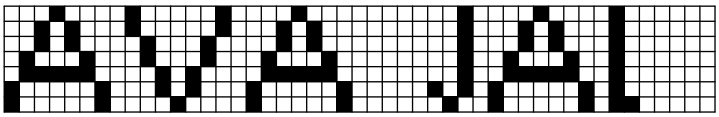
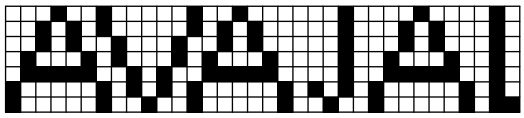
1. Сжатие текста

1 секунда 20 очков

Для выведения текста на растровое устройство (экран, принтер и т.д.) каждую букву алфавита представляют картинкой, состоящей из $K \times L$ пикселей. Например, на рисунке снизу видно изображение, составленное из растровых изображений нескольких букв, так что между каждыми двумя соседними растровыми изображениями 7×7 добавлен также один пустой столбик.



Такой наивно составленный текст, как правило, не выглядит особо хорошо, так как промежутки между буквами неравномерные. Например, на приведённом рисунке промежуток между буквами 'A' и 'J' больше, чем между 'A' и 'V', а он в свою очередь кажется большим, чем между 'A' и 'L'. Для улучшение впечатления при добавлении очередной буквы в конец ряда её двигают влево настолько, насколько возможно, так чтобы ни один её пиксель не касался с имеющимися чёрными пикселями (ни краем, ни углом). Результат сжатия приведённого выше текста показан на следующем рисунке.



При расположении текста на странице необходимо знать ширину каждого слова в пикселях. Написать программу, которая получает растровые изображения всех букв алфавита и подсчитывает ширину данного текста в пикселях после сжатия.

Входные данные. В этом задании входные данные заданы в двух файлах.

В текстовом файле `kernfont.sis` 26 блоков, которые описывают буквы A. . . Z латинского алфавита. Каждый блок состоит из 7 строк, в каждой из которых 7 целых чисел, где 1 означает чёрный, и 0 белый пиксель. Этот файл одинаковый во всех тестах и доступен вместе с остальными примерами входными и выходных данных на сервере соревнований.

В единственной строке файла `kern.sis` дано слово S, состоящее из больших латинских букв (без пробелов) длиной до 100 символов. Этот файл разный в разных тестах.

Выходные данные. В единственную строку текстового файла `kern.val` вывести длину в пикселях сжатого растрового изображения данного слова.

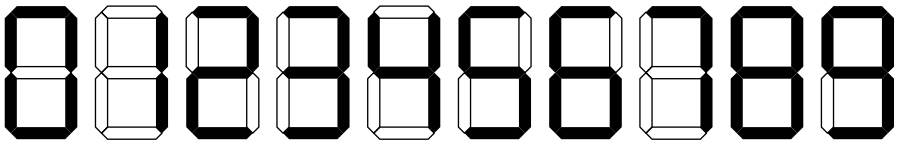
Пример.

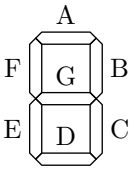
`kern.sis`	`kern.val`
`AVAJAL`	`34`

2. Часы

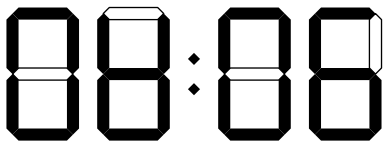
1 секунда 40 очков

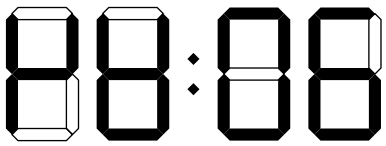
На табло дигитальных часов для показа времени от 00 : 00 до 23 : 59 цифры 0...9 образуются путём включения семи индикаторов в разных комбинациях, как видно на рисунке слева. Сегменты обозначены буквами A...G, как видно на рисунке справа.





Если какой-то сегмент перегорит, на табло может высветиться фигура, не являющаяся цифрой. Например, на рисунке внизу слева вместо второй цифры планироваось показать цифру 8, но сегмент A не работает. Также может случиться, что цифра, горящая на табло, на самом деле не та, которую хотели показать. Например, на последней позиции горит цифра 6, но возможно, там хотели показать 8, а сегмент B не работает.





