Студенческий командный чемпионат мира по программированию ACM 2002-2003
Северо-восточный Европейский регион
Восточный (Уральский) подрегион
Первый тур

Во всех задачах ввод данных производится из файла INPUT.TXT, а вывод результатов - в файл OUTPUT.TXT. Формат входного файла соответствует спецификации, дополнительные проверки не нужны. Все строки, в том числе последняя, оканчиваются символом перехода на новую строку.

Время теста указано для процессора Celeron 500 МГц.

Персонажи и некоторые цитаты изъяты из книги С. Лема "Кибериада".

1. Счастливые числа	*5 сек*
2. Вечный Двигатель	*5 сек*
3. Электрувер	*5 сек*
4. Игра	*5 сек*
5. Формула Счастья	*5 сек*
6. Две дороги	*5 сек*
7. Муха	*5 сек*
8. WRITEF	*5 сек*
9. Ловушка	*5 сек*

1. Счастливые числа

Входной файл	*INPUT.TXT*
Выходной файл	*OUTPUT.TXT*
Ограничение времени	*5 сек*

Клапауций любит счастливые числа. Они приносят ему удачу. В рулетку он ставит только на счастливые номера – 7, 22, 00 и другие. Его номер телефона 7634295 также является счастливым числом. Когда Клапауций видит число, не являющееся счастливым, он старается переставить его цифры таким образом, чтобы получилось счастливое число. Его не интересует основание системы счисления, в котором записано число, он превращает в счастливые двоичные, десятичные, шестнадцатеричные и даже 36-ричные числа. Если Клапауцию не удается переставить цифры в числе для получения счастливого числа, он заменяет такое число текстом “ALUCKYNUMBER”.

Клапауций считает счастливыми числа, у которых сумма первых $\lfloor N/2 \rfloor$ цифр равна сумме $\lfloor N/2 \rfloor$ последних цифр, где N – количество цифр в числе. Напишите программу, которая облегчит работу Клапауция по превращению всех чисел в счастливые.

Ввод

Во входном файле в первой строке содержится целое число K ($1 <= K <= 10$) – количество чисел в файле. Далее следует K строк, содержащих по одному числу в строке. Числа состоят из цифр от 0 до 9 и прописных латинских букв от A до Z. Буква A соответствует цифре 10, буква Z – цифре 35. Количество цифр в каждом числе не превышает 100.

Вывод

В выходной файл вывести K строк с результатами для каждого числа входного файла. Если i-ое число входного файла можно сделать счастливым, переставив его цифры, то в i-ой строке выходного файла вывести один из вариантов (любой) такой перестановки. Если не существует счастливой перестановки цифр числа, то в i-ой строке вывести текст “ALUCKYNUMBER”.

Пример INPUT.TXT:

```
3
113
405808AF
123
```

OUTPUT.TXT для примера:

```
131
84580A0F
ALUCKYNUMBER
```

2. Вечный Двигатель

Входной файл	*INPUT.TXT*
Выходной файл	*OUTPUT.TXT*
Ограничение времени	*5 сек*

Однажды конструктор Трурль решил создать Вечный Двигатель.

– Работа этой машины описывается функцией вида $f(x) = A\ x^2 + B\ x + C$, – сказал Трурль Клапауцию. – Если уравнение $f(x) = 0$ имеет действительные решения, то Двигатель будет работать вечно. Сейчас я решу это простое квадратное уравнение и …

– Вздор! Твоя машина не будет работать вечно, так как x каждую секунду изменяется на 1, а результат накапливается, – возразил Клапауций. – Таким образом, нужно рассматривать функцию $g(x, k) = f(x) + f(x+1) + f(x+2) + \ldots + f(x+k)$, не $f(x)$. Как только у уравнения $g(x, k) = 0$ не будет действительных решений, твой Вечный Двигатель остановится!

– Ничего страшного, – ответил Трурль. – Думаю, я смогу подобрать параметры A, B и C таким образом, чтобы машина никогда не останавливалась.

– Ничего у тебя не получится! – саркастически воскликнул Клапауций и отправился домой.

“Ну, если не вечность, то хотя бы миллиард секунд, а там, глядишь, Клапауций и забудет о машине”, – подумал Трурль, принимаясь за расчеты. – “Так, сумма первых k квадратов равна $k(k+1)(2k+1)/6$”…

Напишите программу для Трурля, которая по введенным коэффициентам A, B и C определяет время работы машины, то есть находит такое минимальное целое k>=0, при котором уравнение $g(x, k) = 0$ не имеет действительных решений.

Ввод

Во входном файле в первой строке содержится целое число *N* (1<=*N*<=1000) – количество вариантов, которые Трурль хочет рассчитать. Далее следует *N* строк, в каждой строке содержится три целых числа A, B и C, разделенные одним пробелом (0<A<=10^6, |B|<=10^6, |C|<=10^6).

Вывод

В выходной файл для каждого варианта вывести на отдельной строке ожидаемое время работы машины k. Если k>10^9, то вывести число 1000000000.

Пример INPUT.TXT:

```
2
1 -2 1
1 1 -6
```

OUTPUT.TXT для примера:

```
1
8
```

3. Электрувер

Входной файл	*INPUT.TXT*
Выходной файл	*OUTPUT.TXT*
Ограничение времени	*5 сек*

Случилось как-то Трурлю построить машину для счета, которая оказалась способной к одному-единственному действию, а именно: умножала два на два, да и то при этом ошибалась. С тех пор Клапауций отравлял Трурлю жизнь, и так и эдак его подзуживая, и тогда тот не на шутку разозлился и решил построить машину, которая сочиняла бы стихи. А чтобы еще больше досадить Клапауцию, добавил Трурль подпрограмму, которая вставляла бы в стихи дразнилки про Клапауция в зашифрованном виде.

Когда Электрувер был готов, Трурль позвал на его запуск Клапауция. Тот бросил все свои дела и пошел посмотреть на поражение друга.

Трурль прежде всего включил нагревательные контуры, потом дал малый ток и, когда амплификационные указатели достигли максимума, решительно включил большой рубильник. Почти сразу машина произнесла торжественным, не лишенным обольстительных переливов баритоном:

Купи пай – цалый луг, паку циплай.

Паук пыгал: Цыкай, пулай ли гул.

– По-каковски это? – осведомился Клапауций.

– Черт бы ее побрал! – завопил Трурль и исчез во внутренностях машины. Вскоре оттуда донесся лязг, грохот, треск двоичных разрядов и проклятия конструктора – Трурль догадался, что переборщил с настройками подпрограммы, и анаграмма фразы “Глупый Клапауций” появлялась в стихах чересчур часто.

Напишите программу, которая позволит настроить подпрограмму, подсчитывая количество анаграмм заданной фразы в тексте. Текст А является анаграммой текста Б, если текст А состоит из тех же букв (и в том же количестве), что и текст Б. Знаки препинания, пробелы и переходы на новую строку при этом не учитываются. Например, “ПЛЯС АУКЦИОНА” является анаграммой текста “СОН КЛАПАУЦИЯ”. Количеством анаграмм фразы в тексте будем называть число непрерывных подпоследовательностей в этом тексте, начинающихся с буквы и заканчивающихся буквой и являющихся анаграммой заданной фразы.

Ввод

Во входном файле используются только прописные латинские буквы от A до Z, пробелы, знаки препинания и символы перехода на новую строку. В файле содержится от 2 до 4000 строк длиной до 250 символов. В первой строке входного файла содержится текст дразнилки, далее следует текст стихов, сочиненных Электрувером.

Вывод

В выходной файл вывести количество анаграмм дразнилки, которые можно найти в тексте стихов.

Пример INPUT.TXT:

```
KLAPAUCIUS
PAULAUS PICK AUDIENCE,
CIKLAUPUS’ ACID.
```

OUTPUT.TXT для примера:

```
6
```

4. Игра

Входной файл	*INPUT.TXT*
Выходной файл	*OUTPUT.TXT*
Ограничение времени	*5 сек*

– Ну что, сыграем? – спросил Клапауций, тасуя колоду карт.

– Сыграем, – согласился Трурль, надевая очки.

Клапауций вытащил из колоды карты красной и зеленой масти с номерами от 1 до 10. Он отдал Трурлю все карты зеленой масти, а затем быстро перетасовал десять своих карт красной масти. Трурль тасовал карты неспешно, изредка поглядывая на стопку карт в руках Клапауция. Карты лежали рубашкой вверх, но рентгеновские очки позволяли видеть все номера на картах. “Клапауций, наверно, думает, что очки нужны мне, чтобы лучше видеть карты. Что ж, он не ошибается”.

– Ладно, начнем, – сказал Трурль, закончив перемешивать карты.

Клапауций взял верхнюю карту, перевернул и выложил на стол – это оказалась шестерка. Трурль также бросил рядом с картой Клапауция верхнюю карту из своей стопки – тоже шестерка! Клапауций отбросил обе карты в сторону и выложил следующую карту. Тоже сделал и Трурль. У Трурля оказалась пятерка, а Клапауция – тройка.

– Мое, – заметил Трурль, взял пятерку со стола и положил ее на тройку. Получившуюся стопку из двух карт он перевернул рубашкой вверх и положил под стопку карт, которые держал в руках.

Следующая карта Трурля оказалась двойкой, а у Клапауция – девятка. Клапауций положил девятку на двойку, и, перевернув, положил карты под стопку своих карт.

Игра закончилась через минуту.

– И как тебе удается всегда выигрывать? – спросил Клапауций, оставшись без карт.

– Просто мне больше везет. Сыграем еще раз?

Правила игры достаточно просты. В начале игры у игроков на руках находится два одинаковых набора карт, различающихся только цветом и порядком. Карты на руках лежат рубашкой вверх. Игра происходит следующим образом. Игроки выкладывают на стол, переворачивая, по одной верхней карте. Если карты имеют одинаковые номера, они отбрасываются и в дальнейшей игре не участвуют. Если карты имеют различные номера, то тот, у кого номер карты больше, забирает обе карты со стола. При этом карта выигравшего “раунд” игрока кладется на карту проигравшего игрока, а затем карты переворачиваются и кладутся под стопку карт в руках выигравшего игрока. Если у одного из игроков больше нет карт на руках, то он проигрывает. Игра может закончиться и вничью, если карты закончились одновременно у обоих игроков.

Очевидно, Трурль может легко свести игру к ничьей, расположив свои карты в том же порядке, что и Клапауций. Но ему нужен только выигрыш. Напишите программу, которая позволяет по известному начальному расположению карт у противника составить выигрышную перестановку из тех же карт.

Ввод

Во входном файле в первой строке содержится целое число N (2<=N<=500) – количество карт на руках у каждого игрока. Во второй строке содержится перестановка чисел от 1 до N, разделенных пробелом – расположение карт у Клапауция, перечисленных в порядке сверху вниз.

Вывод

В выходной файл в первой строке вывести слово “YES”, если существует выигрышная перестановка из карт с номерами от 1 до N, или слово “NO”, если такой перестановки нет. Во второй строке выходного файла нужно вывести одну (любую) из выигрышных перестановок, если таковая существует. Номера карт перечисляются в порядке сверху вниз и разделяются одним пробелом.

Пример INPUT.TXT:

```
4
2 3 4 1
```

OUTPUT.TXT для примера:

```
YES
2 4 1 3
```

5. Формула Счастья

Входной файл	*INPUT.TXT*
Выходной файл	*OUTPUT.TXT*
Ограничение времени	*5 сек*

Как-то сумеречной порой знаменитый конструктор Трурль пришел к своему другу Клапауцию задумчивый и молчаливый. Когда же приятель попробовал развеселить его последними кибернетическими анекдотами, неожиданно отозвался:

– Напрасно хмурое расположение моего духа пытаешься ты обратить во фривольное! Меня снедает открытие столь же печальное, сколь несомненное: я понял, что, проведя всю жизнь в неустанных трудах, для Всеобщего Блага мы не сделали ничего! Представь себе только, каким изумительным монументом нашего с тобой конструкторства была бы сияющая где-то в небе планета, к которой премножество племен галактических с упованием возводили бы очи, восклицая: “Да! Поистине, счастье возможно, в виде неустанной гармонии, как доказал великий конструктор Трурль при некотором участии друга своего Клапауция, и свидетельство этого здравствует и процветает в пределах досягаемости нашего восхищенного взора!”

– Знаешь что, Трурль? Надо бы тебя заковать и засадить в погреб, чтобы дать время опомниться. Ведь если ты и создашь неведомо где счастливцев (в чем я сомневаюсь), по-прежнему останутся создания гадкие и жестокие, и разгорится такая зависть, такие пойдут раздоры и склоки, что ты окажешься перед выбором не слишком приятным: либо твои счастливцы уступят завистникам, либо придется им этих докучных, настырных и дефективных перебить до единого – ради полной гармонии.

Трурль в бешенстве вскочил, но опомнился и разжал кулаки.

– Прощай, Фома неверующий! – заявил он холодно. – Не словами я отвечу тебе, но делом!

Воротившись домой, Трурль принялся за сочинение Формулы Счастья. За один гед (единицу счастья) он принял интенсивность блаженства путника, который с гвоздем в ботинке протопал четыре мили, а после гвоздь вынул. Путь конструктор помножил на время, поделил на колючесть гвоздя, вынес за скобки коэффициент натертости пятки и таким образом выразил счастье в системе сантиметр-грамм-секунда. Это приободрило его. Он вычислил значение Формулы еще в нескольких точках и решил изобразить их на графике.

Напишите программу, которая позволяет нарисовать график зависимости y от x по ее табличному представлению. График нужно изобразить с помощью символов ‘|’ (вертикальная черта) для изображения оси X, ‘–’ (минус) для изображения оси Y, ‘+’ (плюс) для изображения пересечения оси X и оси Y, ‘*’ (звездочка) для изображения точек функциональной зависимости и ‘.‘ (точка) для незаполненных участков. Ось X на графике направлена вправо, а ось Y – вверх. Один символ на графике соответствует единице по оси X по горизонтали и единице по оси Y по вертикали. Оси X и Y обязательно должны присутствовать на графике.

Ввод

Во входном файле в первой строке содержится целое число N (1<=N<=100) – количество известных значений переменной y. Далее следует N строк с двумя целыми числами x_i и y_i (–500<=x_i<=500, –200<=y_i<=200, i=1…N), разделенными пробелом – значения Формулы в зависимости от параметра x.

Вывод

В выходной файл вывести одну или более строк. Первая строка соответствует максимальному значению y (или 0), а последняя – минимальному значению y (или 0). Во всех строках содержится одинаковое количество символов, необходимое и достаточное для изображения всех вычисленных значений и оси Y. Первый символ строки соответствует минимальному значению x (или 0), а последний – максимальному значению x (или 0). В позиции, соответствующей оси X, выводится символ ’–’, в позиции, соответствующей оси Y, – символ ’|’, на пересечении осей – символ '+’, а в позициях, соответствующем точкам (x_i,y_i), – символ ‘*’. В остальных позициях выводится символ ‘.’.

Пример INPUT.TXT:

```
8
-10 5
-7 3
-4 2
-9 4
0 1
6 -1
3 0
8 -3
```

OUTPUT.TXT для примера:

```
*.........|........
.*........|........
...*......|........
......*...|........
..........*........
----------+--*-----
..........|.....*..
..........|........
..........|.......*
```

6. Две дороги

Входной файл	*INPUT.TXT*
Выходной файл	*OUTPUT.TXT*
Ограничение времени	*5 сек*

Однажды утром Трурль повесил на дверь записку “Ушел к Клапауцию” и отправился в гости. Подойдя к дому Клапауция, он увидел, что дверь заперта, а на двери висит записка “Пошел в гости к Трурлю”. “Как это я не заметил его по дороге?” – подумал Трурль, сделал приписку на записке и отправился обратно. В пути он внимательно смотрел по сторонам, но Клапауция так и не встретил. На своей двери Трурль обнаружил записку с припиской “Не застал. Иду домой. Клапауций”.

Это вполне могло произойти, если приятели шли по разным дорогам. Сеть дорог в городе, где живут Трурль и Клапауций, обладает следующим свойством. На каком бы перекрестке Трурль не устраивал засаду на Клапауция, Клапауций всегда мог пройти от своего дома до дома Трурля, не встретившись с ним.

Представим сеть дорог в городе в виде графа. Вершинами графа будут перекрестки (развилки), а ребрами – улицы, соединяющие эти перекрестки. Предположим, что дом Трурля находится в вершине А, а дом Клапауция – в вершине Б. Если при удалении произвольной вершины графа (кроме А и Б) всегда остается цепь, соединяющая вершины А и Б, то между этими вершинами можно построить две цепи, не имеющие общих вершин, кроме А и Б (это следует из теоремы Менгера).

Напишите программу, которая позволяет найти две цепи в графе между вершинами А и Б, не имеющие общих вершин.

Ввод

Во входном файле в первой строке содержится два целых числа, разделенных пробелом: количество вершин в графе N (3<=N<=100) и число ребер в графе M (3<=M<=N(N-1)/2). Далее следует M строк с описанием ребер графа. В каждой строке содержится два целых числа a_i и b_i (1<=a_i<=N, 1<=b_i<=N, 1<=i<=M), разделенных пробелом – это означает, что ребро соединяет вершины a_i и b_i графа.

Вывод

В выходной файл вывести две строки, содержащих информацию о двух найденных цепях между вершинами 1 и N, не имеющих общих вершин, кроме вершин 1 и N. Первым числом в строке указывается длина пути (количество ребер) k, далее следует (k+1) целое число через пробел – номера вершин, через которые проходит цепь, начиная с вершины 1 и заканчивая вершиной N. Цепь не должна содержать циклов. Если существует несколько вариантов двух цепей, вывести один (любой) из них.

Пример INPUT.TXT:

```
6 7
1 2
2 3
1 4
2 5
3 6
4 5
5 6
```

OUTPUT.TXT для примера:

```
3 1 2 3 6
3 1 4 5 6
```

7. Муха

Входной файл	*INPUT.TXT*
Выходной файл	*OUTPUT.TXT*
Ограничение времени	*5 сек*

Чтобы разузнать о новом изобретении Клапауция, Трурль смастерил электронную муху и отправил ее в дом Клапауция. Влетев в открытое окно, муха шлепнулась на лампу и стала передавать Трурлю изображение бумаг, разложенных на столе.

– Что это за абракадабра?! – воскликнул Трурль, глядя на экран.

Текст оказался разорван фасетчатыми глазами мухи на отдельные кусочки по две буквы, настолько хаотично перемешанные, что прочитать его не представлялось возможным. Объяснение неудачи было простым – Трурль настолько спешил разузнать секреты Клапауция, что неправильно подсоединил контакты отдельных элементов электронных глаз к передатчику и не проверил работоспособность электромухи.

Напишите программу, которая позволит получить хоть какую-нибудь информацию об изобретении Клапауция, восстановив текст по его искаженному изображению. Так как каждая фасетка глаза мухи направлена под своим углом, то большинство символов текста попали в изображение дважды – как первый и как второй символ в паре символов. Соединяя пары по совпадающему символу, нужно объединить в единый текст все пары символов. Возможно, текст не удастся восстановить, так как некоторые фасетки остались неприсоединенными к передатчику.

Ввод

Во входном файле в первой строке содержится число N ($0<N<=10000$) – количество пар символов, попавших в поле зрение мухи. Далее следует N строк, содержащих по два символа. Используются только прописные латинские буквы A–Z и символ _ (подчеркивание), заменяющий символ пробела.

Вывод

В выходной файл вывести одну строку – результат склейки всех пар символов в единый текст. Если существует несколько вариантов интерпретации, вывести один (любой) из них. Если склейка фрагментов невозможна, то вывести сообщение “IMPOSSIBLE!”.

Пример INPUT.TXT:

```
9
SE
ET
OP
EC
RE
_S
CR
TO
P_
```

OUTPUT.TXT для примера:

```
TOP_SECRET
```

8. WRITEF

Входной файл	*INPUT.TXT*
Выходной файл	*OUTPUT.TXT*
Ограничение времени	*5 сек*

Для управления своими машинами Трурль пишет программы на языке Пи, соединившем в себе все достоинства и недостатки языков Си и Паскаль.

В языке Пи можно обрабатывать данные следующих типов: int (16-битовое знаковое целое), longint (32-битовое знаковое целое), single (32-битовое вещественное число), double (64-битовое вещественное число), extended (80-битовое вещественное число), char (символ), string (строка произвольной длины, оканчивающая символом с кодом 0).

Для форматированного вывода результатов в языке Пи используется специальный оператор WRITEF. Оператор WRITEF вызывается с одним и более аргументами. Первым аргументом является строка, содержащая спецификации форматов, применяемые к остальным аргументам оператора. Кроме спецификаций формата строка содержит символы, выводимые без изменений. Для некоторых символов существует специальное обозначение, начинающее с символа '\' (обратная косая черта). Переход на новую строку обозначается \n, табуляция – \t, кавычки – \".

Порядок спецификаций формата должен соответствовать порядку выводимых значений. Каждая спецификация формата имеет следующую форму:

% [*флаги*] [*ширина*] [**.** *точность*] *формат*

флаги (опционально)	Флаг '–' указывает на выравнивание влево, а флаг '+' – на необходимость вывода знака результата (+ или –).
ширина (опционально)	Указывает минимальный размер поля для вывода значения. Ширина может быть задана двумя способами: 1) явно, как последовательность десятичных цифр (если ширина начинается с символа 0, то выводимое значение дополняется слева нулями); 2) неявно с помощью символа '*' (звездочка), значение для ширины содержится в списке аргументов, этот аргумент должен иметь тип int и предшествовать выводимому значению.
точность (опционально)	Указывает максимальное число знаков (или число десятичных знаков для вещественных чисел) для вывода значения. Точность может быть задана двумя способами: 1) явно, как последовательность десятичных цифр; 2) неявно с помощью символа '*' (звездочка), значение для точности содержится в списке аргументов, этот аргумент должен иметь тип int и предшествовать выводимому значению.
формат	Указывает тип выводимого значения. Для вывода значения типа int используется формат '*d*', для longint – '*ld*', для single – '*f*', '*e*', '*g*', '*E*' и '*G*', для double – '*lf*', '*le*', '*lg*', '*lE*' и '*lG*', для extended – '*Lf*', '*Le*', '*Lg*', '*LE*' и '*LG*', для char – '*c*', для string – '*s*'.

Так как символ '%' является признаком начала спецификации формата, то для вывода символа '%' в строке указывают последовательность '%%'.

