## **Задача A.** Малый бизнес

*Исходный файл: input.txt*
*Выходной файл: output.txt*
*Время тестирования: 2 секунды*

Выставка компьютерных фирм проходит в зале, разделенном на $N \times N$ павильонов. Каждая из четырех стен любого павильона, кроме граничных стен, имеет дверь в соседний павильон. Каждый павильон занят какой-либо фирмой, которая раздает посетителям предметы какого-либо одного вида (ручки, пакеты, проспекты и т.д.). Начинающий компьютерный бизнесмен Вася желает набрать бесплатных предметов на возможно большую сумму. К сожалению, в одни руки выдают только по одному предмету за одно посещение, поэтому Вася вынужден постоянно переходить из павильона в павильон. Посещать один и тот же павильон можно сколько угодно раз. На получение одного предмета Васе требуется одна минута. Требуется определить максимальную сумму, на которую Вася может набрать предметов в течение $K$ минут. Зал определяется квадратной матрицей, содержащей стоимость предметов, выдаваемых в каждом павильоне. $A = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1N} \\ a_{21} & a_{22} & \dots & a_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ a_{N1} & a_{N2} & \dots & a_{NN} \end{pmatrix}$, $2 \le N \le 100$, $0 < a_{ij} < 10\,000$, $a_{ij} \in Z$

Путь Васи всегда начинается с павильона (1,1) и состоит из последовательности пар координат $(i_1, j_1)$, $(i_2, j_2)$,… ,$(i_K, j_K)$, $1 \le K \le 10\,000$, в которой для любого $t$ $1 \le t < K$ $1 \le i_t, j_t \le N$ и $|i_t - i_{t+1}| + |j_t - j_{t+1}| = 1$. По данным $N$, $K$ и матрице $A$ найти $(i_t, j_t)$, $t=1..K$ такие, что $i_1 = j_1 = 1$ и сумма

$$\sum_{t=1}^{K} a_{i_t j_t}$$ максимальна.

***Входные и выходные данные***

Первая строка входного файла содержит числа $N$ и $K$. Следующие $N$ строк представляют $1,2,\dots,N$-ю строки матрицы, по $N$ чисел в строке.

Выходной файл должен содержать единственное число — максимальную сумму.

***Пример файла input.txt:***

```
5 7
1 1 1 1 1
1 1 3 1 9
1 1 6 1 1
1 1 3 1 1
1 1 1 1 1
```

***Соответствующий файл output.txt:***

```
21
```

## **Задача B.** ЛОГО-равенство

*Исходный файл: input.txt*
*Выходной файл: output.txt*
*Время тестирования: 2 секунды*

Программа, записанная на подмножестве языка ЛОГО, состоит из последовательности операторов, расположенных по одному в строке. Операторы управляют перемещением ориентированного пера (“черепахи”) по виртуальному графическому экрану, состоящему из пикселей. Имеются следующие команды:

- `FORWARD n` / `BACK n` — перемещение вперед/назад на `n` пикселей;
- `LEFT a` / `RIGHT a` — поворот влево/вправо на `a` градусов;
- `PENDOWN` / `PENUP` — опустить/поднять перо (оставлять/не оставлять след при движении), сама по себе команда `PENDOWN` не оставляет следа;
- `FINISH` — конец программы.

Аргумент `n` — целое число пикселей от 0 до 10 000, аргумент `a` — один из углов 0, 45, 90, 135, 180, 225, 270, 315, 360 градусов (т.е. кратен 45 градусам). В начальном состоянии перо опущено и ориентированно одинаково для обеих программ. В результате работы программа на ЛОГО рисует на экране некоторую фигуру.

Например, программа

```
BACK 20
RIGHT 45
FORWARD 20
LEFT 135
FORWARD 20
FINISH
```

Нарисует прямоугольный треугольник.

