1. Шифр Цезаря

30 очков

Шифром Цезаря или шифром со сдвигом называется следующая система шифрования:

1. Фиксируется шаг сдвига N ($1 \leqslant N <$ размер алфавита).
2. В шифруемом тексте каждая буква заменяется буквой, идущей через букв N за ней в алфавите. При этом алфавит рассматривают циклично, то есть первую букву считают идущей за последней.

Например, в случае латинского алфавита A...Z и шага $N = 2$ замены должны производиться следующим образом: A → C, B → D, C → E, ..., X → Z, Y → A, Z → B.

При шифровании текста заменяются только буквы. Все остальные символы остаются неизменёнными.

Расшифровать текст, про который известно, что он закодирован шифром Цезаря и все слова в нём происходят от данного списка английских слов.

Оценивание. В этой задаче дано 10 комплектов входных данных в файлах `cstest.01.sis` до `cstest.10.sis`, и в качестве решения требуется представить соответствующие комплекты выходных данных в файлах `cstest.01.val` до `cstest.10.val`. Программу не представлять, она не оценивается. Дополнительно в файле `cs.txt` дан список слов, которые могут встречаться в выходных файлах, каждое слово в отдельной строке.

Входные данные. Во входном файле до 100 строк, в каждой строке до 100 знаков. Сдвиг N не задан.

Выходные данные. В выходной файл вывести результат расшифровки данных, полученных из входного файла.

Пример.

Входной файл	Выходной файл
`JGNNQ, YQTNF!`	`HELLO, WORLD!`

В данном случае текст зашифрован со сдвигом $N = 2$.

2. Перечисление слов

1 секунда

30 очков

Написать программу, которая подсчитывает, сколько можно составить различных непустых слов длиной не более M букв, используя алфавит из N различных букв.

Например, из 2-буквенного алфавита можно составить 14 слов длиной до 3 символов. Если использовать буквы A и B, то эти 14 слов следующие: A, AA, AAA, AAB, AB, ABA, ABB, B, BA, BAA, BAB, BB, BBA, BBB.

Входные данные. В единственной строке текстового файла `sl.sis` дано количество букв алфавита N ($1 \leqslant N \leqslant 100$) и максимальная длина составляемых слов M ($1 \leqslant M \leqslant 100$).

Выходные данные. В единственную строку текстового файла `sl.val` вывести количество слов длиной не более M, составленных с помощью N-буквенного алфавита. Можно предполагать, что во всех тестах ответ меньше 2^{31}.

Пример.

`sl.sis`	`sl.val`
`2 3`	`14`

3. Крыса и сыр

1 секунда 40 очков

Рассмотрим поведение крысы в лабиринте, построенном из единичных кубиков, расположенном в координатной сетке. Дно лабиринта — прямоугольник размером $N \times M$, а обнесён он стеной, через которую крыса перелезть не может.

Опыты показывают, что если в правый нижний угол лабиринта поместить кусочек сыра и куда-нибудь в другое место поместить крысу, то крыса начинает двигаться на запах, с каждым шагом двигаясь вправо или вниз.

Написать программу, которая для каждой пустой клетки выясняет, сможет ли помещённая в неё крыса добраться до сыра или нет, и в случае положительного ответа — в каком направлении должна двигаться крыса из этой клетки для достижения своей цели.

Входные данные. В первой строке текстового файла `rj.sis` даны размеры лабиринта N и M ($1 \leqslant N, M \leqslant 100$). В каждой из следующих N строк дано ровно M символов: описание лабиринта, где `O` означает пустую клетку, `#` — непроходимую клетку, и `J` — клетку, в которой находится кусок сыра (во входном файле может встречаться только одна буква `J` и она всегда в самой правой нижней клетке).

Выходные данные. В выходной файл `rj.val` вывести ровно N строк, в каждую строку ровно M символов: описание заданного лабиринта, где в каждой пустой клетке (помеченной знаком `O` во входном файле) находится знак `>`, если крыса должна двигаться из этой клетки направо, знак `V`, если крыса из этой клетки должна двигаться вниз, знак `X`, если крыса из этой клетки может двигаться в обоих направлениях, и знак `@`, если крыса не может попасть к сыру из этой клетки. Все непустые клетки должны остаться неизменёнными.

Пример.

