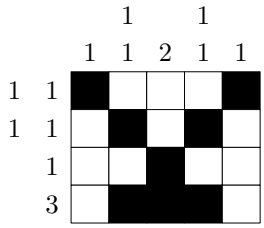
1. Японский кроссворд (nono)

1 секунда 10 очков

Японский кроссворд (нонограмма) – это головоломка, в которой необходимо восстановить изображение, закрасив часть клеток черным на квадратном поле. Изображение зашифровано числами, расположенными слева от строк а также сверху над столбцами. Числа перед каждой строкой показывают длины последовательных горизонтальных слитных групп черных клеток в этой строке – любые две группы должны быть разделены хотя бы одной белой клеткой. Аналогично числа над столбцами показывают порядок и длины последовательных вертикальных групп.



Необходимо написать программу, которая поможет составителю головоломки найти для заданного изображения описания строк и столбцов.

Входные данные. В первой строке текстового файла `nonosis.txt` дано два целых числа – число строк N ($1 \leqslant N \leqslant 100$) и столбцов M ($1 \leqslant M \leqslant 100$) заданного изображения. Каждая из следующих N строк состоит из M символов, где символ точки обозначает пустую клетку, а решетка – черную.

Выходные данные. В текстовый файл `nonoval.txt` необходимо вывести в точности $N + M$ строк. В первых N строках вывести наборы чисел, которые должны находиться перед строками изображения (на 1-й строке файла вывесли числа для первой (верхней) строки изображения, на 2-й строке числа для второй, итд). На следующих M строках вывести числа для столбцов (на $(N+1)$-й строке файла вывести числа для левой колонки, на $(N+2)$-й строке числа для второй слева колонки, итд). Если на какой-либо строке или столбце изображения нет ни одной черной клетки, в соответствующей строке файла необходимо вывести число 0.

Пример.

`nonosis.txt`
```
4 5
#...#
.#.#.
..#..
.###.
```

`nonoval.txt`
```
1 1
1 1
1
3
1
1 1
2
1 1
1
```

Пример.

`nonosis.txt`
```
1 3
#.#
```

`nonoval.txt`
```
1 1
1
0
1
```

2. Арифметические операции (tehe)

1 секунда 20 очков

Дана последовательность положительных целых чисел и положительный "ответ". Необходимо поставить между числами знаки плюс и минус так, чтобы значение получившегося выражения было равно заданному ответу.

Входные данные. На первой строке текстового файла `tehesis.txt` дана длина последовательности N ($1 \leqslant N \leqslant 1\,000$), на второй строке числа последовательности, разделенные пробелами, а на третьей – требуемый ответ. Ни одно число из последовательности, ни ответ не превышают 10 000.

Выходные данные. На единственной строке текстового файла `teheval.txt` необходимо вывести искомое выражение. Можно предполагать, что найдется хотя бы одно решение, в котором абсолютная величина ни одного промежуточного значения не превышает 10 000. Если возможных решений несколько, можно вывести любое из них.

Пример.

tehesis.txt	teheval.txt
5 5 4 3 2 1 5	5-4+3+2-1

3. Финансовый учёт (fin)

1 секунда 30 очков

Коля ведет учет домашних финансов, сохраняя все свои затраты в текстовый файл в следующем виде:

```
Затраты за март - 1000
   Еда - 500
      Сырочки - 250
      Мясо - 250
   Развлечения - 400
      Вечеринки - 200
      Кино - 200
   Спорт - 100
```

Сегодня, при очередном сохранении, текстовый редактор случайно потерял в файле все отступы, и теперь файл выглядит следующим образом:

```
Затраты за март - 1000
Еда - 500
Сырочки - 250
Мясо - 250
Развлечения - 400
Вечеринки - 200
Кино - 200
Спорт - 100
```

Напишите программу, которая поможет Коле восстановить изначальные отступы. Известно, что первая строка файла соответствует общей сумме всех расходов. Заметьте, что дерево расходов не обязательно трехуровневое, как в примере.

