиков P($n = 2$, $s = i$) уже вычислены.

$$P(n=3, s=10) = P(n=2, s=4) * P(n=1, s=6) + P(n=2, s=5) * P(n=1, s=5) +$$
$$+ P(n=2, s=6) * P(n=1, s=4) + P(n=2, s=7) * P(n=1, s=3) +$$
$$+ P(n=2, s=8) * P(n=1, s=2) + P(n=2, s=9) * P(n=1, s=1) =$$
$$= 3/36 * 1/6 + 4/36 * 1/6 + 5/36 * 1/6 + 6/36 * 1/6 + 5/36 * 1/6 + 5/36 * 1/6 =$$
$$= 3/216 + 4/216 + 5/216 + 6/216 + 5/216 + 4/216 = 27/216$$

1575. Боже! Спаси меня

Обозначим через P ожидаемое время, через которое можно выбраться из комнаты в безопасное место. Тогда оно равно

$$P = \sum_{i=1}^{n} p_i \cdot t_i ,$$

где t_i – время, через которое можно попасть в безопасное место, если пойти в i – ую дверь. Очевидно, что $t_i = x_i$, если $x_i > 0$. Если $x_i < 0$, то для того чтобы выбраться, следует потратить время $-x_i$ чтобы снова оказаться в комнате, а потом время P, через которое можно выбраться. То есть в этом случае $t_i = -x_i + P$. Таким образом, получаем линейное уравнение относительно P. Построить и решить уравнение можно за время $O(n)$, где n – количество дверей.

Пример. Рассмотрим входные данные для первого теста. Составим по ним уравнение:

$$P = 0.33 * 2 + 0.33 * (3 + P) + 0.34 * (5 + P),$$

откуда 0.33 * P = 3.35 или P = 10.15.

1576. Потерянный подарок

Пусть Оли подарил Стэну bb черных шариков (из которых b он потом нашел). Тогда из фразы Оли, сказанной при вручении подарка следует, что

$$\frac{r}{r+bb} \cdot \frac{r-1}{r+bb-1} + \frac{bb}{r+bb} \cdot \frac{bb-1}{r+bb-1} = \frac{1}{2}$$

Упрощая выражение, получим квадратное уравнение:

$$r^2 - (1+2bb)r + bb^2 - bb = 0$$

Поскольку r известно, решаем его относительно bb. Для этого перепишем уравнение в виде

$$bb^2 - (1+2r)bb + r^2 - r = 0$$

Дискриминант равен $D = (1+2r)^2 - 4(r^2 - r) = 1 + 8r$.

Корни уравнения: $bb_{1,2} = \frac{1 + 2r \pm \sqrt{1+8r}}{2}$.

Отсюда заключаем, что:

- Если $1 + 8r$ не является полным квадратом, то количество красных шариков не корректно.
- Если оба корня bb_1 и bb_2 не целые, то количество красных шариков не корректно.
- Для того чтобы количество черных шариков было корректным, необходимо чтобы корень bb удовлетворял двум соотношениям:
  1. $b \le bb$ (количество найденных черных шариков b должно быть не больше количества подаренных bb).
  2. $0.7bb \le b$ (количество найденных черных шариков b не меньше 70% их общего количества).

Если оба значения r и b являются корректными, то следует вывести только неотрицательные найденные значения bb.

1577. Так Вы хотите стать 2^n-эром?

Пусть f(n, a) – максимально возможное значение выигрыша, если игроку будет задано n вопросов, а начальная сумма равна a. Если $n = 0$, то игрок остается с начальной суммой, то есть f(0, a) = a. Вероятность правильного ответа равна p, $t \le p \le 1$. Если на первый вопрос игрок отвечает правильно, то дальше ему остается ответить на $n - 1$ вопросов имея призовую сумму $2a$. С вероятностью $1 - p$ дается неверный ответ, и деньги пропадают. То есть ожидаемая сумма выигрыша после первого ответа станет равной p * f($n - 1$, $2a$) + $(1 - p)$ * 0 = p * f($n - 1$, $2a$). Если это значение больше предыдущей суммы a, то стоит отвечать на вопрос, иначе следует остановить игру. Ожидаемый выигрыш после ответа на вопрос составит max(a, p * f($n - 1$, $2a$)). Поскольку вероятность p равномерно распределена на отрезке [t .. 1], то

$$f(n, a) = \frac{1}{1-t}\int_{t}^{1}\max(a, p \cdot f(n-1, 2a))\,dp$$

Если t = 1, то вероятность дать правильный ответ равна 1 и в таком случае следует отвечать на все n вопросов, получив при этом выигрыш 2^n.