Одним из существенных недостатков WRITEF является отсутствие проверки соответствия числа и типа аргументов спецификациям формата в строке. Напишите программу, которая позволит Трурлю проверить корректность аргументов оператора WRITEF. Спецификации формата в строке и список аргументов оператора WRITEF обрабатываются последовательно, слева направо. При обнаружении первой ошибки проверку нужно завершить и выдать сообщение об ошибке. Программа должна обнаруживать три вида ошибок.

1. Тип выводимого значения не соответствует спецификации формата.
2. Спецификаций форматов больше, чем количество выводимых значений.
3. Есть аргументы, для которых нет спецификаций формата.

Ввод

Во входном файле в первой строке содержится число N ($0<N<=50$) – количество аргументов у оператора WRITEF. Во второй строке (длиной не более 200 символов) указывается первый аргумент – строка в кавычках, содержащая корректные спецификации форматов. Далее следует (N–1) строка, содержащая имена типов остальных аргументов. Количество и типы аргументов могут не соответствовать спецификациям форматов.

Вывод

В выходной файл вывести одну строку – сообщение “OK” или сообщение об ошибке в форме “ERROR #”, где # – номер ошибки.

Пример INPUT.TXT:

```
3
”a[%d]=%*.2lf\n”
int
double
```

OUTPUT.TXT для примера:

```
ERROR 1
```

Пример INPUT.TXT:

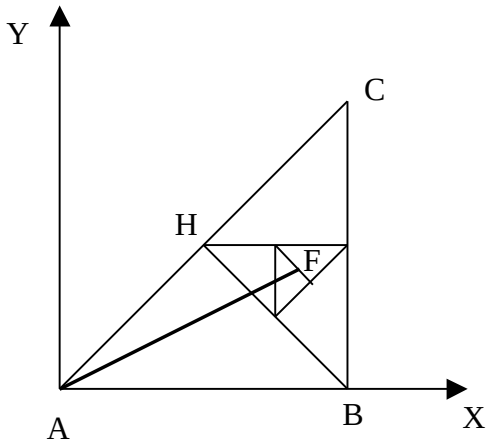
```
4
”a[%d]=%*.2lf\n”
int
int
double
```

OUTPUT.TXT для примера:

```
OK
```

9. Ловушка

Входной файл	*INPUT.TXT*
Выходной файл	*OUTPUT.TXT*
Ограничение времени	*5 сек*



Во время одного из путешествий конструкторы Трурль и Клапауций столкнулись с коварным царем Жестокусом. Но друзьям удалось спастись, поймав Жестокуса в ловушку, которую они нашли, моделируя поведение царя на бумаге.

Пусть равнобедренный прямоугольный треугольник с вершинами в точках A(0,0), B(10,0), C(10,10) определяет весь спектр возможного поведения. Царь Жестокус действует весьма прямолинейно, и его поведение описывается отрезком AF, где F – какая-то внутренняя точка треугольника ABC.

Треугольник ABC делится на две части высотой BH, проведенной к гипотенузе треугольника. Если отрезок AF пересекает отрезок BH (царю удалось вырваться из ловушки), то далее рассматриваем треугольник BHC. Если отрезок AF не пересекает отрезок BH, то пытаемся загнать царя в ловушку в треугольнике ABH. Аналогично поступаем и с выбранным треугольником, проводя в нем высоту и выбирая тот треугольник, в котором оказалась точка F. И так далее до бесконечности.

Напишите программу, которая подсчитает суммарную площадь построенных треугольников, “пронзенных” отрезком AF (т.е. треугольников, граница которых пересекается отрезком AF в двух точках).

Ввод

Во входном файле в первой строке содержится два вещественных числа x и y ($0<y<x<10$), разделенных пробелом – координаты точки F. Числа x и y заданы с тремя десятичными знаками после десятичной точки.

Вывод

В выходной файл в первой строке вывести суммарную площадь “пронзенных” отрезком AF треугольников с точностью 0.001.

Пример INPUT.TXT:

```
7.500 4.000
```

OUTPUT.TXT для примера:

```
28.125
```