Входные данные. На первой строке текстового файла `finsis.txt` дано количество строк N ($1 \leqslant N \leqslant 20$) в Колином файле. На каждой из следующих N строк дано число A_i ($1 \leqslant A_i \leqslant 1\,000\,000\,000$) – сумма расходов на соответствующей строке.

Выходные данные. В текстовый файл `finval.txt` вывести в точности N строк. На строке i должно быть одно число – отступ для суммы, данной на строке $i + 1$ входного файла. Обратите внимание, что первый отступ всегда равен 0, а второй (если в Колином файле более одной строки) всегда равен 1.

Пример.

finsis.txt	finval.txt
6	0
1000	1
500	2
250	2
250	1
500	2
500	

4. Богач и дети (mort)

1 сек. / 10 сек. 40 очков

Жил как-то в Англии, еще во времена Чарльза Диккенса, миллионер Мортимер. С ним в одном городе находилось три детских дома, где жили бедные дети, которым Мортимер любил дарить подарки на Рождество.

Процедура раздачи подарков на каждое Рождество следующая:

1. Каждый ребенок берет в своем детском доме корзинку, с которой он идет получать подарок.
2. Мортимер кидает подарки в толпу детей, а они ловят их своими корзинками.
3. Каждый подарок кто-либо обязательно поймает.
4. То, кто именно поймает каждый подарок – дело случая, и вероятность для каждого ребенка поймать очередной подарок пропорциональна площади его корзинки.
5. У всех детей одного и того же детского дома одинаковые корзинки.
6. Как только какой-либо ребенок ловит в свою корзинку подарок, он тут же возвращается с ним в детский дом и больше в ловле подарков не участвует.
7. Детей может быть больше, чем подарков :(

У каждого подарка есть стоимость. Необходимо найти для каждого детского дома, какова средняя ожидаемая суммарная стоимость подарков, которые принесут дети этого дома.

Входные данные. В текстовом файле `mortsis.txt` задано следующее:

1. На первой строке даны два целых числа $0 \leqslant L_1 \leqslant 100$ и $1 \leqslant K_1 \leqslant 100$, которые обозначают количество детей в первом детском доме, и площадь каждой из корзинок этих детей соответственно.
2. На второй строке даны два целых числа $0 \leqslant L_2 \leqslant 100$ и $1 \leqslant K_2 \leqslant 100$, которые обозначают количество детей во втором детском доме, и площадь их корзинок соответственно.
3. На третьей строке даны два целых числа $0 \leqslant L_3 \leqslant 100$ и $1 \leqslant K_3 \leqslant 100$, которые обозначают количество детей в третьем детском доме и площадь их корзинок соответственно.
4. На четвертой строке дано количество подарков $1 \leqslant N \leqslant L_1 + L_2 + L_3$.
5. На оставшихся N строках даны стоимости подарков – целые числа в промежутке $1 \ldots 1\,000$. Мортимер кидает подарки точно в заданном порядке.

Выходные данные. В текстовый файл `mortval.txt` необходимо вывести на трех строках три действительных числа (с точностью не менее 0.0001), которые обозначают среднюю ожидаемую суммарную стоимость подарков для каждого из детских домов.

Пример.

mortsis.txt	mortval.txt
1 1	6.6666666666667
1 2	11.3333333333333
1 3	12
2	
10	
20	

Единственный ребенок из первого детского дома получит первый подарок с вероятностью $1/6$, это даст ожидаемую суммарную стоимость $10/6 = 5/3 \approx 1.666667$ и в поимке других подарков он участвовать не будет. С вероятностью $1/3$ первый подарок получит ребенок из второго детского дома и уйдет домой. В таком случае первый ребенок получит

второй подарок с вероятностью 1/4. Ожидаемая стоимость этого подарка в таком случае $20/4/3 = 5/3 \approx 1.666667$. С вероятностью 1/2 первый подарок получит ребенок из третьего детского дома и уйдет домой. В таком случае первый ребенок получит второй подарок с вероятностью 1/3, ожидаемая стоимость этого $20/2/3 = 10/3 \approx 3.333333$. Итого получаем ответ для первого детского дома $5/3 + 5/3 + 10/3 = 20/3 \approx 6.666667$. Аналогичный подсчет можно провести и для детей из второго и третьего детских домов.

Пример.

mortsis.txt	mortval.txt
2 1	27.75
2 2	51.5
1 2	25.75
2	
63	
42	

5. Двоичное дерево множеств (puu)

1 секунда 50 очков

Двоичное дерево множеств – это структура данных, аналогичная двоичному дереву, но позволяющая в каждой вершине хранить не один, а несколько элементов. Так же как и в случае обычного двоичного дерева, дерево множеств должно удовлетворять условию упорядоченности: элементы в каждой вершине должны быть все строго больше всех элементов всех вершин левого поддерева, и строго меньше всех элементов всех вершин правого поддерева.

Так же как и обычное двоичное дерево, дерево множеств позволяет осуществлять поиск элементов. В каждой вершине можно решить, принадлежит ли искомое значение множеству элементов этой вершины, или же нужно продолжить поиск в левом или в правом поддереве. Если считать одно такое решение за одну элементарную операцию, то для нахождения вершины, содержащей заданное значение, понадобится столько же элементарных операций, сколько уровней дерева нужно пройти от корня до этой вершины (т.е. глубина этой вершины в дереве).

В данном задании мы рассмотрим ситуацию, где из-за определенных особенностей управления памятью максимальное количество элементов, которые могут храниться в одной вершине дерева разное, и зависит от глубины вершины. А именно, для вершины на глубине i максимальное количество хранимых в ней элементов m_i задается следующим образом:

$$m_i = \begin{cases} M & \text{если } i = 1 \text{ (т.е. это корень дерева)}, \\ \max(1, m_{i-1} - D_{((i-1) \bmod K)+1}) & \text{если } i > 1, \end{cases}$$

где M, K и D_j – фиксированные целые числа, а $(i-1) \bmod K$ означает остаток от деления числа $i-1$ на K. Например, если $M = 4$, $K = 2$, $D_1 = 1$, $D_2 = 2$, то в корне дерева может храниться не более $m_1 = M = 4$ элементов. В обеих вершинах, непосредственно связанных с корнем не более $m_2 = m_1 - D_1 = 4 - 1 = 3$. В детях этих вершин, в свою очередь, не более $m_3 = m_2 - D_2 = 3 - 2 = 1$ элементов, а в вершинах во всех остальных слоях дерева может храниться максимум $m_i = \max(1, ...) = 1$ элемент.

Вдобавок, известно, что в дереве придется искать разные элементы с разной вероятностью, поэтому для заданного набора элементов сбалансированное дерево не обязательно будет оптимальным в плане среднего времени поиска. Например, если в подавляющем числе случаев в дереве запрашивается поиск минимального из хранящихся там элементов, стоит расположить его в самом корне, оставив в таком случае всё левое поддерево пустым.

Задача – для заданного набора элементов и вероятностей их запросов, найти среднее ожидаемое количество элементарных операций, необходимое для осуществления таких запросов в двоичном дереве множеств оптимальной формы.

Входные данные. На первой строке текстового файла `puusis.txt` даны три целых числа: количество элементов N ($1 \leqslant N \leqslant 100$), а также M ($1 \leqslant M \leqslant N$) и K ($1 \leqslant K \leqslant N$). На следующих K строках по одному целому числу: на строке $j+1$ дано значение D_j ($0 \leqslant D_j \leqslant M$). На последней строке даны N действительных чисел P_i ($0 \leqslant P_i \leqslant 1$; $\sum_{i=1}^{N} P_i = 1$): вероятности запросов каждого элемента. Вероятности упорядочены по значению соответствующего элемента (первая вероятность P_1 соответствует наименьшему элементу). Обратите внимание, что сами значения элементов для решения задачи не нужны, и можно предполагать что они все различны.