Также может оказаться, что другая электроника в часах сломана, и на табло показано что-то, что вообще не может быть временем, даже с учётом того, что какие-то сегменты могли перегореть. Такая ситуация показана на рисунке вверху справа, где первая цифра количества часов может быть только 8, но такого не бывает ни в какой момент времени.

Написать программу, которая находит самое раннее и самое позднее время, которое может показывать данное табло, если известно, которые сегменты сейчас горят и какие не горят.

Входные данные. В текстовом файле `kell.sis` четыре строки, соответственно десятки и единицы числа часов и десятки и единицы числа минут. В каждой строке дано 7 разделённых пробелами целых чисел A, B, C, D, E, F, G. Каждое из этих чисел может быть либо 1, либо 0, где 1 означает, что соответствующий сегмент горит, и 0 означает, что не горит — он либо выключен, либо перегорел.

Выходные данные. В первую строку текстового файла `kell.val` вывести самое раннее, и во вторую строку — самое позднее время в виде $HH : MM$, где HH — это двухразрядное число часов, и MM — двухразрядное число минут. Если то, что показано на табло, не может образоваться только из-за перегоревших сегментов (то есть, очевидно, ещё какая-то электроника в часах дала сбой), вывести в единственную строку файла слово `VIGA`.

Пример.

`kell.sis`
```
1 1 1 1 1 1 0
0 1 1 1 1 1 1
1 1 1 1 1 1 0
1 0 1 1 1 1 1
```

`kell.val`
```
08:06
08:08
```

Пример.

`kell.sis`
```
0 1 0 0 1 1 1
0 1 1 1 1 1 1
1 1 1 1 1 1 0
1 0 1 1 1 1 1
```

`kell.val`
```
VIGA
```

Оценивание. В этом задании за тесты с ответом `VIGA` очки получат только те программы, которые правильно решат хотя бы один из тестов с другими ответами.

3. Ошибки

1 секунда 40 очков

У фирмы по производству программного обеспечения есть своя база данных с ошибками, которые нашли пользователи или тестеровщики в её продуктах. Часто для исправления некоторых ошибок необходимо, чтобы какие-то другие ошибки были уже исправлены.

Также иногда случается, что информация об одной и той же ошибке приходит из нескольких источников, и оператор базы данных, когда добавляет информацию, не замечает, что такая ошибка уже есть. Если это замечают позже, то дубликат нельзя так просто удалить, потому что клиенту уже дали номер этой ошибки, и клиент использует этот номер, чтобы посмотреть в базе, как продвигаются дела с исправлением его проблемы. В этом случае в дублированную запись добавляется ссылка на ту ошибку, копией которой она является.

Написать программу, которая находит, сколько уникальных ошибок зарегистрировано в базе данных, и в каком порядке их следует исправлять.

Перед обнаружением дубликатов данные о зависимостях между ошибками могут быть введены для нескольких записей, дублирующих друг друга. При планировании порядка работ нужно учесть все зависимости, относящиеся к дублированным записям.

Входные данные. В первой строке текстового файла `vead.sis` дано количество записей в базе N ($1 \leqslant N \leqslant 50\,000$), количество дубликатов K ($1 \leqslant K \leqslant 1000$) и количество зависимостей между ошибками M ($1 \leqslant M \leqslant 50\,000$). Записи в базе данных пронумерованы $1 \ldots N$.

В каждой из следующих K строк дано два целых числа A_i и B_i ($1 \leqslant A_i \leqslant N$, $1 \leqslant B_i \leqslant N$), которые означают, что запись B_i является дубликатом записи A_i.

В каждой из следующих M строк дано два целых числа C_i и D_i ($1 \leqslant C_i \leqslant N$, $1 \leqslant D_i \leqslant N$), которые означают, что ошибка D_i не может быть исправлена, пока не исправлена ошибка C_i.

Выходные данные. В первую строку текстового файла `vead.val` вывести количество уникальных ошибок V, и в следующие V строк вывести номера ошибок в том порядке, в котором их можно исправлять. Каждый номер выводить в отдельную строку. Если возможных порядков несколько, вывести любой их них. Можно предполагать, что найдётся как минимум один порядок. Из каждого комплекта дубликатов можно выводить любой из номеров.