`rj.sis`
```
4 5
OOOOO
OO#OO
OOO#O
O#OOJ
```

`rj.val`
```
XX>XV
XV#>V
>>V#V
@#>>J
```

1. IPv6 адреса

1 секунда — 30 очков

В протоколе IPv6 адреса 128-битные. Их записывают в виде последовательности из восьми двухбайтных шестнадцатиричных чисел, например A001:DB8:31:1:20A:95FF:FEF5:246E.

В IPv6 адресах может встречаться довольно много нулей, например A001:DB8:0:0:0:0:0:1. Для упрощения написания договорено, что в адресе можно опустить одну подпоследовательность, состоящую из одних нулей, отмечая место опущения двойным двоеточием, например A001:DB8::1. Поскольку длина адреса известна, отсутствующие нули всегда можно восстановить.

Написать программу, которая считывает из входного файла IPv6 адрес в 16-ричном формате и выводит его в виде 128-битного двоичного числа.

Входные данные. В единственной строке текстового файла `ip.sis` задан один IPv6 адрес в 16-ричном формате — до 39 знаков `0`...`9` (десятичные цифры), `A`...`F` (большие латинские буквы) и `:` (двоеточие).

Выходные данные. В единственную строку текстового файла `ip.val` вывести ровно 128 знаков `0` и `1` — заданный адрес в двоичном формате.

Пример.

ip.sis	ip.val
A001:DB8:0:0:0:0:0:1	10100000000000010000110110111 0...01

Пример.

ip.sis	ip.val
A001:DB8::1	10100000000000010000110110111 0...01

Примечание. В приведённых примерах некоторые нули остались ненапечатанными, потому что ответ не помещался полностью на страницу. Выходные файлы целиком доступны в электронном виде на сервере соревнований.

Примечание. В позиционной системе счисления с основанием b (или b-ичной системе) для обозначения чисел используются цифры $0 \ldots b-1$, а числовое значение последовательности цифр $a_n a_{n-1} \ldots a_1 a_0$ равно

$$a_n \cdot b^n + a_{n-1} \cdot b^{n-1} + \ldots + a_1 \cdot b + a_0.$$

Если $b > 10$, то в дополнение к обычным арабским цифрам используются буквы латинского алфавита. Таким образом, в шестнадцатиричной системе счисления используются арабские цифры 0...9 и латинские буквы A...F (=10...15), а значение 16-ричного числа A001 равно

$$10 \cdot 16^3 + 0 \cdot 16^2 + 0 \cdot 16 + 1 = 40961.$$

Стоит заметить, что каждое 16-ричное число записывается в двоичной системе ровно четырьмя битами:

16-ричный	2-ичный	16-ричный	2-ичный
0	0000	8	1000
1	0001	9	1001
2	0010	A	1010
3	0011	B	1011
4	0100	C	1100
5	0101	D	1101
6	0110	E	1110
7	0111	F	1111

2. Скобочные выражения

1 секунда — 30 очков

Корректное скобочное выражение определяется следующим образом:

1. Пустая строчка является корректным скобочным выражением.
2. Если A — корректное скобочное выражение, то (A) тоже корректное скобочное выражение. Составленное по этому правилу выражение (A) в дальнейшем будем называть атомом, а его подвыражение A — ядром этого атома.
3. Если A и B — корректные скобочные выражения, то AB тоже корректное скобочное выражение.
4. Корректные скобочные выражения — это только те выражения, которые могут быть получены с помощью правил 1 до 3.

Например, (), ()(), (()) и (()()) — корректные скобочные выражения, а выражения (,)(и ((())) не являются корректными.

Говорим, что скобочное выражение неизбыточное, если:

1. В этом выражении нет идущих подряд трёх атомов. То есть, в неизбыточном выражении не должно встречаться подвыражения вида $(A)(B)(C)$.
2. В этом выражении нигде нет атома, ядро которого тоже является атомом. То есть, в неизбыточном выражении не должно встречаться подвыражения вида $((A))$.

Например, (()()) — это корректное неизбыточное скобочное выражение, а ((()()()), (((()))) и (()())((()())(()())) — корректные, но избыточные скобочные выражения.

Написать программу, которая конструирует корректное неизбыточное скобочное выражение заданной длины.

Входные данные. В единственной строке текстового файла `sa.sis` задана длина выражения N ($2 \leqslant N \leqslant 100\,000$, N — чётное число).

Выходные данные. В единственную строку текстового файла `sa.val` вывести ровно N знаков `(` и `)` — корректное неизбыточное скобочное выражение длины N. Если существует несколько подходящих решений, вывести любое из них.

Пример.

sa.sis	sa.val
6	(()())

3. