Задача 1. Точное время

Входной файл: input.txt
Выходной файл: output.txt
Время на тест: 1 секунда

У одного холостяка, жившего в глухой сибирской деревне, были настенные и ручные часы. Однажды ручные часы сломались, а настенные остановились, так как хозяин забыл завести их. Радио, телефонов и мобильников в деревне нет. Умный холостяк, поразмыслив, все же придумал, каким образом выставить на настенных часах точное время. Он решил сходить к своему приятелю, у которого, как он знал, были часы, и перед выходом из дома завел свои. Побыв некоторое время у приятеля, он вернулся домой и установил на своих часах точное время. При этом время, которое требовалось, чтобы дойти от своего дома до приятеля холостяк никогда ранее не измерял, но, будучи человеком педантичным, ходил размеренной походкой, поэтому туда и обратно он прошел за одно и то же время.

Входной файл (input.txt)

В первой строке входного файла ***a*** **–** время выхода холостяка из дома, во второй строке ***b*** **–** время прихода к приятелю, в третьей строке ***c*** **–** время ухода от приятеля, в четвертой строке ***d*** **–** время прихода домой от приятеля. Каждое время задается в формате: hh:mm (hh– часы, mm– минуты, $00 \le hh \le 23$, $00 \le mm \le 59$). При этом между временем ***a*** и временем ***d*** прошло не более суток.

Выходной файл (output.txt)

В первую строку вывести точное время, выставленное холостяком на настенных часах по возвращении домой, в формате: hh:mm.

Пример

input.txt
14:15
13:05
13:25
14:49

output.txt
13:32

Задача 2. Дублирование

Входной файл: input.txt
Выходной файл: output.txt
Время на тест: 2 секунды

Для защиты информации от случайных помех при ее передаче по каналам связи нередко используется защитное дублирование, состоящее в том, что передаваемая информация многократно повторяется. Например, передача числа 505 может быть осуществлена следующим образом: 505505505.

Числа, полученные при повторении других чисел, обладают удобным свойством: их легко можно представить в виде произведения двух сомножителей (для облегчения процесса восстановления информации), таких, что одним из них является число, повторением которого было получено исходное.

Входной файл (input.txt)
В первой строке файла натуральное число N ($1 \le N \le 10^{250}$).

Выходной файл (output.txt)
Начиная с первой строки файла, все пары чисел, произведение которых равно N, причем первое число в каждой паре – это число, повторением которого получено N. Каждую пару чисел выводить отдельной строкой по возрастанию значений первого числа в паре.

Пример
input.txt
6666
output.txt
6 1111
66 101
6666 1

Задача 3. Игровой автомат

Входной файл: input.txt
Выходной файл: output.txt
Время на тест: 2 секунды

Создатели игровых автоматов выпустили на рынок математическую игру. Игра состоит в следующем: в каждом автомате создателями задана уникальная комбинация из M+1 числа в P-ичной системе счисления. Каждое число состоит из N разрядов. Первые M чисел во время игры не меняют своего значения, обозначим их – $a_1, a_2, \ldots a_M$. Во время игры автомат несколько раз случайным образом выбирает из первых M чисел одно число и поразрядно прибавляет его к M+1-му числу (будем называть M+1-е число счетчиком) по модулю P ,тем самым, изменяя его (т.е. счетчик накапливает сумму всех сложений по модулю P). Игрок получает выигрыш, если в результате игры счетчик обнулится.

Поразрядное сложение по модулю P выполняется следующим образом: если в каком-либо разряде числа получено значение большее P-1, то оно уменьшается на P , например, при P= 5 и N=3 результат сложения чисел 123 и 144 равен 212.

Вам прислали на инспекцию несколько таких автоматов, удостоверьтесь в том, что выигрыш принципиально возможен.

Входной файл (input.txt)

Первая строка файла содержит натуральное число – количество автоматов присланных на инспекцию. В следующих строках описываются сами игровые автоматы. Каждый автомат описывается отдельно по формату: первая строка содержит числа P, N, M $(1 \le N, M \le 100,\ 2 \le P \le 255)$. Следующие M +1 строка содержат по N чисел в P –ичной системе счисления, каждое из которых – значение одного разряда P –ичного числа из уникальной комбинации чисел описываемого автомата. Значения разрядов P –ичного числа задаются десятичным числом через пробел.