Пример. Рассмотрим третий тест, $n = 2$, $t = 0.6$. Начальный капитал $a = 1$.

$$f(2, 1) = \frac{1}{0.4}\int_{0.6}^{1}\max(1, p \cdot f(1, 2))\,dp,\ f(1, 2) = \frac{1}{0.4}\int_{0.6}^{1}\max(2, p \cdot f(0, 4))\,dp,\ f(0, 4) = 4$$

Вычислим значение f(1, 2) через f(0, 4):

$$f(1, 2) = \frac{1}{0.4}\int_{0.6}^{1}\max(2, p \cdot f(0, 4))\,dp = 2.5\int_{0.6}^{1}\max(2, 4p)\,dp =$$

/ учитываем, что $4p > 2$ при $0.6 \le p \le 1$ /

$$2.5\int_{0.6}^{1}4p\,dp = 2.5 \cdot 2p^2\Big|_{0.6}^{1} =$$

$$5 * (1 - 0.36) = 5 * 0.64 = 3.2$$

Вычислим значение f(2, 1) через f(1, 2):

$$f(2, 1) = \frac{1}{0.4}\int_{0.6}^{1}\max(1, p \cdot f(1, 2))\,dp = 2.5\int_{0.6}^{1}\max(1, 3.2p)\,dp =$$

/ учитываем, что $3.2p > 1$ при $0.6 \le p \le 1$ /

$$2.5\int_{0.6}^{1}3.2p\,dp = 2.5 \cdot 1.6p^2\Big|_{0.6}^{1} =$$

$$4 * (1 - 0.36) = 4 * 0.64 = 2.56$$

1578. Заданная вероятность

Вычислим вероятность P того, что в точности r друзей совершат покупки. Для этого переберем все подмножества из r друзей (их не более C_{20}^{10}). Одновременно для каждого друга вычисляем вероятность $prob_i$ того, что он совершит покупку при условии, что в точности r друзей купили что-нибудь. Таким образом, вероятность покупательской способности i - ого друга при условии совершении покупок в точности r друзьями, составит $prob_i$ / P.

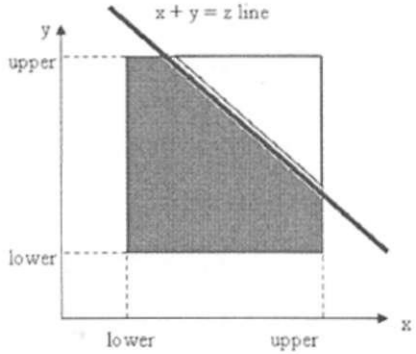
Пример. Рассмотрим первый тест. Двум купившим друзьям из трех будут соответствовать последовательности 011, 101, 110. Требуемые вычисления приведены в таблице:

последоват.	подмножество	вероятность	prob[1]	prob[2]	prob[3]
011	{2, 3}	0.9 * 0.2 * 0.3 = 0.054	-	0.054	0.054
101	{1, 3}	0.1 * 0.8 * 0.3 = 0.024	0.024	-	0.024
110	{1, 2}	0.1 * 0.2 * 0.7 = 0.014	0.014	0.014	-
сумма вероятностей P		0.092	0.038	0.068	0.078
prob[i] / P			0.413043	0.739130	0.847826

1579. Тройной прыжок

Отнимем величину первого прыжка от всех результатов прыжков соперников, таким образом сведя задачу к "двойному" прыжку. Из условия задачи следует, что длина каждого прыжка является независимой случайной величиной, равномерно распределенной на отрезке [$lower$, $upper$]. Сумма двух прыжков также является случайной величиной, определенной на отрезке [2*$lower$, 2*$upper$]. Но она уже не будет равномерно распределенной.

Определим случайную величину F(z) = {сумма двух прыжков не более z}. F(z) пропорционально площади области квадрата под прямой $z = x + y$, изображенной красным цветом.



Функция распределения F(z) имеет вид:

$$F(z) = \begin{cases} 0, z \le 2 \cdot lower \\ \dfrac{(z - 2 \cdot lower)^2}{2 \cdot (upper - lower)^2}, 2 \cdot lower < z \le upper + lower \\ 1 - \dfrac{(z - 2 \cdot upper)^2}{2 \cdot (upper - lower)^2}, upper + lower < z \le 2 \cdot upper \\ 1, z > 2 \cdot upper \end{cases}$$

Отсортируем длины прыжков соперников по убыванию. Вероятность занять первое место равна 1 – F(opponents[0]). Вероятность занять второе место равна F(opponents[0]) – F(opponents[1]) и так далее. Вероятность занять последнее место равна F(opponents[$n - 1$]), где n – количество соперников.

