# Задача 1. Простая арифметика

**Время на тест: 2сек.**

Детей учат складывать многоразрядные числа справа налево, по одной цифре за один раз. Многие дети считают операцию «переноса», когда 1 переносится в следующий разряд, достаточно сложной. Ваша работа состоит в том, чтобы сосчитать число операций переноса для каждой задачи на сложение (с целью оценки ее сложности). Обобщим задачу и решим её для любых P – ичных чисел.

**Входной файл (input.txt)**

В первой строке вводится одно целое число P, определяющее систему счисления ($2 \le P \le$ **62)**. Во второй строке целое число $L_1$ ($1 \le L_1 \le$ **62000)** – длина первого слагаемого (в символах). Третья строка – первое слагаемое. Четвёртая строка содержит целое число $L_2$ ($1 \le L_2 \le$ **62000)** – длину второго слагаемого. Пятая строка – второе слагаемое.

Цифры для оснований >10 — большие латинские буквы (A-Z);
для оснований >36 — малые латинские буквы (a-z).

**Выходной файл (output.txt)**

Выведите одно целое число – количество операций переноса, которое необходимо выполнить при сложении двух чисел, заданных в P-ичной системе.

**Пример1:**

| **input.txt** | **output.txt** |
|---|---|
| 10<br>3<br>123<br>3<br>456 | 0 |

**Пример2:**

| **input.txt** | **output.txt** |
|---|---|
| 8<br>3<br>123<br>3<br>456 | 2 |

# Задача 2. Развёртка октаэдра

**Время на тест: 2 сек.**

Дана развертка некоторой фигуры (рис.1), представленная в виде смежных, равносторонних треугольников. Определите, можно ли свернуть эту развертку в октаэдр? Наложение нескольких граней друг на друга допускается. (Октаэдр – правильный восьмигранник с гранями, являющимися правильными треугольниками).

**Входной файл (input.txt)**

Все треугольники пронумерованы от 1 до N. В первой строке вводится одно целое число N – количество заштрихованных треугольников, $4 \le N \le 1000$.

В последующих N строках описывается заштрихованная фигура по следующему формату: в каждой i-й строке по часовой стрелке указаны номера соседних с i-м треугольников. Если соседнего треугольника нет — цифра 0.

**Выходной файл (output.txt)**

Yes, если заштрихованная фигура является развёрткой октаэдра. No, если заштрихованная фигура не является развёрткой октаэдра.

**Пример:**

| **input.txt** | **output.txt** |
|---|---|
| 9<br>0 2 0<br>1 0 3<br>2 0 4<br>0 3 5<br>4 6 0<br>5 9 7<br>6 0 8<br>0 7 0<br>0 6 0 | Yes |

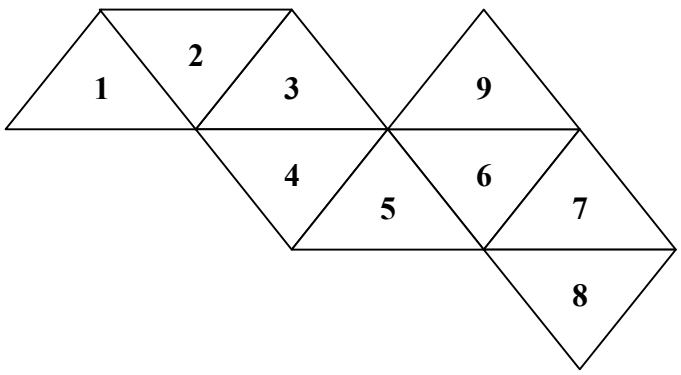


Рис.1

# Задача 3. Слова на кубе

**Время на тест: 2сек.**

На заданной сетке с квадратными ячейками разместили развёртку куба, каждая грань которого разбита на квадраты, размер которых, в свою очередь, совпадает с размером ячейки сетки. В каждом квадратике куба записана буква.

По заданной развёртке куба и списку слов необходимо определить позиции в сетке, в которых находятся первые буквы заданных слов. Если заданное слово записано на кубе многократно, то вывести координаты того слова, первая буква которого находится, во-первых, левее всех на развертке и, во-вторых, выше.