Пример.

vead.sis	vead.val
7 3 3	4
7 6	2
6 1	3
2 5	1
2 1	4
5 4	
3 4	

Записи 1, 6 и 7 дублируют друг друга. Записи 2 и 5 также дублируют друг друга. Исправление ошибки 4 зависит от исправления ошибок 3 и 2-5. Также исправление ошибок 1-6-7 зависит от исправления ошибок 2-5. Ошибки 1-6-7 и 4, а также 1-6-7 и 3, также как 2-5 и 3 можно исправлять вне зависимости друг от друга.

Оценивание. В этом задании нахождение числа уникальных записей даёт 25% очков. Если ваша программа не может найти порядок исправления ошибок, просто не выводите его (создайте выходной файл с одной строкой).

1. Телефонная книга

1 секунда — 20 очков

Рассмотрим следующий алгоритм для поиска номера в телефонной книжке по имени:

- находим страничку ровно посередине телефонной книжки (если в книжке чётное число страниц, выбираем первую из двух страниц в середине);
- если искомое имя есть на этой странице, поиск закончен;
- если искомое имя раньше по алфавиту, то выкидываем текующую страницу и все, что за ней, и продолжаем поиск по тому же алгоритму среди оставшихся страниц;
- если искомое имя дальше по алфавиту, то выкидываем текующую страницу и все, что до неё, и продолжаем поиск по тому же алгоритму среди оставшихся страниц.

Написать программу, которая по заданной толщине телефонной книжки и местонахождению искомой записи находит, какие страницы просматривали в ходе поиска.

Входные данные. В единственной строке текстового файла `tele.sis` дано два разделённых пробелом целых числа: число страниц в телефонной книжке N ($1 \leqslant N \leqslant 1\,000\,000$) и номер страницы L ($1 \leqslant L \leqslant N$), на которой находится искомая запись.

Выходные данные. В текстовой файл `tele.val` вывести номера тех страниц, которые рассматривали в ходе поиска. Номера вывести в том порядке, в каком их рассматривали, каждый в отдельной строке.

Пример.

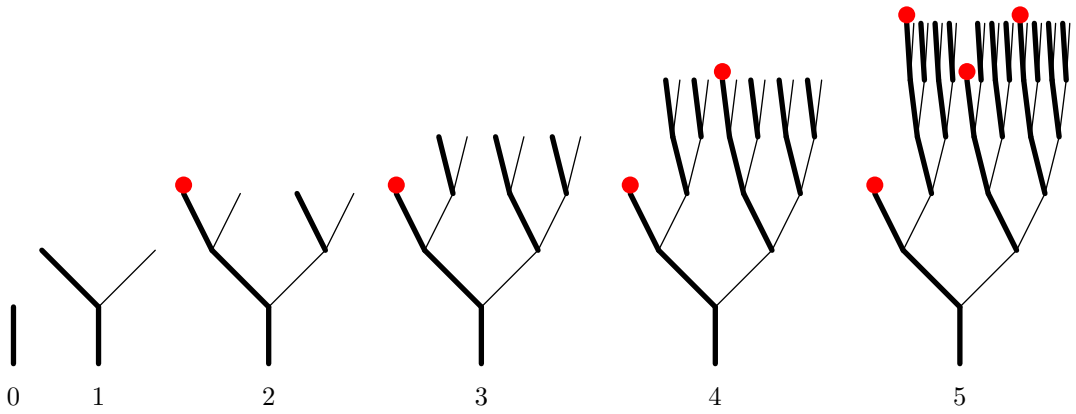
tele.sis	tele.val
9 3	5 2 3

В 9-страничной телефонной книжке начинаем поиск с 5-ой страницы. Заглянув туда, понимаем, что искомая запись находится ближе к началу книжки, и поэтому продолжаем поиск среди страниц от 1 до 4. Следующей рассматриваем 2-ую страницу, и видим, что искомая запись должна находиться ближе к концу книжки. Продолжаем поиск среди страниц от 3 до 4. Теперь заглядываем на 3-ю страницу, где и находим искомую запись.

2. Растение

1 секунда 40 очков

Комнатное растение *planta progressibilia* растёт по определённым правилам. Каждую неделю каждая ветка растения разветвляется на две, главную и боковую ветку, и они обе становятся в течение недели на одну единицу длиннее. Если какая-то ветвь растения росла как главная ветвь в течение трёх недель, на её конце появляется прекрасный цветок, и эта ветвь больше не растёт.