1580. Простой футбол

Обозначим вероятность того, что первая команда забьет количество голов, являющееся простым числом, через *pa*. Для второй команды эту вероятность обозначим через *pb*. Поскольку события забивания голов первой и второй командой независимы, то искомая вероятность равна

$$1 - (1 - pa) * (1 - pb) = pa + pb - pa * pb$$

Весь матч состоит из 18 пятиминуток, поэтому забито каждой командой может не более 18 голов. Для каждого простого значения p, меньшего 18, следует вычислить вероятность того, что каждая их команд забьет в точности p мячей. Просуммировав эти вероятности для каждой команды, получим значения *pa* и *pb*.

k голов команда может забить, забивая в некоторые из k указанных пятиминуток по одному голу, а в остальные пятиминутки не забивая ни одного. Пусть s – вероятность забить гол командой в одну пятиминутку. Тогда вероятность того, что команда забьет мячи в некоторых конкретных k пятиминутках, равна $s^k * (1 - s)^{18 - k}$. Всего разных k пятиминуток существует $C_{18}^k = \frac{18!}{k!(18-k)!}$. Поэтому вероятность того, что команда забьет ровно k голов, равна $\frac{18!}{k!(18-k)!} s^k (1-s)^{18-k}$. Искомая вероятность *pa* равна

$$\sum_{k=2,3,5,7,11,13,17} \frac{18!}{k!(18-k)!} skillOfTeamA^k (1 - skillOfTeamA)^{18-k}$$

Аналогично вычисляется вероятность *pb*.

1581. Игра с кубиком

Пусть r[i] содержит количество бросков, необходимое для получения в точности i конфет. Очевидно, что r[0] = 0, а также r[i] = 0 при $i < 0$.

Для $i > 0$ имеем соотношение:

$$r[i] = 1 + \frac{1}{6}(r[i-1] + r[i-2] + r[i-3] + r[i-4] + r[i-5] + r[i-6])$$

На кубике может выпасть число от 1 до 6. Для получения в точности i конфет можно, например, получить $i - k$ $(1 \le k \le 6)$ конфет, после чего бросить кубик и с вероятностью 1 / 6 получить еще k конфет.

Последовательно вычисляем значения r[i] для i от 1 до *candies* и возвращаем r[*candies*].

1582. Стрельба из лазера

Математическое ожидание суммы случайных величин равно сумме математических ожиданий этих величин. Поэтому достаточно вычислить для каждой цели вероятность ее поражения и просуммировать полученные значения.

Обозначим через a_1 и a_2 углы, при которых поражаются концы i - ой цели. Поскольку значения элементов массива x положительны, то $a_1 = \text{arctg}(y1_i / x_i)$, $a_2 = \text{arctg}(y2_i / x_i)$. i - ая цель поражается, если угол выстрела лежит между $\min(a_1, a_2)$ и $\max(a_1, a_2)$. Вероятность попадания в i - ую цель равна вероятности того, что точка, выбранная из отрезка $[-\pi / 2, \pi / 2]$, попадет в отрезок $[\min(a_1, a_2)$ и $\max(a_1, a_2)]$. Она равна $(\max(a_1, a_2) - \min(a_1, a_2)) / \pi$.

1583. Тир

Вероятность того, что друг не попадет в мишень с одного выстрела, равна 1 – *accuracy* / 100.0. Вероятность того, что друг не попадет в мишень сделав n выстрелов, равна

$$(1 - accuracy / 100.0)^n$$

Ответом задачи будет такое наименьшее n, для которого указанная выше вероятность станет не больше 0.5.

1584. Произвольное тасование

Занесем в массив cur последовательность $\{1, 2, \ldots, n\}$. Будем моделировать все возможные процессы тасования, выбирая в качестве значения r все значения от 1 до n. Процесс тасования состоит из n итераций, на каждой из которых в качестве r можно выбрать любое из n чисел (от 1 до n). Таким образом, из последовательности $\{1, 2, \ldots, n\}$ в результате тасования можно получить n^n других последовательностей, некоторые из которых возможно будут одинаковыми ($n^n > n!$). В переменной s подсчитаем, сколько раз в процессе моделирования из $\{1, 2, \ldots, n\}$ получится последовательность, заданная в массиве outputArray. Разделив найденное число s на n^n, получим искомую вероятность.

Функция shuffle имеет единственный параметр *pos* – номер проводимой итерации. Для прохождения по времени следует совершить следующее отсечение. Пусть производится *pos* итерация, а текущий массив cur отличается от исходного m в c позициях. Тогда если $c > 2*(n - pos)$, то очевидно, что за оставшиеся $n - pos$ обменов невозможно из cur получить m. Это следует из того, что за каждый из оставшихся обменов мы можем переставлять максимум 2 элемента.