Выходной файл (output.txt)

Ответ по каждому автомату должен содержаться в отдельной строке. Ответ – это число 0, если игроку вообще не удастся выиграть. Если выигрыш возможен, то ответ – это число 1 и M чисел (через пробел в этой же строке файла) – $k_1, k_2, \ldots k_M$, где значение k_i ($k_i \le P$) указывает, сколько раз нужно прибавить к счетчику число a_i , чтобы в результате всех сложений счетчик обнулился. Если решение не единственно, то вывести любое из них.

Пример

input.txt	**output.txt**
3 4 2 2 2 2 2 2 3 3 3 2 4 1 0 2 0 0 0 0 1 2 1 14 2 2 12 12 10 10 3 3	0 1 1 0 0 2 0

Задача 4. Собачка и другие

Входной файл: input.txt
Выходной файл: output.txt
Время на тест: 1 секунда

Друзья – мальчик, девочка и собачка снова пошли погулять. Они шли по прямой дороге и как всегда мальчик, широко и быстро шагая, шел впереди девочки, постоянно оглядываясь, чтобы обменяться впечатлениями, а собачка с веселым лаем бегала между ними туда и сюда. Добежав до одного из друзей, собачка мгновенно разворачивалась и с той же скоростью бежала в противоположном направлении. Несмотря на хорошую погоду, настроение девочки постепенно ухудшалось, потому что уже в который раз она шла и думала, что скоро настанет тот момент, когда до нее уже не долетят слова мальчика, собачка все реже будет задевать ее своим хвостом и ей придется тащиться по прямой и пыльной дороге практически в одиночку. Разве это прогулка с друзьями?

Ваша задача определить координаты каждого из участников прогулки через заданный промежуток времени, зная текущее положение участников прогулки.

Ограничения

1. Скорости всех участников прогулки постоянны.
2. Координаты участников прогулки отсчитываются относительно начального положения девочки.

Входной файл (input.txt)

В первой строке через пробел шесть вещественных чисел S_1, S_2, V_g, V_b, V_d, t, где
S_1 – расстояние между девочкой и собачкой в начальный момент движения,
S_2 – расстояние между собачкой и мальчиком в начальный момент движения,
$S_1 \geq 0$, $S_2 \geq 0$;
V_g, V_b, V_d – скорости девочки, мальчика и собачки, соответственно, причем
$0 < V_g < V_b < |V_d|$, скорость собачки V_d может быть и отрицательной;
t – время, по прошествии которого нужно определить положение каждого из друзей,
$0.001 \leq t \leq 8$ (в часах),
$1 \leq V_g < V_b < |V_d| \leq 9$ (в км/ч),
$0.01 \leq S_1 + S_2 \leq 1$ (в километрах).

Выходной файл (output.txt)

В первой строке файла три числа: конечные положения девочки, мальчика и собачки соответственно через время t. Числа выдавать с точностью до двух знаков после запятой.

Пример
input.txt
0.5 0.5 1 4 6 0.01

output.txt
0.01 1.04 0.56

Задача 5. Если б я был султан

Входной файл: input.txt
Выходной файл: output.txt
Время на тест: 2 секунды

Жил да был султан, много лет в его темнице томились пленники без всяких надежд на спасение. С годами султан становился мудрее и добрее, пока, наконец, не понял, что каждый пленник лелеет мечту о спасении, да и поднадоели они ему чрезвычайно. Султан решил постепенно выпускать их на свободу. Но султанская должность не позволяла ему открыто показывать свою доброту, и он придумал хитроумный план, чтобы запутать пленников и тем самым лишить необоснованных надежд на скорое спасение. В его темнице было N занятых камер, а в каждой камере по одному пленнику. Двери камер были снабжены запорами, которые могли находиться только в двух положениях: «открыто» и «закрыто». Султан строго следил за своими пленниками, и ни один из них не знал, открыта или закрыта дверь его камеры. Он приказал страже каждый час обходить все камеры. В первый обход стража по приказу султана должна была тайком открыть все замки. При втором обходе изменить на каждой второй двери положения замков на противоположное, при третьем обходе аналогично на каждой третьей двери и т.д. до тех пор, пока стража не совершит N обходов. Узников, которые после всех обходов окажутся в открытых камерах, султан приказал отпустить. Он повелел использовать эту процедуру для всех занятых камер каждый праздник.

Входной файл (input.txt)
Первая строка файла содержит число N ($1 \le N \le 32000$) – количество камер.

Выходной файл (output.txt)
В первой строке файла через пробел в порядке возрастания значений последовательность чисел – номера камер, из которых будут выпущены пленники в ближайший праздник.