Выходные данные. На единственной строке текстового файла `puuval.txt` вывести одно действительное число – среднее ожидаемое число элементарных операций, необходимое для осуществления поиска элемента в построенном на заданном множестве элементов оп-

тимальным образом двоичном дереве множеств. Значение может отличаться от точного ответа не более чем на 10^{-6}.

Пример.

`puusis.txt`
```
4 2 1
2
0.2 0.2 0.3 0.3
```

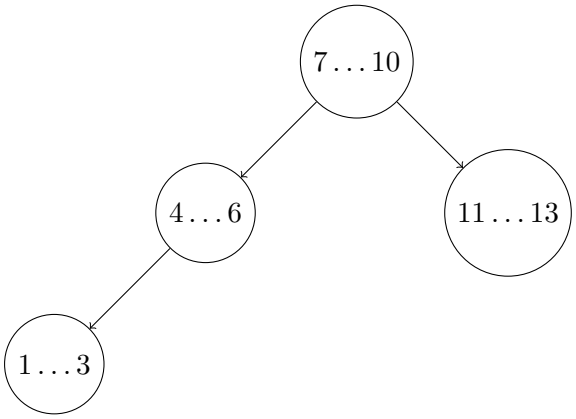
`puuval.txt`
```
1.5000000
```

Пример.

`puusis.txt`
```
13 4 2
1
0
0.06 0.06 0.06 0.06 0.06 0.06 0.115 0.115 0.115 0.115 0.06 0.06 0.06
```

`puuval.txt`
```
1.7200000
```

Оптимальное дерево во втором примере следующее:



6. Периметр (per)

50 очков

В олимпиадном подготовительном лагере по информатике ученики должны были решить следующую задачу:

Периметр объединения прямоугольников

На плоскости дано N прямоугольников, стороны которых параллельны координатным осям. Необходимо найти периметр объединения этих прямоугольников, т.е. длину общего внешнего контура. "Дырки" внутри получившейся фигуры учитывать не нужно, но если внутри какой-либо "дырки", окажется в свою очередь еще какая-нибудь составленная из прямоугольников фигура, которая не касается внешней ни в одной точке, то ее необходимо учитывать при подсчете общей длины контура.

Входные данные. На первой строке файла дано число прямоугольников N ($1 \leqslant N \leqslant 1\,000$). На каждой из следующих N строк даны четыре действительных числа LX, LY, UX и UY, где LX и LY – координаты нижнего левого угла прямоугольника, а UX и UY – координаты верхнего правого угла. Значения координат – числа между 0 и 1 000 000 включительно, для представления которых достаточно 32-хбитного числового типа `float`.

Выходные данные. На единственной строке выходного файла необходимо вывести длину периметра объединения данных прямоугольников с точностью не менее трех цифр после запятой.

Пример.

Входной файл	Выходной файл
3 7.0 170.0 99.5 190.0 0.5 100.0 12.0 225.0 10.0 80.0 50.0 110.0	564.0

Ваша задача – создать тесты для проверки корректности предоставленных учениками решений.

Оценивание. Все тестовые данные необходимо записать в один текстовый файл в следующем формате. На первой строке файла вывести число тестов K ($1 \leqslant K \leqslant 20$). За ним следуют K блоков, соответствующие K тестам. Первая строка каждого блока должна содержать ожидаемый правильный ответ для соответствующего теста, а последующие строки – входные данные в формате, описанном в задании. Данный текстовый файл нужно закачать в качестве решения.

Пример.