На кубе слово может размещаться в любом направлении по периметру сечения куба, параллельного одной из граней куба.

**Пояснение:** В пределах одной грани куба слово может располагаться только по прямой непрерывной линии. Слова могут быть расположены на смежных гранях куба, поэтому на развёртке слова могут быть разорваны, если они находятся на разных гранях куба.

**Входной файл (input.txt)**

Первая строка содержит два целых числа: M – число строк сетки и N – число столбцов сетки ($3 \le M, N \le 25$) . В следующей строке одно целое число P ($1 \le P \le 10$) – количество сторон квадратов на одном ребре куба. Далее M строк описывающих развертку. Строка содержит N символов записанных через пробел. Пустые клетки сетки содержат символ ‘*’, остальные содержат буквы, при этом строчные и прописные буквы не различаются. После описания сетки вводится целое K ($1 \le K \le 20$) – количество распознаваемых слов. С новой строки вводятся слова, по одному в каждой строке. Длина слов не превышает 15 символов.

**Выходной файл (output.txt)**

В каждой строке выходного файла записывается два целых числа – координаты начала слова: номер строки сетки и номер столбца сетки, разделённых пробелом. Количество выходных строк равно количеству разыскиваемых слов.

**Пример**

**input.txt**

```
24 18
6
* * * * * * E S z d a g * * * * * *
* * * * * * b e r a a f * * * * * *
* * * * * * g V H g k r * * * * * *
* * * * * * a m r b t J * * * * * *
* * * * * * D n c s F S * * * * * *
* * * * * * m A Q w e r * * * * * *
* * * * * * B E T Y W e * * * * * *
* * * * * * A S d g j l * * * * * *
* * * * * * b B e t t y * * * * * *
* * * * * * z x E r T y * * * * * *
* * * * * * d A R Y U X * * * * * *
* * * * * * A Q W E D g * * * * * *
E S z d a g z a b b e r a a F g D a
A b e r a a f D a E S z d a G h A D
m A Q w e r E m M q H a d r W k G m
m A Q w e r b G b a p y t a L a B a
E S z d a g g r I D o y l s H D S e
b e r a a f a a w m v D i j b m D h
* * * * * * D D a c o o * * * * * *
* * * * * * m m d R f t * * * * * *
* * * * * * B d f w b a * * * * * *
* * * * * * E S z d a g * * * * * *
* * * * * * t a a a m w * * * * * *
* * * * * * r q D r e a * * * * * *
3
Bambi
Betty
Dagbert
```

**output.txt**

```
13 9
9 8
5 7
```

# Задача 4. Проектирование

**Время на тест: 2 сек.**

При проектировании некоторого прибора на плоской, квадратной плате, расчерченной на квадраты, размещены элементы, которые должны быть соединены проводами по определённым правилам:

- все элементы, размещённые на плате пронумерованы от 1 до M, где M – количество деталей, монтируемых на плате.
- детали соединяются в порядке следования номеров
- провода, соединяющие детали, должны проходит через середину горизонтальных и вертикальных клеток без самопересечений
- через любую клетку платы может проходить только один провод

| | | | |
|---|---|---|---|
| | 4 | | 1 |
| | | | |
| | | 3 | |
| 5 | | | 2 |


**Входной файл ( input.txt)**

В первой строке вводятся два целых числа N и M через пробел, где N – размер платы (2<=N<=7), а M – количество деталей на плате (2<=M<=N*N). Далее построчно вводится квадратная таблица, которая определяет положение деталей.

**Выходной файл (output.txt)**

Если соединить детали по перечисленным выше правилам возможно, то в первую строку выведите слово **«Solution»** и начиная со второй строки перечислите координаты клеток, через которые проходит соединяющий провод. Последовательность координат завершается двумя нулями. Если при данном размещении деталей соединение невозможно, то вывести единственное **«No solution»**.