1585. Игра в лотерею

Рассмотрим случай, когда надо угадать 5 номеров из 39. Очевидно, что вероятность выигрыша составит $1 / C_{39}^5$. Вычислим вероятность того, что угадано будет в точности 4 номера. В этом случае из 5 номеров, задуманных игроком, 4 номера должны находиться среди 5 выпавших (C_5^4 вариантов), а один номер должен находиться среди 39 – 5 = 34 не выпавших (C_{34}^1 вариантов).

В общем случае, для угадывания в точности x номеров, необходимо чтобы эти x номеров находились среди m выпавших, а остальные загаданные $m - x$ номеров игрока находились среди $n - m$ невыпавших. Вероятность угадывания в точности x номеров равна

$$\frac{C_m^x \cdot C_{n-m}^{m-x}}{C_n^m}$$

Для нахождения искомой вероятности следует просуммировать вероятности того, что будет угадано в точности x номеров для x от k до m включительно.

РЕАЛИЗАЦИЯ ЗАДАЧ

1316. Одинаковые шары

Объявим необходимые массивы. В ячейке ans[i] будем вычислять значение $C_{i-2}^{\frac{i}{2}-1} p^{\frac{i}{2}-1} q^{\frac{i}{2}-1}$.

```
#define MAX 100001
double lf[MAX], ans[MAX];
```

Основная часть программы. Занесем в lf[i] значение $\ln i!$.

```
res = lf[1] = 0.0;
for(res = 0, i = 2; i < MAX; i++)
{
  res += log((double)i);
  lf[i] = res;
}

for(i = 2; i < MAX; i += 2)
{
  res = lf[i/2-1];
  res = lf[i-2] - (i-2)*log(2.0) - res - res;
```

Переменнмая *res* содержит $\ln C_{i-2}^{\frac{i}{2}-1} p^{\frac{i}{2}-1} q^{\frac{i}{2}-1}$. Заносим в ans[$i$] значение $C_{i-2}^{\frac{i}{2}-1} p^{\frac{i}{2}-1} q^{\frac{i}{2}-1}$.

```
  ans[i] = exp(res);
}
```

Для каждого входного значения n выводим ответ.

```
while(scanf("%d",&n) == 1)
  printf("%.8lf\n",1 - ans[n]);
```

1317. Дни рождения

В ячейках p[i] будем запоминать значения вероятностей p(i).

```
double p[401];
```

Положим p[1] = 1. Остальные значения ячеек p[i] вычислим по формуле, указанной в анализе задачи.

```
p[1] = 1.0;
for(i = 2; i < 401; i++)
  p[i] = p[i-1] * (1.0 - (i - 1.0) / 365);
```

Для каждого входного значения n выведем ответ.

```
while(scanf("%d",&n) == 1)
  printf("%.8lf%%\n",(1 - p[n]) * 100);
```

1318. Толстая монета

Читаем входные данные, вычисляем искомую толщину монеты и выводим ее.

```
while(scanf("%lf %lf",&r,&n) == 2)
{
  res = r / sqrt(n*n-1)*2;
  printf("%.6lf\n",res);
}
```

1319. Дуэлянты

Читаем входные данные, вычисляем искомую вероятность поединка и выводим ее.

```
while(scanf("%lf",&n) == 1)
{
  res = 1 - (1.0-n/60.0)*(1.0-n/60.0);
  printf("%.6lf\n",res);
}
```

1571. Какая вероятность?

Читаем входные данные.

```
  scanf("%d",&s);
  while(s--)
  {
    scanf("%d %lf %d",&n,&p,&i);
```

Если вероятность p равна 0, то выводим 0.

```
    if (p < 1e-7) printf("0.0000\n");
    else
    {
```

Иначе вычисляем искомую вероятность при помощи приведенной выше формулы. Результат выводим с 4 знаками после десятичной точки.

```
      res = p*pow(1-p,i-1)/(1-pow(1-p,n));
      printf("%.4lf\n",res);
    }
  }
```

1572. Купоны

В структуре RatNumber храним рациональное число: числитель x и знаменатель y.

```
struct RatNumber
{
  long long x,y;
} c, temp;
```

Сложение рациональных чисел *a* и *b*. Возвращаемая сумма является несократимой дробью. Функция gcd вычисляет наибольший общий делитель.

```
struct RatNumber add(struct RatNumber a,struct RatNumber b)
{
  struct RatNumber res;
  res.x = a.x*b.y + a.y*b.x;
  res.y = a.y * b.y;
  d = gcd(res.x,res.y);
  if (d)
  {
    res.x  /= d;res.y  /= d;
  }
  return res;
}
```

Основной цикл программы. Читаем входное значение *n*.

```
while(scanf("%d",&n) == 1)
{
  c.x = 0; c.y = 1; temp.x = 1;
```