Пример
input.txt
7

output.txt
1 4

Задача 6. Мозаика и мастер

Входной файл: input.txt
Выходной файл: output.txt
Время на тест: 3 секунды

Один высококвалифицированный мозаичных дел мастер получил от некоего художника заказ и с энтузиазмом принялся за его выполнение. Ему надо было выложить из стеклянных пластин, имеющих форму ромба, мозаику. Под мозаикой здесь понимается набор ромбов, такой, что любые два ромба либо имеют общую сторону, либо одну общую точку, либо не пересекаются. При этом ромбы, имеющие общую сторону, называются соседними. Художник поставил мастеру свои условия: а) использовать пластины только трех его любимых цветов, б) располагать пластины так, чтобы пластины, имеющие общие стороны, имели разный цвет, в) форма мозаики изначально задана, г) может быть изначально определен цвет некоторых ромбов. Через некоторое время энтузиазм мастера начал угасать, так как оказалось, что эта работа требует от него дополнительных усилий и не всегда получается вариант, соответствующий условиям художника. Попробуйте помочь мастеру.

Входной файл (input.txt)

В первой строке файла число N – количество пластин-ромбов в мозаике ($1 < N < 1525$). Начиная со второй строки файла, в каждой строке содержится информация о мозаике по следующему формату: K_i – номер пластины мозаики ($1 \le i \le N$), затем через пробел Z_i – ее цвет, далее через пробел номера пластин соседних с K_i . Z_i может принимать значение одного из четырех символов: *a, b, c* или *z* ; символы *a, b, c* задают один из трех возможных цветов, зафиксированных художником; символ *z* – означает, что художник не задал цвет этой пластины.

Выходной файл (output.txt)

Если удалось создать мозаику, то в первую строку файла выводить информацию о мозаике по формату: K_i пробел Z_i пробел для всех $1 \le i \le N$ в порядке возрастания номеров пластин (при этом, Z_i не может принимать значение символа *z*). Если не удалось создать мозаику, то в первую строку вывести *no*. Если решение не единственно, то вывести любое из них.

Пример
input.txt

```
4
1 a 2 3
3 z 1 4
4 a 2 3
2 z 1 4
```

output.txt

```
1 a 2 b 3 b 4 a
```

Задача 7. Конструктор

Входной файл: input.txt
Выходной файл: output.txt
Время на тест: 3 секунды

Довольно часто приходится встречаться с задачей распознавания цепочек в тексте. Её невозможно, например, обойти в программах, которые каким-либо образом интерпретируют или обрабатывают тексты.

Рассмотрим специальное множество таких цепочек, называемое регулярным множеством. Регулярные множества порождаются с помощью регулярных выражений (все необходимые определения даны в приложении).

Ваша задача: написать программу, которая для заданного регулярного выражения A, определяющего множество предложений над словарём V(назовём это множество A'), формирует по заданным правилам программу, определяющую принадлежность цепочки символов множеству A'. При этом будем полагать, что множество V – множество строчных латинских букв.

Опишем схему, следуя которой по заданному выражению можно конструировать алгоритм анализа.

Правила конструирования:

P(X) обозначает схему программы, соответствующую регулярному выражению X, переменная ch всегда содержит очередной сканируемый символ.

1. P(s) = if (ch='s') then read(ch) else error;
2. P(AB) = P(A) P(B)
3. P(A+B) = if (ch='s_1') or (ch='s_2') or …or (ch='s_n') then begin P(A) end else begin P(B) end;
4. P(A*) = while (ch = 's_1') or (ch='s_2') or …or (ch='s_n') do begin P(A) end;

Символы s_1 ,$s_{2....}$,s_n – это символы, с которых могут начинаться предложения из A', причем $s_1 < s_2 <_{....} < s_n$.

Заметим, что эта схема применима лишь тогда, когда выражение детерминировано, т.е., если выражение не содержит составляющих вида A+B, таких, что A и B начинаются с одинаковых символов. Любой символ *s*, с которого начинается предложение из A', называется его начальным символом.

Чтобы получить правдоподобную программу, соответствующую выражению A, схему P(A) нужно поместить в «рамку»:
Var ch: char; begin read(ch); P(A) if (ch<>'.') then error; end.

Ограничения.

Выражения не могут содержать фрагменты вида A+B+C. При необходимости задания выражений такого типа для определения приоритета используются скобки. Например, (A+B)+C и A+(B+C).

Входные данные (input.txt):

В первой строке входного файла записана одна строка, которая задаёт регулярное выражение. Длина строки меньше 110.

Выходные данные (output.txt):

В выходной файл записывается текст сконструированной программы-анализатора без пробелов.

