Задача 1 Poly-число

Введём понятие «Poly-числа» порядка N для любого целого N > 1. «Poly-число» порядка N – это целое положительное число, двоичная запись которого содержит единицы только в разрядах с номерами, являющимися обобщёнными числами Фибоначчи порядка N. В остальных разрядах «Poly-числа» содержатся нули. Младший разряд записи «Poly-числа» имеет номер 0, следующий за ним – 1 и т. д.. Обобщённые числа Фибоначчи порядка N задаются рекуррентными формулами: $F^N_i = F^N_{i-1} + F^N_{i-2} + ... + F^N_{i-N}$ для i > N-1, $F^N_0 = F^N_1 = ... = F^N_{N-2} = 0$, $F^N_{N-1} = 1$. Например, последовательность обобщённых чисел Фибоначчи порядка N = 7 имеет вид F^7_i: 0, 0, 0, 0, 0, 0, 1, 1, 2, 4, 8, 16, 32, 64, 127, 253, ... Двоичная запись «Poly-числа» порядка N = 7 имеет вид: ...00010000000000000010000000100010111. Последовательность обобщённых чисел Фибоначчи порядка N = 2 имеет вид F^2_i: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, ... Двоичная запись «Poly-числа» порядка N = 2 имеет вид: ...0100000000000010000000 10000100101111.

Составьте программу, которая принимает на вход целые числа N, A и B ($2 \le N \le 100$, $0 \le A, B \le 63000$). Число N даётся в первой строке, число A – во второй, число B – в третьей. Программа выводит разряды двоичной записи «Poly-числа» порядка N, номера которых заключены между A и B. Разряды выводятся *по возрастанию* их номеров. При A = B программа выводит разряд двоичной записи «Poly-числа» с номером A.

Формат ввода: В первой строке содержится число N – целое, неотрицательное ($2 \le N \le 100$). Во второй строке содержится число A – целое, неотрицательное ($0 \le A \le 63000$). В третьей строке содержится число B – целое, неотрицательное ($0 \le B \le 63000$).

Формат вывода: В первой и единственной строке выводятся искомые разряды двоичной записи «Poly-числа» порядка N, сначала младшие, затем старшие (то есть, в обратном порядке, если сравнивать с обычной записью двоичных чисел.

Пример №1:	*Пример №2:*	*Пример №3:*
ввод:	*ввод:*	ввод:
7	100	2
16	0	5
2	0	8
вывод:	*вывод:*	*вывод:*
101000100000001	1	1001

Задача 2.1 CPU-usage - 1

При миграции тестирующих серверов системы “Пунформатикс” на отечественное железо возникла задача перепроверки всех решений. Но так как решений много, нужно понять сколько времени займет эта процедура. Для оценки времени работы решили использовать оценку суммарного процессорного времени работы программы на всех тестах.

Вам дается архив с протоколами тестирования в формате xml. Один файл - протокол одной программы. Отчет по каждому запущенному тесту находится в теге <testing-report>/<tests>/<test> с атрибутом "time", статус успешности выполнения теста в атрибуте status.

Вам требуется для каждой пары значений из тегов cpu-model и cpu-mhz вывести их значения в формате "cpu-model" cpu-mhz и после пробела два числа - первое число это суммарное время работы всех программ на всех тестах с вердиктом OK/WA/PE/RE, второе число суммарное время работы всех программ с вердиктом TL. Например:

"Intel Xeon E3-12xx v2 (Ivy Bridge, IBRS)" 2099.998 2 0

Отсортируйте результат в порядке убывания первого числа, при равенстве в порядке убывания второго, при равенстве по паре cpu-model / cpu-mhz лексикографически.

Файлы которые не выглядят как протокол тестирования являются протоколом ошибок компиляции и могут быть проигнорированы.