Вычисление суммы $c = n * \sum_{i=1}^{n} \frac{1}{i}$

```
  for(i = 1; i <= n; i++)
  {
    temp.y = i;
    c = add(c,temp);
  }
  c.x = n * c.x;
  d = gcd(c.x,c.y);
  c.x  /= d; c.y  /= d;
  d = c.x / c.y;
  c.x -= d * c.y;
```

Переменная *d* содержит целую часть результата *c*. Если знаменатель результата равен 1, то выводим только числитель. Иначе форматируем вывод ответа, как показано в условии задачи. Функция digits находит количество знаков числа *x*.

```
  if (c.y > 1)
  {
    for(i = 0; i <= digits(d); i++) printf(" ");
    printf("%lld\n",c.x);
    printf("%lld ",d);
    for(i = 0; i < digits(c.y); i++) printf("-"); printf("\n");
    for(i = 0; i <= digits(d); i++) printf(" ");
    printf("%lld\n",c.y);
  }
  else printf("%lld\n",d);
}
```

1573. Коровы и машины

Читаем входные данные, вычисляем число дверей *doors*, находим ответ по выше приведенной формуле и выводим его. Поскольку входные данные целые, то чтобы избежать потери точности при делении, следует делимое перевести в действительный тип, умножив его на 1.0.

```
while(scanf("%d %d %d",&cows,&cars,&shown) == 3)
{
  doors = cows + cars;
  res = 1.0*((cars-1)*cars + cars*cows)/(doors*(doors-shown-1));
  printf("%.5lf\n",res);
}
```

1574. Бросание кубиков

Все вычисления будем проводить в 64 - битовом беззнаковом целочисленном типе unsigned long long. Поскольку вычисления следует производить с дробями, определим тип рационального числа в структуре RatNumber. Определим рабочий массив res и дополнительные переменные.

```
struct RatNumber
{
  unsigned long long x, y;
} one6, temp, res[25][150], s;
```

Для работы программы нам понадобятся функции вычисления наибольшего общего делителя (gcd) и наименьшего общего кратного (lcm).

```
unsigned long long gcd(unsigned long long a, unsigned long long b)
{
  return (!b) ? a : gcd(b,a % b);
}

unsigned long long lcm(unsigned long long a, unsigned long long b)
{
  return a / gcd(a,b) * b;
}
```

Функция RatNumber складывает два рациональных числа по формуле

$$\frac{a.x}{a.y}+\frac{b.x}{b.y}=\frac{HOK(a.y,b.y)/a.y\cdot a.x+HOK(a.y,b.y)/b.y\cdot b.x}{HOK(a.y,b.y)}$$

Вычисляя наименьшее общее кратное знаменателей, можно избежать переполнения. Далее находим наибольший общий делитель числителя и знаменателя результата и сокращаем полученную сумму.

```
struct RatNumber add(struct RatNumber a,struct RatNumber b)
{
  struct RatNumber res;
  res.y = lcm(a.y,b.y);
  res.x = res.y/a.y*a.x + res.y/b.y*b.x;
  d = gcd(res.x,res.y);
  if (d)
  {
    res.x  /= d;res.y  /= d;
  }
  return res;
}
```

Основная часть программы. Установим значения рациональных чисел res[*i*][*j*] равными 0, интерпретируя 0 как 0/1.

```
int main(void)
{
  for(i = 0; i < 25; i++)
  for(j = 0; j < 150; j++)
    res[i][j].x = 0, res[i][j].y = 1;
```

Определим рациональное число one6 = 1/6 и инициализируем им значения res[1][*i*], *i* = 1, 2, …, 6.

```
  one6.x = 1; one6.y = 6;
  for(i = 1; i <= 6; i++) res[1][i] = one6;
```

Вычисляем значения res[*i*][*j*] по выше приведенной рекуррентной формуле.

```
  for(n = 2; n < 25; n++)
  for(i = n - 1; i <= 6 * (n - 1); i++)
  for(j = 1; j <= 6; j++)
  {
    temp = res[n-1][i];
    temp.y = temp.y*6;
    res[n][i+j] = add(res[n][i+j],temp);
  }
```

Для каждой пары входных значений *n* и *x* находим вероятность того, что сумма очков на всех кубиках будет как минимум *x*. Для этого вычисляем сумму

$$s = \sum_{i=x}^{6n} res[n][i]$$

```
  while(scanf("%d %d",&n,&x),n+x)
  {
    s.x = 0, s.y = 1;
    for(i = x; i <= 6 * n; i++)
      s = add(s,res[n][i]);
```

Выводим искомую вероятность в виде дроби. Если результат равен 0, то выводим один 0. Если вероятность равна 1, то выводим 1.

```
    if (s.x == 0) printf("0\n"); else
    if (s.y == 1) printf("%lld\n",s.x); else
    printf("%lld/%lld\n",s.x,s.y);
  }
  return 0;
}
```