**Пример:**

**input.txt**

```
4 5
0 4 0 1
0 0 0 0
0 0 3 0
5 0 0 2
```

**output.txt**

```
Solution
1 4
2 4
3 4
4 4
4 3
4 2
3 2
3 3
2 3
2 2
1 2
1 1
2 1
3 1
4 1
0 0
```

# Задача 5. Загрузка парома

**Время на тест: 2сек.**

Ордынский паром перевозит машины через Обское «море» с одного берега на другой на платформе. Платформа устроена так, что машины заезжают на него с одного конца, а съезжают с другого конца. Машины устанавливаются на платформу в два ряда, параллельных борту. Имеется одна очередь из машин, заезжающих на паром. Капитан и палубный матрос распределяют машины на платформе по рядам так, чтобы нагрузка была сбалансирована. Все машины в очереди имеют разную длину. Опытный капитан, видя всю очередь и зная длины всех машин при загрузке парома, определяет в какой ряд поставить очередную машину из очереди. При этом он стремится к тому, чтобы вошло максимальное количество машин, и платформа была загружена как можно более равномерно (под равномерностью загрузки понимается минимальная разница загрузки между левым и правым бортом по оставшемуся свободному месту). Напишите программу, которая распределяет машины на платформе с выше изложенными правилами.

**Входной файл (input.txt)**

В первой строке входного файла вводится целое число N ($1 \leq N \leq 100$) – длина платформы в метрах. Для каждой машины в очереди имеется своя строка в файле, которая содержит одно целое число M ($100 \leq M \leq 3000$) – длину машины в сантиметрах. Последняя строка входных данных содержит 0

**Выходной файл (output.txt)**

В первой строке выходного файла вывести одно целое число – количество машин загруженных на платформу парома. Для каждой машины, которая может быть загружена на платформу, в порядке их следования во входном файле выведите строку, содержащую «left», если машина должна быть направлена на левую сторону, «right», если машина должна быть направлена на правую сторону. Если поставленная задача имеет несколько вариантов решения, подойдёт любой вариант.

**Пример:**

| **input.txt** | **output.txt** |
|---|---|
| 50 | 6 |
| 2500 | right |
| 3000 | left |
| 1000 | left |
| 1000 | right |
| 1500 | right |
| 700 | left |
| 800 | |
| 0 | |

## Задача 6. Эффективный процент владения

**Время на тест: 2 сек.**

Как-то за чаепитием в клубе группа инвесторов обнаружила, что все они совершенно запутались, какой частью акций известных компаний владеет каждый участник чаепития – ведь если компания «А» владеет 40% акций компании «Б», то инвестор компании «А» за каждый процент акций этой компании фактически владеет 0.4% акций компании «Б»… Поэтому, они наняли Вас, чтобы вы раз и на всегда решили споры, составив точные объемы владений для каждого инвестора.

**Входной файл (input.txt)**

Входной файл содержит два числа M и N – число инвесторов и число компаний ($0<M<10$, $0<N<30$)

Затем идёт M строк, каждая строка содержит N чисел. Каждое j-е число (нумеруя от 0 до N-1) i-й строки (нумеруя от 0 до M-1) содержит процент владения инвестором i акциями компании j.

После идёт N строк по N чисел, где каждое j-е число (нумеруя от 0 до N-1) k-й строки (нумеруя от 0 до N-1) содержит процент владения компании k акциями компании j.

**Выходной файл (output.txt)**

Выходной файл содержит M строк по N чисел. Каждая i-я строка содержит эффективные проценты владения акциями известных компаний i-м инвестором. Порядок нумерации компаний и инвесторов такой же, как во входном файле. Данные выводятся округлёнными до двух знаков после запятой. Одна величина внутри одной строки отделяется от другой пробелом.

**Пример**

| **input.txt** | **output.txt** |
|---|---|
| 1<br>2<br>70 0<br>0 20<br>70 0 | 81.40 56.98 |

# Задача 7. Паутина

**Время на тест: 2 сек.**

Как известно, пауки большие мастера плести паутину. В заповедном лесу жил да был такой паук и плёл он паутину по-особому. Его паутина - это множество окружностей различного радиуса и набор отрезков (см. рис.1). Причем, начало и конец отрезков обязательно лежат на окружностях. Сплетёт он паутину, сам спрячется и поджидает жертву – муху или комара, или ещё кого-нибудь. Как только жертва запутается в паутине, паук тут как тут. Не даёт бедняжке шанса на спасение. Всегда добирается до мухи кратчайшим путём. Ваша задача – определить длину этого кратчайшего пути.

Предполагаем, что центр паутины совпадает с началом координат.

**Входной файл (input.txt)**

В первой строке входного файла содержится целое число M - число окружностей. ($0 < M \le 100$)

Затем идёт M строк. Каждая строка содержит одно число - радиус окружности $0 < R_i < 10^3$.

Далее – строка, содержащая целое число N - число отрезков. ($0 < N < 10^6$)

После идёт N строк по 4 вещественных числа в каждой строке – $(r,\phi)$ - полярные координаты отрезка (координаты всегда лежат на окружностях), причем $r > 0, \phi \in \left(-10^6; 10^6\right)$, угол измеряется в радианах.

Предпоследняя строка содержит два числа $(r,\phi)$ - полярные координаты паука.

Последняя строка содержит два числа $(r,\phi)$ - полярные координаты жертвы.

**Выходной файл (output.txt)**

В единственную строку выходного файла выводится одно число – длина пути от паука до жертвы. Результат выводится с округлением до 2х знаков после запятой. В случае, если пути от паука до жертвы нет – вывести -1.

**Пример.**

input.txt

```
5
1
2
3
4
5
12
1.00000 0.00000 2.00000 0.00000
1.00000 2.09440 2.00000 2.09440
1.00000 4.18879 2.00000 4.18879
2.00000 0.20000 3.00000 0.20000
2.00000 2.29440 3.00000 2.29440
2.00000 4.38879 3.00000 4.38879
3.00000 0.40000 4.00000 0.40000
3.00000 2.49440 4.00000 2.49440
3.00000 4.58879 4.00000 4.58879
4.00000 0.60000 5.00000 0.60000
4.00000 2.69440 5.00000 2.69440
4.00000 4.78879 5.00000 4.78879
5.00000 1.57080
1.00000 3.14159
```

output.txt

```
12.47
```

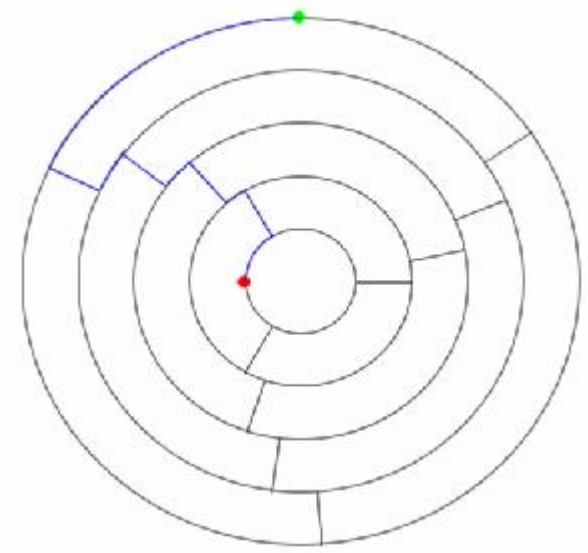

Рис. 1

## Задача 8. Мушкетёры за круглым столом

**Время на тест: 3 сек.**

Друзья-мушкетёры за круглым столом
Песню надумали спеть о былом.
Вспыхнули давние споры опять:
Кто из них главный, кому запевать?

Хитрый Атос предлагает: «Чем драться,
Лучше гораздо, друзья, посчитаться.
Я и считалку уже изобрёл:
Эники-беники-прячься-под-стол!»

Как вы, конечно, смогли догадаться,
Хочет он сам запевалой остаться.
Будем внимательны: тонкость вопроса
В том, чтоб под стол не отправить Атоса…

Как пользоваться считалками, все знают с детства: начинается счёт, и тот, на кого выпало последнее слово, выбывает (в данном случае, согласно считалке, ему придётся лезть под стол и сидеть там до окончания счёта). Далее счёт продолжается со следующего за ним человека в том же направлении, выбывает ещё кто-то, и так до тех пор, пока из всей компании не останется один человек.