Написать программу, которая подсчитывает, сколько новых цветков появится на растении в конце N-ой недели после того, как в горшке посадили ветку длиной в одну единицу.

Входные данные. В единственной строке текстового файла `taim.sis` дан порядковый номер недели N ($1 \leqslant N \leqslant 75$).

Выходные данные. В единственную строку текстового файла `taim.val` вывести количество цветок, которые появятся на растении по прошествии N недель.

Пример.

`taim.sis`	`taim.val`
`5`	`2`

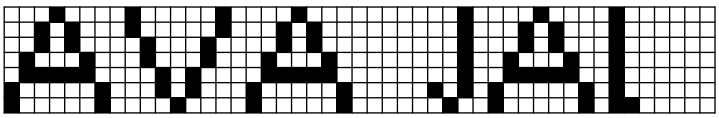
Пример.

`taim.sis`	`taim.val`
`6`	`3`

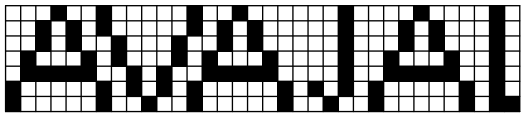
3. Сжатие текста

1 секунда 40 очков

Для выведения текста на растровое устройство (экран, принтер и т.д.) каждую букву алфавита представляют картинкой, состоящей из $K \times L$ пикселей. Например, на рисунке снизу видно изображение, составленное из растровых изображений нескольких букв, так что между каждыми двумя соседними растровыми изображениями 7×7 добавлен также один пустой столбик.



Такой наивно составленный текст, как правило, не выглядит особо хорошо, так как промежутки между буквами неравномерные. Например, на приведённом рисунке промежуток между буквами 'A' и 'J' больше, чем между 'A' и 'V', а он в свою очередь кажется большим, чем между 'A' и 'L'. Для улучшение впечатления при добавлении очередной буквы в конец ряда её двигают влево настолько, насколько возможно, так чтобы ни один её пиксель не касался с имеющимися чёрными пикселями (ни краем, ни углом). Результат сжатия приведённого выше текста показан на следующем рисунке.



При расположении текста на странице необходимо знать ширину каждого слова в пикселях. Написать программу, которая получает растровые изображения всех букв алфавита и подсчитывает ширину данного текста в пикселях после сжатия.

Входные данные. В этом задании входные данные заданы в двух файлах.

В текстовом файле `kernfont.sis` 26 блоков, которые описывают буквы A...Z латинского алфавита. Каждый блок состоит из 7 строк, в каждой из которых 7 целых чисел, где 1 означает чёрный, и 0 белый пиксель. Этот файл одинаковый во всех тестах и доступен вместе с остальными примерами входными и выходных данных на сервере соревнований.

В единственной строке файла `kern.sis` дано слово S, состоящее из больших латинских букв (без пробелов) длиной до 100 символов. Этот файл разный в разных тестах.

Выходные данные. В единственную строку текстового файла `kern.val` вывести длину в пикселях сжатого растрового изображения данного слова.

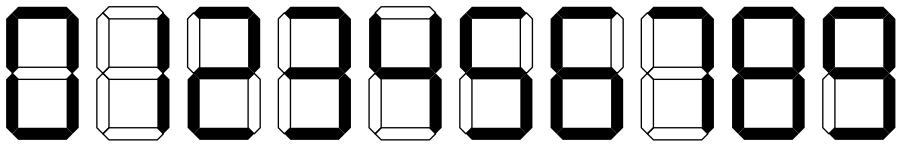
Пример.

`kern.sis`	`kern.val`
`AVAJAL`	`34`

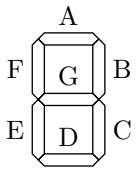
1. Дигитальное табло

1 секунда 20 очков

На многих дигитальных табло числа 0...9 изображают с помощью включения семи сегментов индикатора в различных комбинациях, как показано на рисунке внизу.



Сегменты обозначены буквами A...G, как показано на рисунке внизу.



Если один из сегментов перегорит, на табло может получиться нечто, не являющееся цифрой. Например, на рисунке внизу слева, очевидно, планировалось изобразить цифру 8, но сегмент A не работает. Также может случиться, что цифра, изображённая на табло, вовсе не та, которую планировалось показать. Например, на рисунке внизу справа видна цифра 6, но возможно, что на самом деле планировалось показать 8, а сегмент B не работает.