1575. Боже! Спаси меня

Линейное уравнение будем решать следующим образом. Все слагаемые, в которых встречается множитель P, будем переносить влево, а коэффициент при нем хранить в переменной *left*. Все слагаемые-константы будем собирать справа, и хранить в переменной *right*. Изначально *left* = 1, *right* = 0. Читаем *n* пар чисел *x* и *p*. Если *x* > 0, то увеличиваем правую сторону на величину *x* * *p*. Если *x* < 0, то в правой части уравнения появится слагаемое *p* * (-*x* + P). И тогда следует прибавить к правой части величину *p* * (-*x*), а из левой вычесть *p*.

```
left = 1.0; right = 0.0;
scanf("%d",&n);
for(j = 0; j < n; j++)
{
  scanf("%lf %lf",&x, &p);
  if (x < 0.0)
  {
    left -= p;
    right += p * (-x);
  } else
      right += x * p;
}
```

Имеем уравнение *left* * P = *right*. Если *left* = 0, то выбраться из комнаты невозможно (не существует двери, ведущей в безопасное место). Иначе искомое время равно P = *right* / *left*.

```
if (left > 0.0)
{
  res = right / left;
  printf("Case %d: %.2lf\n",i,res);
} else
    printf("Case %d: God! Save me\n",i);
```

Здесь переменной *i* соответствует номер теста.

1576. Потерянный подарок

Пусть *bres* – один из удвоенных корней уравнения $bb^2 - (1+2r)bb + r^2 - r = 0$. Если он не делится нацело на 2, то количество красных шариков некорректно, в таком случае возвращаем -1. В случае некорректности значения *b* возвращаем -2.

```
int check(int bres)
{
  if (bres % 2) return -1; bres /= 2;
  if (!((0.7 * bres <= b) && ((b <= bres)))) return -2;
  return bres - b;
}
```

Основная часть программы.

```
while(scanf("%d %d",&r,&b),r + b)
{
```

Вычисляем дискриминант *d* квадратного уравнения. Если он не является полным квадратом, то количество красных шариков не корректно.

```
  d = 1 + 8 * r; dd = sqrt(1.0 * d + 1e-7);
  if (dd * dd != d) printf("No. of red balls invalid\n");
  else
  {
```

Вычисляем корни квадратного уравнения *res1* и *res2*. Выводим результат в зависимости от их значения.

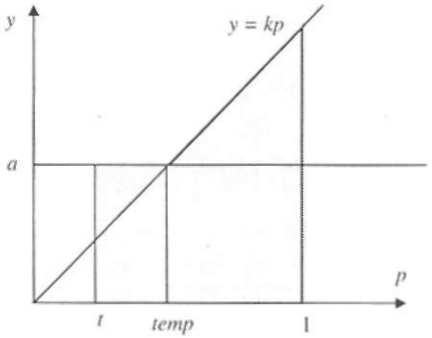
```
    res1 = check(1 + 2*r - dd);
    res2 = check(1 + 2*r + dd);
    if ((res1 == -1) && (res2 == -1)) printf("No. of red balls invalid\n");
    else if ((res1 == -2) && (res2 == -2))
      printf("No. of black balls invalid\n");
```

```
    else
    {
      if ((res1 >= 0) && (res2 >= 0)) printf("%d %d\n",res1,res2); else
      if (res1 >= 0) printf("%d\n",res1); else printf("%d\n",res2);
    }
  }
}
```

1577. Так Вы хотите стать 2^n-эром?

Функция integral вычисляет значение интеграла $I(a, k) = \frac{1}{1-t}\int_t^1 \max(a, kp)\,dp$ для заданных действительных чисел a и k. При $t = 1$ вероятность угадывания p равна единице, значение интеграла $I(a, k)$ полагается равным $\max(a, k)$.

Ниже приведена область, площадь которой равна значению интеграла $\int_t^1 \max(a, kp)\,dp$:



Найдем точку пересечения прямых $y = a$ и $y = kp$: $a = kp$, откуда $p = a/k$. Положим $temp = a / k$. Значение интеграла $\int_t^1 \max(a, kp)\,dp$ рассмотрим как сумму $\int_t^{temp} a\,dp + \int_{temp}^1 kp\,dp$. В зависимости от положения точки $temp$ относительно точек t и 1 вычисляем значение интеграла $I(a, k)$.

Если $t \le temp \le 1$, то $\int_t^{temp} a\,dp + \int_{temp}^1 kp\,dp = a * (temp - t) + k * (1 - temp * temp) / 2$.

Если $temp \le t$, то $\int_t^{temp} a\,dp + \int_{temp}^1 kp\,dp = \int_t^1 kp\,dp = k * (1 - t * t) / 2$.

Если $temp > 1$, то $\int_t^{temp} a\,dp + \int_{temp}^1 kp\,dp = \int_t^1 a\,dp = a * (1 - t)$.

```
double integral(double a, double k)
{
  double s = 0, temp = a / k;
  if (t == 1) return max(a,k);
  if (temp > 1) return a * (1 - t);
  if (temp >= t) s = a * (temp - t);
  else temp = t;
  s += k * (1 - temp * temp) / 2;
  return s / (1 - t);
}
```

Рекурсивное вычисление $f(n, a) = \begin{cases} I(a, f(n-1, 2a)), & n > 0 \\ a, & n = 0 \end{cases}$.

```
double f(int n, int a)
{
  if (!n) return a;
  double k = f(n-1,2*a);
  return integral(a,k);
}
```

Основной цикл программы. Читаем входные данные и выводим значение $f(n, 1)$.

```
while(scanf("%d %lf",&n,&t),n + t)
  printf("%.3lf\n",f(n,1));
```

1578. Заданная вероятность

Исходные вероятности покупки друзей p_i храним в массиве p. В переменной *all* вычисляем вероятность того, что в точности r друзей из n совершат покупки. Вероятность покупки каждого друга при условии, что в точности r друзей купили что-нибудь, будем хранить в ячейках массива prob.

```
double p[21], prob[21], all;
int m[21];
```

Функция generate генерирует последовательности длины n, состоящие из $n - r$ нулей и r единиц. Каждой такой последовательности соответствует подмножество из r друзей. Для каждого подмножества друзей вычисляем вероятность их покупок и добавляем к *all*. Если в это подмножество входит i - ый друг, то вероятность прибавим и к prob[i].

```
void generate(int pos, int r)
{
  double p1;
  int i;
  if ((pos + r > n) || (r < 0)) return;
  if (!r && (pos == n))
  {
     for(p1 = 1.0, i = 0; i < n; i++)
       if (m[i]) p1 *= p[i]; else p1 *= (1 - p[i]);
     all += p1;
     for(i = 0; i < n; i++)
       if (m[i]) prob[i] += p1;
     return;
  }
  generate(pos+1,r);
  m[pos] = 1;
  generate(pos+1,r-1);
  m[pos] = 0;
}
```

Основной цикл программы. Читаем входные данные. Заданные вероятности сохраняем в массиве p.

```
while(scanf("%d %d",&n,&r), n + r)
{
  for(all = i = 0; i < n; i++) scanf("%lf",&p[i]);
  memset(m,0,sizeof(m)); memset(prob,0,sizeof(prob));
```

Генерируем последовательности длины n из r единиц.

```
  generate(0,r);
```

Для каждого друга выводим искомую вероятность.

```
  printf("Case %d:\n",cs++);
  for(i = 0; i < n; i++)
   printf("%.6lf\n",prob[i]/all);
}
```

1579. Тройной прыжок

Реализация функции распределения F(z).

```
double f(double lower, double upper, double z)
{
  if (z <= 2 * lower) return 0.0;
  if (z <= lower + upper) return (z - 2 * lower) * (z - 2*lower) / 2 /
                                 (upper-lower) / (upper - lower);
  if (z <= 2 * upper) return 1 - (z - 2 * upper) * (z - 2 * upper) / 2 /
                                 (upper-lower) / (upper - lower);
  return 1.0;
}
```

Искомые вероятности заносим в массив res. Отдельно вычисляем значения res[0] и res[n].

```
void getProbabilities(int lower, int upper, int first)
{
   int i;
   for(i = 0; i < n; i++) opponents[i] -= first;
   sort(opponents,opponents+n,greater<int>());
   res[0] = 1 - f(lower,upper,opponents[0]);
   res[n] = f(lower,upper,opponents[n-1]);
   for(i = 1; i < n; i++)
     res[i] = f(lower,upper,opponents[i-1]) - f(lower,upper,opponents[i]);
}
```

Основная часть программы.

```
while(scanf("%d %d %d %d",&lower,&upper,&first,&n) == 4)
{
  for(i = 0; i < n; i++) scanf("%d",&opponents[i]);
  getProbabilities(lower, upper, first);
  for(i = 0; i < n + 1; i++) printf("%.4lf ",res[i]); printf("\n");
}
```

1580. Простой футбол

Занесем в массив primes все простые числа, не больше количества пятиминуток в матче.

```
int primes[7] = {2, 3, 5, 7, 11, 13, 17};
```