Требуется сдать только файл ответа, пример для архива состоящего только из одного файла с содержанием

```
Content-type: text/xml

<?xml version="1.0" encoding="utf-8"?>
<testing-report run-id="1" judge-id="40" status="OK" scoring="KIROV" run-tests="50" contest-id="2252" real-time-available="yes" max-memory-used-available="yes" correct-available="yes" tests-passed="50" score="100" max-score="100" time-limit-ms="3000" real-time-limit-ms="6000" marked-flag="no" >
  <uuid>fc1b49c2-3a85-47e5-a90b-7a681bc829e6</uuid>
  <host>ejudge-vm-64</host>
  <cpu-model>Intel Xeon E3-12xx v2 (Ivy Bridge, IBRS)</cpu-model>
  <cpu-mhz>2099.998</cpu-mhz>
  <compiler_output></compiler_output>
  <tests>
    <test num="1" status="OK" time="2" real-time="3" max-memory-used="1925120" nominal-score="1" score="1" checker-comment="OK" /></tests></testing-report>
```

ответ должен выглядеть так:

```
"Intel Xeon E3-12xx v2 (Ivy Bridge, IBRS)" 2099.998 2 0
```

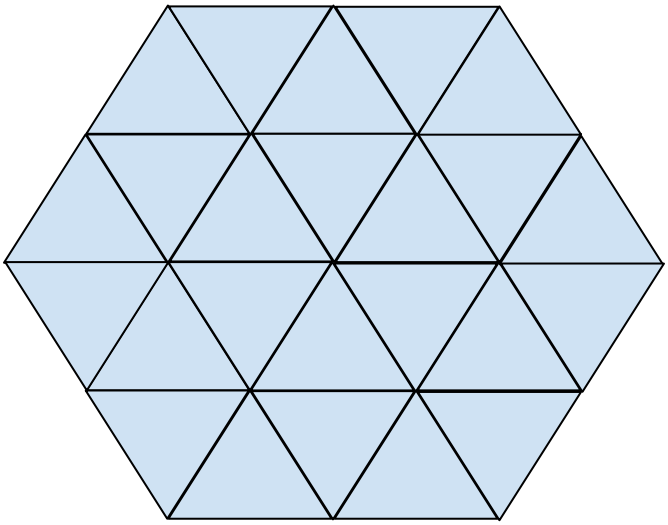
Задача 2.2 CPU-usage - 2

Напишите программу которая решает предыдущую задачу - на стандартный поток ввода передается путь к директории содержащей подмножество не более чем из 100 файлов из архива предыдущей задачи (с сохранением иерархии директорий. Программа на стандартный поток должны вывести ответ в том же формате.

Задача 3. Фиксики

В одном шестиугольном треугольно-клетчатом мире жили фиксики и кработы. Однажды управление кработами сломалось и они начали охотиться за фиксиками. Чтобы защитить фиксиков Дедус решил отгородить всех фиксиков от кработов забором. Ваша задача построить забор минимальной длины, который отделит фиксиков от кработов.

Мир представляет собой шестиугольник со стороной N. Все клетки занумерованы от 1 до 6*N^2, как показано на рисунке.



Забор единичной длины можно построить на границе двух клеток. Кработы могут перемещаться по полю через стороны клеток без забора, но не могут выходить за границы мира.

Формат входных данных
Вводится число 0<N<=50 - размер мира, 0<=M<=50 число кработов, затем M номеров клеток, где они находятся, 0<=K<=50 число фиксиков, затем K номеров клеток,+ где они находятся. Гарантируется, что в одной клетке находится не более одного существа.

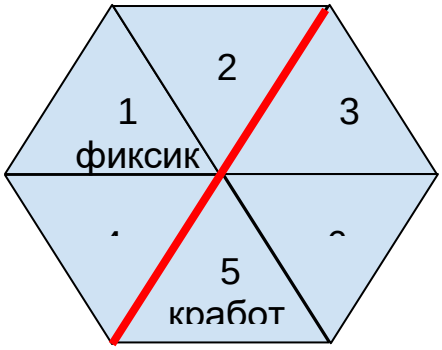
Формат выходных данных
Выведите число стенок W, которые необходимо построить. Затем W пар чисел - номера клеток между которыми строится забор.

Пример
1
2
3 5
1
1

Ответ
2
2 3

4 5

Картинка из примера:



Задача 4. Leader Election

Для обеспечения стабильности в одной стране решили провести выборы нового лидера. Будем считать, что страна представляет собой неориентированный связный граф, в котором соседние вершины могут обмениваться 64-битными сообщениями. Вам требуется написать программу, которая будет запущена одновременно в каждой вершине графа. Выборы представляют из себя несколько шагов, на каждом из которых программа в каждой вершине получает информацию:
* Свой номер - целое 32-битное число
* Число ребер, исходящих из вершины
* Для каждого ребра 64-битное число - сообщение, пришедшее по этому ребру. Число 0 означает, что сообщения не пришло.
В результате выполнения каждая программа в вершине должна вывести на стандартный поток один из вариантом:
* FINISH если она закончила выборы и не является лидером
* LEADER если она закончила выборы и стала лидером
* набор из чисел для каждого ребра, где 0 это отсутствие сообщения, любое другое 64-битное число - сообщение отправляемое соседу, которое будет прочитано соседом на следующем ходе.