Написать программу, которая находит все цифры, которые, возможно, хотели показать на табло, если известно, какие сегменты сейчас горят, а какие нет.

Входные данные. В единственной строке текстового файла `nait.sis` дано семь разделённых пробелами целых чисел A, B, C, D, E, F, G. Каждое из них может иметь значение 1 или 0, где 1 означает, что соответствующий сегмент горит, и 0 означает, что не горит — он либо не включен, либо перегорел.

Выходные данные. В текстовой файл `nait.val` вывести все цифры, которые могут подразумеваться на табло, описанном во входном файле. Если возможных цифр несколько, вывести их в возрастающем порядке, каждое в отдельной строке.

Пример.

nait.sis	nait.val
0 1 1 1 1 1 1	8

Пример.

nait.sis	nait.val
1 0 1 1 1 1 1	6 8

2. Рабочий план

1 секунда 40 очков

Для постройки дома нужно сделать ряд работ. Для начала некоторых работ необходимо, чтобы некоторые другие работы были уже сделаны к этому моменту. Например, нельзя начать класть крышу прежде, чем построены стены, а стены нельзя начать строить, пока не готов фундамент.

Написать программу, которая, получив данные о том, какие работы от каких зависят, выводит список тех работ, которые можно начинать выполнять сразу.

Входные данные. В первой строке текстового файла `tood.sis` дано два разделённых пробелом целых числа: число работ N ($1 \leqslant N \leqslant 50\,000$) и число зависимостей между работами M ($1 \leqslant M \leqslant 50\,000$). Работы пронумерованы $1 \ldots N$. В каждой из следующих M строк файла дано два разделённых пробелом целых числа A_i и B_i ($1 \leqslant A_i \leqslant N, 1 \leqslant B_i \leqslant N$), которые означают, что работу B_i нельзя начать выполнять, пока не будет выполнена работа A_i.

Выходные данные. В текстовой файл `tood.val` вывести номера тех работ, которые можно начать выполнять сразу. Каждый номер вывести в отдельной строке. Порядок номеров в файле неважен, но каждый номер следует вывести ровно один раз. Можно предполагать, что всегда найдётся хотя бы одна работа, которую можно сразу начать выполнять.

Пример.

tood.sis	tood.val
4 3	2
2 1	3
2 4	
3 4	

3. Телефонная книга

1 секунда 40 очков

Рассмотрим следующий алгоритм для поиска номера в телефонной книжке по имени:

- находим страничку ровно посередине телефонной книжки (если в книжке чётное число страниц, выбираем первую из двух страниц в середине);
- если искомое имя есть на этой странице, поиск закончен;
- если искомое имя раньше по алфавиту, то выкидываем текующую страницу и все, что за ней, и продолжаем поиск по тому же алгоритму среди оставшихся страниц;
- если искомое имя дальше по алфавиту, то выкидываем текующую страницу и все, что до неё, и продолжаем поиск по тому же алгоритму среди оставшихся страниц.

Написать программу, которая по заданной толщине телефонной книжки и местонахождению искомой записи находит, какие страницы просматривали в ходе поиска.

Входные данные. В единственной строке текстового файла `tele.sis` дано два разделённых пробелом целых числа: число страниц в телефонной книжке N ($1 \leqslant N \leqslant 1\,000\,000$) и номер страницы L ($1 \leqslant L \leqslant N$), на которой находится искомая запись.

Выходные данные. В текстовой файл `tele.val` вывести номера тех страниц, которые рассматривали в ходе поиска. Номера вывести в том порядке, в каком их рассматривали, каждый в отдельной строке.

Пример.

tele.sis	tele.val
9 3	5 2 3

В 9-страничной телефонной книжке начинаем поиск с 5-ой страницы. Заглянув туда, понимаем, что искомая запись находится ближе к началу книжки, и поэтому продолжаем поиск среди страниц от 1 до 4. Следующей рассматриваем 2-ую страницу, и видим, что искомая запись должна находиться ближе к концу книжки. Продолжаем поиск среди страниц от 3 до 4. Теперь заглядываем на 3-ю страницу, где и находим искомую запись.