Пример
input.txt
a((b+c)a)*
output.txt
Varch:char;beginread(ch);if(ch='a')thenread(ch)elseerror;while(ch='b')or(ch='c')dobeginif(ch='b')thenbeginif(ch='b')thenread(ch)elseerror;endelsebeginif(ch='c')thenread(ch)elseerror;end;if(ch='a')thenread(ch)elseerror;end;if(ch<>'.')thenerror;end.

Приложение:

Введём следующие обозначения:

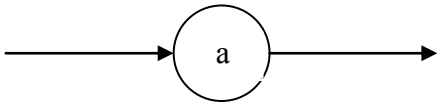
1. Строчными латинскими буквами обозначаются символы основного словаря V. Все предложения являются последовательностями символов из V.
2. Греческими буквами обозначаются последовательности символов, определённые над словарём V. Буквой ε обозначается пустая последовательность.
3. Прописными буквами обозначаются регулярные выражения или множество предложений, определённых с помощью регулярных выражений.
4. Множество последовательностей символов, получаемых конкатенацией предложений из A и предложений из B, называется произведением A и B.

   $AB=\{\alpha\beta;\ \alpha\in A \text{ и } \beta\in B\}$
5. Объединение A и B обозначается как $A+B = \{\gamma;\ \gamma\in A \text{ или } \gamma\in B\}$
6. Множество предложений, получаемых путём произвольного числа конкатенаций предложений из A, обозначается как A*

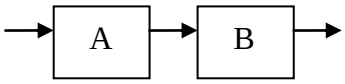
   $$A^*= \{ \varepsilon +A+AA+AAA+\ldots+A^n\}.$$

Правила построения регулярных выражений и их графическая интерпретация:

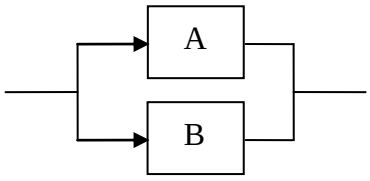
1. Любой основной символ $a \in V$ есть регулярное выражение



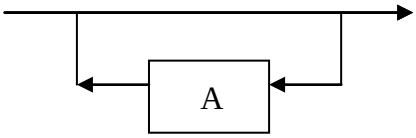
2. Любое произведение двух регулярных выражений есть регулярное выражение AB



3. Любое объединение двух регулярных выражений есть регулярное выражение A+B



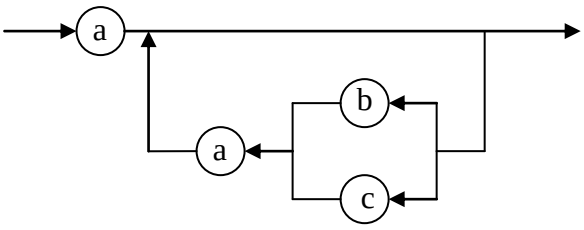
4. Если A – регулярное выражение, то A* – регулярное выражение



5. Регулярны только те выражения, которые получены с помощью правил 1-4.

Пример

Регулярное выражение: a((b+c)a)* и его графическая интерпретация:



Задача 8. Загрузка контейнера

Входной файл: input.txt
Выходной файл: output.txt
Время на тест: 1 секунда

Контейнер, имеющий форму прямоугольного параллелепипеда, необходимо загрузить ящиками. Все ящики имеют кубическую форму и одинаковые размеры, так что ребро куба можно принять за единицу длины.

Содержимое ящиков имеет разный вес и степень хрупкости, в соответствии с которыми каждому ящику присвоен целочисленный индекс из интервала от 1 до 9.

Правила загрузки разрешают ставить несколько ящиков друг на друга в том случае, если каждый ящик несет на себе груз, суммарный индекс которого меньше собственного индекса этого ящика. Так, поставить „1 на 3 на 7" можно, потому что ящик „7" несет на себе груз суммарного индекса 4, а ящик „3" – суммарного индекса 1. А вот поставить „1 на 2 на 3 на 8" нельзя, так как ящик „3" несет на себе груз, суммарный индекс которого равен собственному индексу.

Требуется определить, можно ли загрузить в контейнер указанный набор ящиков без нарушения правил.

Входной файл (input.txt)

Первая строка файла содержит три целых числа L, N, H, определяющих вместимость контейнера в порядке: L – длина основания, N – ширина основания и H – высота
($0 \le L, N, H \le 40000$).