Вычисление искомой вероятности. Функция fact(n) находит факториал числа n.

```
double getProbability(int skillOfTeamA, int skillOfTeamB)
{
  double pa, pb, sa = skillOfTeamA / 100.0, sb = skillOfTeamB / 100.0,
                 f18 = fact(18);
  for(pa = pb = i = 0; i < 7; i++)
  {
    pa += f18 / fact(primes[i]) / fact(18 - primes[i]) * pow(sa, primes[i]) *
          pow(1 - sa, 18 - primes[i]);
    pb += f18 / fact(primes[i]) / fact(18 - primes[i]) *
          pow(1.0 * sb, primes[i]) * pow(1.0 - sb, 18 - primes[i]);
  }
  return pa + pb - pa * pb;
}
```

Основная часть программы.

```
while(scanf("%d %d",&skillOfTeamA,&skillOfTeamB) == 2)
{
  res = getProbability(skillOfTeamA, skillOfTeamB);
  printf("%.4lf\n",res);
}
```

1581. Игра с кубиком

Объявим глобальный массив r.

```
double r[1000001];
```

Функция expectedThrows вычисляет элементы массива r[i] согласно выше приведенной формуле.

```
double expectedThrows(int candies)
{
  int i, j;
  double s;
  r[0] = 0;
  for(i = 1; i <= candies; i++)
  {
    for(s = 0, j = max(i-6,0); j < i; j++)
      s += r[j];
    r[i] = 1 + s / 6;
  }
  return r[candies];
}
```

Основная часть программы.

```
while(scanf("%d",&n) == 1)
{
  res = expectedThrows(n);
  printf("%.4lf\n",res);
}
```

1582. Стрельба из лазера

Объявим массивы, которые будут содержать координаты концов целей.

```
int x[51], yy1[51], yy2[51];
```

Функция numberOfHits возвращает искомое ожидаемое количество целей, которое может быть поражено одним выстрелом.

```
double numberOfHits(void)
{
  double a1, a2, res = 0.0;
  int i;
  for(i = 0; i < n; i++)
  {
    a1 = atan(1.0 * yy1[i] / x[i]);
    a2 = atan(1.0 * yy2[i] / x[i]);
    res += fabs(a1 - a2) / PI;
  }
  return res;
}
```

Основная часть программы.

```
while(scanf("%d",&n) == 1)
{
  for(i = 0 ;i < n; i++) scanf("%d",&x[i]);
  for(i = 0 ;i < n; i++) scanf("%d",&yy1[i]);
  for(i = 0 ;i < n; i++) scanf("%d",&yy2[i]);
  res = numberOfHits();
  printf("%.4lf\n",res);
}
```

1583. Тир

Функция profitableBetвозвращает максимальное количество выстрелов, при котором Вам есть смысл спорить.

```
int profitableBet(int accuracy)
{
  double miss = 1 - accuracy / 100.0,prob = miss;
  int n = 0;
  while (prob > 0.5) n++,prob *= miss;
  return n;
}
```

Основная часть программы.

```
while(scanf("%d",&accuracy) == 1)
{
  res = profitableBet(accuracy);
  printf("%d\n",res);
}
```

1584. Произвольное тасование

Объявим глобальные массивы.

```
int m[10], cur[10];
```

Реализация функции произвольного тасования.

```
void shuffle(int pos)
{
  int i, c;
  if (pos == n)
  {
    for(i = 0; i < n; i++)
      if (m[i] != cur[i]) return;
    s++;
    return;
  }

  for(c = i = 0; i < n; i++)
    if (m[i] != cur[i]) c++;
  if (c > 2 * (n - pos)) return;

  for(i = 0; i < n; i++)
  {
    swap(cur[i],cur[pos]);
    shuffle(pos + 1);
    swap(cur[i],cur[pos]);
  }
}
```

Основная часть программы.

```
while(scanf("%d",&n) == 1)
{
  for(s = i = 0; i < n; i++)
    scanf("%d",&m[i]), cur[i] = i + 1;
  shuffle(0);
  res = 1.0 * s / pow((double)n,(double)n);
  printf("%.4lf\n",res);
}
```

1585. Игра в лотерею

Функция Cnk(k, n) вычисляет значение C_n^k.

```
int Cnk(int k, int n)
{
  long long res = 1;
  if (k > n) return 0;
  if (k > n - k) k = n - k;
  for(int i = 1; i <= k; i++)
    res = res * (n - i + 1) / i;
  return (int)res;
}
```

Основная часть программы. Вычисляем результат, равный

$$res = \sum_{x=k}^{m} \frac{C_m^x \cdot C_{n-m}^{m-x}}{C_n^m} = \frac{1}{C_n^m} \sum_{x=k}^{m} C_m^x \cdot C_{n-m}^{m-x}$$

```
while(scanf("%d %d %d",&n,&m,&k) == 3)
{
  for(res = 0, x = k; x <= m; x++)
    res += Cnk(x,m) * Cnk(m-x,n-m);
  res = res / Cnk(m,n);
  printf("%.4lf\n",res);
}
```