Задумка Атоса вполне очевидна: как автор идеи, он сам и будет считать, так что ему нужно (как бы невзначай) начать счёт не откуда попало, а так, чтобы последним остался он сам. Например, нетрудно проверить, что если за столом сидят четыре мушкетёра, то Атосу, считая по часовой стрелке, нужно начинать с того, кто сидит от него по правую руку.

**Входной файл (input.txt)**
Во входном файле два целых числа: количество мушкетёров за столом $2<N<=50000$ и количество слов в считалке $1<K<50000$.

**Выходной файл (output.txt)**
В выходном файле одно целое число от 1 до N, указывающее, с кого нужно начинать счёт. Единица соответствует самому Атосу, двойка — тому, кто сидит от него по левую руку, и так далее по часовой стрелке. Число N соответствует мушкетёру, сидящему справа от Атоса. Предполагается, что счёт ведётся также по часовой стрелке (т.е., по возрастанию номеров)

**Пример:**
**Input.txt:**
4 5
**Output.txt:**
4

# Задача 9. Из гвардейцев в мушкетёры

**Время на тест: 2 сек.**

Гвардейцы кардинала, известные своими буйствами и бесчинствами, учинили очередную антиобщественную выходку, и терпение короля истощилось. Он приказал кардиналу передать виновников из роты гвардейцев под командование капитана мушкетёров (дескать, там они будут находиться под хорошим влиянием и, возможно, исправятся).

Кардинал отнюдь не пришёл в восторг, но ослушаться королевского приказа он не мог. С другой стороны, он не был бы кардиналом, если бы не сумел придумать по этому случаю какую-нибудь небольшую каверзу.

Вызвав к себе виновников происшествия (их было шестеро), он приказал им следующее: явиться к капитану мушкетёров, но не представляться ему, а вручить кардинальское письмо и построиться по росту. Бывшие гвардейцы в точности исполнили приказ.

Распечатав письмо, капитан мушкетёров прочёл:

> «В строю ***E*** стоит где-то слева от ***B***. Справа от ***A*** в строю нет ***F***. Где-то слева от ***D*** стоит ***C***. Вторую позицию занимает ***A***. ***E*** и ***D*** стоят в строю подряд друг за другом. Вот ваше пополнение, господин капитан, а кто из них кто — извольте разбираться сами!»

(Разумеется, вместо букв ***A***,***B***,***C***,***D***,***E***,***F*** в кардинальском письме были указаны настоящие имена и титулы, но для нас они за давностью лет совершенно не важны.)

Мысленно выругав кардинала и поразмыслив некоторое время, капитан всё же догадался, что бывшие гвардейцы стоят перед ним в следующем порядке: F,A,C,E,D,B.

Необходимо разработать программу, которая по списку условий определяет порядок расстановки в строю.

**Входной файл(input.txt)**

В первой строке файла два числа: количество гвардейцев *n* и число условий расстановки *k*. В каждой из последующих *k* строк записано по одному условию. Условия кодируются следующим образом:

Запись «`E<B`» означает «***E*** стоит левее ***B***» (при этом между ними может стоять кто-то ещё). Аналогично, «`C<D`» означает, «***C*** стоит левее ***D***».

Запись «`A<f`» означает «справа от ***A*** в строю нету ***F***» (обратите внимание, что в данном случае используется символ не «`F`», а «`f`». Аналогично, запись «`a<F`», означала бы «слева от ***F*** в строю нету ***A***»).

Запись «`A=2`» означает «***A*** стоит в строю вторым».

Запись «`E,D`» означает «***E*** и ***D*** стоят в строю подряд: сначала ***E***, потом ***D***».

Гарантируется, что набор правил описывает один и только один возможный порядок расстановки.

**Выходной файл(output.txt)**

В первой строке файла должны быть записаны первые *n* букв латинского алфавита *в верхнем регистре*. Порядок записи соответствует порядку расстановки гвардейцев в строю.

**Пример.**

input.txt

```
6 5
E<B
A<f
C<D
A=2
E,D
```

output.txt

```
FACEDB
```