Требуется по двум данным программам на ЛОГО определить, совпадают ли выводимые ими фигуры с точностью до параллельного переноса. Пустые фигуры (не содержащие ни одного пикселя) считаются совпадающими.

***Входные и выходные данные***

Во входном файле одна за другой расположены две ЛОГО-программы длиной не более 1000 шагов каждая. Каждая из них заканчивается оператором FINISH. Операторы записаны по одному в строке. Программы могут содержать произвольное количество незначащих пробелов и пустых строк. Выходной файл должен содержать либо строку “YES”, либо “NO”, в зависимости от результата решения задачи.

***Пример файла input.txt:***

```
FORWARD 100
BACK 50
LEFT 90
FORWARD 50
FINISH
RIGHT 90
BACK 50
LEFT 45
PENUP
```

***(продолжение input.txt):***

```
FORWARD 50
LEFT 45
PENDOWN
BACK 100
FINISH
```

***Соответствующий файл output.txt:***

```
YES
```

## **Задача C.** Марковский цикл

*Исходный файл: input.txt*
*Выходной файл: output.txt*
*Время тестирования: 2 секунды*

Ограниченный алгоритм Маркова состоит из последовательности предложений
$s_1s_2...s_N$ -> $d_1d_2...d_N$
Где $s_i$ и $d_i$ — символы из алфавита “A”, “B”, “C”, $1 \le N \le 12$. Подстрока $s_1s_2...s_N$ называется левой частью, а $d_1d_2...d_N$ — правой частью предложения.
Алгоритм выполняется над исходной текстовой строкой, состоящей из заглавных латинских букв “A”, “B”, “C”, следующим образом: перебираются все предложения, начиная с первого. Если левая часть предложения входит в текстовую строку, то самое левое вхождение заменяется на правую часть этого предложения, и поиск вновь начинается с первого предложения. Если ни одно предложение не может быть применено, алгоритм останавливается.
При выполнении алгоритма возможны два результата: либо остановка, либо бесконечный цикл с определенным периодом. По данной строке и набору предложений алгоритма Маркова определить количество “ациклических” (выполненных до начала цикла) шагов и длину самого цикла. Если алгоритм останавливается, то длина цикла считается нулевой, а все выполненные шаги — ациклическими.

***Входные и выходные данные***
В первой строке входного файла находится исходная текстовая строка длиной от 1 до 12 букв, а в следующих строках — не более 50 предложений, по одному предложению в строке. Предложения могут содержать произвольное количество незначащих пробелов.
Выходной файл должен содержать два целых числа — количество ациклических шагов и длину цикла.

***Пример файла input.txt:***
```
ABABC
C  ->A
AB ->BA
BAA->ABC
```
***Соответствующий файл output.txt:***
```
3 3
```

## **Задача D.** Прямоугольное деление

*Исходный файл: input.txt*
*Выходной файл: output.txt*
*Время тестирования: 2 секунды*

Дано $N$ прямоугольников ($1 \le N \le 100$) со сторонами, параллельными осям координат. Требуется определить, на сколько частей эти прямоугольники разбивают плоскость. (Внутри частей не должно быть границ прямоугольников).

***Входные и выходные данные***
Входной файл содержит в первой строке число прямоугольников $N$. Далее идут $N$ строк, содержащих по 4 числа $x_1$ $y_1$ $x_2$ $y_2$ — целые координаты двух противоположных углов прямоугольника, не превосходящие по абсолютной величине 10 000.
Выходной файл должен содержать единственное число — количество частей.