Например для графа из двух вершин и одного ребра возможна следующая последовательность:

1 вершина	2 вершина
ввод 1 1 0 вывод 1	ввод 2 1 0 вывод 2
ввод 1 1 2 вывод LEADER	ввод 2 1 1 вывод FINISH

Во время тура будет проверяться только завершаемость процесса выборов за не более чем 50 итераций для теста из примера. Окончательный результат по каждому из тестов будет выставлен после тура в зависимости от числа переданных содержательных сообщений в сети.

Задача 5. Hard disk

Петр очень любит читать книги про трактора и красный цвет. Для этого он сохранил всю свою библиотеку на жесткие диски. Библиотека представляет собой набор файлов с именами из словаря (/usr/share/dict/words). Все файлы не помещаются на один диск, поэтому Петр выбрал N<=10000 файлов для записи на первый диск и т.д.

Петр понимает, что файлов очень много и по имени файла надо понимать на каком диске он лежит. К счастью у каждого диска Петра есть область энергонезависимой памяти в 8192 байт которая может использоваться для оптимизации поиска микроконтроллером диска.

Ваша задача написать программу для записи в эту область данных (режим 1) и программу для использования этой области контроллером диска (режим 2).

В режиме 1 требуется по заданному подмножеству имен файлов сформировать 1024 64-битных числа которые будут записаны в эту память

В режиме 2 по данным из этой области и K именам файлов которые есть у Петра определить есть ли каждый из файлов на жестком диске или нет.

В случае если такого файл есть на диске, а программа микроконтроллера выдает, что его там нет, решение на этом тесте получает 0 баллов. Если программа микроконтроллера говорит что файл есть на диске, а его нет, то приходится обращаться к диску и тратить время. Поэтому ваше решение должно минимизировать число таких ситуаций. Решения будут оцениваться в зависимости от числа ошибок при проверке наличия файла на диске.

Формат входных-выходных данных
В первой строке вводится режим работы

- В режиме 1 вводится число N - количество файлов на диске. В последующих строках вводятся имена файлов из словаря. Выводится 1024 64-битных числа
- В режиме 2 вводится 1024 64-битных числа. Затем вводится число K <=10000 - количество имен файлов которые ищутся на диске. Выводится K строк YES или NO искать ли такой файл на диске или нет

Пример

Ввод	Вывод
1 6 autotractor tractor	1 84758475 323424 .... это неполный вывод

tractoration tractorist tractorization tractorize	
2 1 84758475 323424 .... тут 1024 числа 6 abstractor attractor autotractor cephalotractor extractor tractor	NO NO YES YES NO YES

Задача 6. COVID-19

В одной лаборатории разработан детектор, который определяет количество вирусов в воздухе. Детектор представляет собой матрицу 8x8, через которую проходит воздух в одном из двух направлений: либо сверху-вниз, либо справа-налево. В детекторе находятся 64 сенсора, срабатывающие при прохождении через них вируса, но эти элементы никак не влияют на ток воздуха, то есть вирус, пролетающий через детектор сверху вниз, вызовет срабатывание всех 8 сенсоров, расположенных в соответствующем столбце.

Каждый сенсор может находиться в одном из двух состояний, обозначаемых 0 и 1. При срабатывании сенсора его состояние изменяется на противоположное.

На вход программы подается исходное и конечное состояния матрицы сенсоров. Определите, сколько вирусов пролетело через детектор. Количество вирусов не должно превышать 65536.

Входные данные.

На вход сначала подаются 8 десятичных чисел в диапазоне от 0 до 255, задающих исходное состояние матрицы. Каждое число задает состояние соответствующей строки матрицы: каждый из 8 битов задает состояние сенсора в строке матрицы. Далее подаются 8 десятичных чисел в диапазоне от 0 до 255, задающих конечное состояние матрицы.

На стандартный поток вывода выведите минимальное количество вирусов, пролетевших через детектор, которое бы переводило матрицу из исходного состояния в заключительное.