Вторая строка содержит 9 чисел, описывающих набор ящиков в следующем порядке: K_1 – число ящиков индекса 1, K_2 – число ящиков индекса 2 и т. д. до K_9 – числа ящиков индекса 9, причем $\sum_{i=1}^{9} K_i \le 2 \cdot 10^9$.

Выходной файл (output.txt)

В первой строке файла одно число. Если загрузка возможна, то 1, в противном случае 0.

Пример
input.txt
3 4 3
15 1 0 0 0 0 0 0 0
output.txt
0

Задача 9. Арифметические выражения

Входной файл: input.txt
Выходной файл: output.txt
Время на тест: 3 секунды

Выпускная работа Васи Пупкина в школе развития связана с разработкой обучающей системы для младших школьников. Вася с жаром взялся за работу. Он решил, что в первую очередь необходимо разработать модуль, обучающий школьников вычислению арифметических выражений. Он разработал проект этой программы и понял, что одна из основных его задач – это обновление арифметических выражений, которые должны вычислять обучающиеся. Ему пришла идея разделить все арифметические выражения по уровням сложности. При этом в качестве меры сложности Вася взял длину арифметического выражения. Множество арифметических выражений заданной длины можно расположить в лексикографическом порядке и случайно выбирать арифметическое выражение по его порядковому номеру.

Очевидно, что у Васи работы непочатый край, а Вы могли бы ему помочь. Напишите программу, реализующую идею Васи о генерации арифметических выражений по заданному порядковому номеру.

Определим арифметическое выражение четырьмя правилами:

1. x – это арифметическое выражение,
2. y – это арифметическое выражение,
3. если A и B – это арифметические выражения, то арифметическими выражениями будут:
   (A)
   (B)
   A + B
   A*B
4. выражениями считаются только те, которые получены по правилам 1 – 3.

Определим лексикографический порядок для символов, используемых в выражениях определённых выше, следующим образом: ‘x’ < ‘y’ < ‘(‘ < ‘)’ < ‘*’ < ‘+’. Занумеруем все правильные записи арифметических выражений заданной длины в соответствии с лексикографическим порядком.

Напишите программу, которая по номеру арифметического выражения в заданном упорядоченном множестве арифметических выражений данной длины выдаёт арифметическое выражение, соответствующее данному номеру.

Входной файл (input.txt)

В первой строке файла записаны через пробел два целых числа N и M $(1 \le N \le 21)$, $(1 \le M \le 2*10^9)$ –длина арифметического выражения в множестве и его номер соответственно.

Выходной файл (output.txt)

В выходной файл нужно вывести арифметическое выражение, которому сопоставлен номер N в упорядоченном множестве выражений длины M. Если выражение с заданным номером не существует, то вывести 0.

Пример 1
input.txt
3 5
output.txt
y*x
Пример 2
input.txt
3 10
output.txt
(y)

Задача 10. Слова

Входной файл: input.txt
Выходной файл: output.txt
Время на тест: 2 секунды

Следующим в обучающей системе Вася решил сделать модуль, развивающий у школьников внимание. Попутно он решил учить ребят грамотно писать трудные слова. Для этого Вася написал программу, которая размещала в клеточном поле различными способами некоторое изучаемое слово. А ученик должен был прочитать это слово по определённым правилам и подсчитать, сколькими способами это можно сделать. Для проверки правильности ответов школьников Васе необходимо написать программу. И он опять обращается к Вам за помощью – напишите программу, подсчитывающую, сколькими способами можно прочитать заданное слово.

Правила чтения слова в клеточном поле:

1. позиция, с которой начинается чтение слова, определена, обозначим её (i,j).
2. следующую букву заданного слова можно прочитать в позициях: (i,j+1) или (i,j-1) или (i+1,j) или (i+1,j+1) или (i+1,j-1).

Входной файл (input.txt)

Во входном файле в первой строке записано слово, длина слова не превосходит 25 символов. Во второй строке записано два целых числа N и M ($2 \le N \le 25$) , ($2 \le M \le 50$) – число строк и столбцов клеточного поля, соотвественно. В третьей строке два целых числа I и J ($1 \le I \le N$, $1 \le J \le M$) – координаты позиции, с которой начинается чтение слова, далее *N* строк клеточного поля, заполненного буквами.

Выходной файл (output.txt)

В выходной файл нужно вывести одно целое число – количество способов.

Пример 1
input.txt
подобие
4 7
1 1
подобие
подобие
подобие
подобие
output.txt
42
Пример 2
input.txt
подобие
4 6
1 3
азпдби
годаие
оободба
бобиее
output.txt
6