Выходной файл

```
2
4.0
1
1.0 1.0 2.0 2.0
564.0
3
7.0 170.0 99.5 190.0
0.5 100.0 12.0 225.0
10.0 80.0 50.0 110.0
```

Оценивание. На предоставленных тестовых данных прогоняется десять программ, каждая из которых содержит какую-нибудь ошибку, но тем не менее дает правильные ответы

на тесты из примера выше. За каждую программу, которая провалится на одном из предоставленных участником тестов, решение получает 5 очков.

7. Взлом паролей (pass)

50 очков

Компьютерная фирма ABCDEF-software создала веб-приложение. В приложении есть модуль управления пользователями, про который известно следующее:

- имя пользователя и пароль могут быть длиной $1 \ldots 20$ знаков, и состоять из латинских символов `A`...`Z`, `a`...`z`, или цифр `0`...`9`;
- имена пользователей должны быть уникальными;
- имена пользователей и пароли чувствительны к регистру.

Вам доступны две функции данного модуля:

- показ списка всех имен пользователей;
- создание нового пользователя с заданным именем и паролем.

В модуль уже введены некоторые пользователи, а ваша задача – установить их пароли.

Оценивание. В качестве решения необходимо предоставить текстовый файл, в каждой строке которого даны имя пользователя и его пароль, разделенные пробелом. При оценивании учитывается количество правильно определенных паролей, а так же количество добавленных в систему новых пользователей:

1. Все решения, которые корректно определили хотя бы один пароль, упорядочиваются по количеству определенных паролей.
2. Если два решения корректно определили одинаковое количество паролей, лучшим считается решение, которое добавило в систему меньше новых пользователей.
3. Первое решение в полученном рейтинге получает 50 очков, последнее –5 очков; остальные решения получают очки пропорционально их расположению в списке.

Примечание. Суть данной задачи – определение паролей с помощью двух предоставленных функций, а не атака на сервер любым другим образом. Разрешено (и даже советуется) использовать программы, которые будут автоматически посылать необходимые запросы, но это необязательно. Стоит помнить, что при добавлении слишком большого количества пользователей работа модуля может стать слишком медленной или вообще прерваться.

Использование модуля управления пользователями

Модуль доступен по адресу `http://cms.eio.ut.ee/pass/`.

Для использования необходимо первым делом авторизоваться с помощью того же имени и пароля, которые вы использовали для логина на сайт соревнования `http://cms.eio.ut.ee/`. В результате модуль выдаст браузеру куки (*cookie*, `https://ru.wikipedia.org/wiki/HTTP_cookie`), которое действует в течение всего соревнования. (Если вы желаете решать задачу с помощью скриптов, не нужно программно реализовывать логин на модуль – достаточно добавить это куки к посылаемым запросам).

Вы можете использовать две версии модуля:

1. `http://cms.eio.ut.ee/pass/test.cgi`
   Эта версия содержит дополнительную функцию сброса, которая удалит всех созданных вами пользователей и восстановит изначальное состояние модуля. Эту версию можно использовать для тестирования своего решения – пунктов за угадывания паролей в этой версии получить невозможно.

   Изначальное содержание базы пользователей в данной версии таково:
   'adam','eve'
   'joe','hacker1'
   'bill','1234'

2. `http://cms.eio.ut.ee/pass/live.cgi`
   В данной версии нет функции сброса, и для каждого участника изначально сгенерирован уникальный набор пользователей и паролей. Оцениваются лишь угаданные пароли для этой версии модуля.

Модуль принимает следующие параметры в строке запроса:

- `q=list` — показывает список пользователей в базе; дополнительных параметров нет;
- `q=reset` — сбрасывает базу в начальное состояние; дополнительных параметров нет;
- `q=adduser` — добавляет нового пользователя; дополнительные параметры: `user=`, `pass1=`, `pass2=` (имя создаваемого пользователя и пароль, повторенный дважды, как принято у форм выбора паролей).

К модулю можно обращаться как `GET` так и `POST` запросами. Модуль отвечает только на авторизованные запросы (т.е. содержащие корректное куки).