***Пример файла input.txt:***
```
3
10 20 50 30
40 10 50 25
40 25 80 30
```
***Соответствующий файл output.txt:***
```
6
```

## **Задача E.** Двойная решетка

*Исходный файл: input.txt*
*Выходной файл: output.txt*
*Время тестирования: 2 секунды*

Две бесконечные равномерные прямоугольные решетки заданы размерами ячеек $x_1$ $y_1$ и $x_2$ $y_2$. Все размеры — целые в диапазоне от *1* до *100*. Решетки расположены на плоскости параллельно друг другу так, что целочисленное смещение одного из узлов второй решетки относительно узла первой составляет $0 \le \Delta x < x_1$ по горизонтали и $0 \le \Delta y < y_1$ по вертикали. В результате наложения образуется новая "составная" решетка с более мелкими ячейками различного размера.
Требуется вывести в порядке возрастания все различные площади ячеек составной решетки.

***Входные и выходные данные***
Во входном файле указаны числа $x_1$ $y_1$ $x_2$ $y_2$ $\Delta x$ $\Delta y$.
Выходной файл должен в первой строке содержать $N$ — количество получившихся площадей, а в следующих $N$ строках — сами площади.

***Пример файла input.txt:***
```
20 20 12 10 2 0
```

***Соответствующий файл output.txt:***
```
4
20
60
100
120
```

## **Задача F.** Покер

*Исходный файл: input.txt*
*Выходной файл: output.txt*
*Время тестирования: 2 секунды*

Игра в покер проходит следующим образом. Игра состоит из нескольких партий. В начале каждой партии банк пуст. Все игроки вносят в банк одинаковый начальный взнос. Текущая ставка устанавливается равной начальному взносу. Далее игроки делают ходы по очереди. В свой ход игрок может:

- поддержать текущую ставку, внеся в банк сумму текущей ставки;
- повысить текущую ставку на определенную величину, внеся сумму увеличенной ставки; или
- выйти из игры до окончания партии, отказываясь от дальнейших взносов и возможности выигрыша (спасовать);
- открыть карты, после чего все остальные игроки тоже открывают карты и немедленно определяют победителя текущей партии. Победителем также становится игрок, оставшийся в игре единственным (когда остальные спасовали).

Каждая партия начинается с хода одного и того же игрока. В ходе игры ведутся записи о действиях игроков:
`Ante <n>` — начало партии с начальным взносом `<n>`;
`Call` — очередной игрок поддерживает ставку;
`Raise <n>` — очередной игрок повышает ставку на `<n>`;
`Fold` — очередной игрок выходит из игры до окончания партии;
`<xxx> wins` — игроки открыли карты и игрок по имени `<xxx>` выиграл.
Партия заканчивается либо по записи `wins`, либо по записи `fold`, после которой остается единственный игрок (победитель). Победителю достается сумма, находящаяся в этот момент в банке.
Имеются записи о ходе игры $N$ игроков ($2 \le N \le 100$). Требуется определить для каждого игрока общую сумму его выигрыша или проигрыша.

***Входные и выходные данные***
В первой строке входного файла находится число $N$. Следующие $N$ строк содержат имена игроков, записанные латинскими буквами. Далее идут записи об игре по одной записи в строке. Записи могут содержать произвольное количество незначащих пробелов. Все записи не различают регистр букв. Все числа, встречающиеся во входных и выходных данных — целые положительные, не превосходящие *1 000 000 000* (одного миллиарда). В каждой партии игроки ходят в порядке перечисления во входном файле, начиная с первого.

Выходной файл должен содержать *N* строк, по строке на игрока, идущие в том же порядке, что и во входном файле. Каждая строка может быть:
`<xxx> earned <nnn>` — если игрок `<xxx>` выиграл сумму `<nnn>`;
`<xxx> lost <nnn>` — если игрок `<xxx>` проиграл сумму `<nnn>`;
`<xxx> retained` — если игрок `<xxx>` остался при своих деньгах.
Имя каждого игрока должно быть выведено только строчными (lowercase) буквами.

***Пример файла input.txt:***
```
3
Alpha
Beta
Gamma
  ante 100
alpha WINS
  aNTE  10
Call
raise 20
fold
raise 40
CALL
fold
```

***Соответствующий файл output.txt:***
```
alpha earned 110
beta retained
gamma lost 110
```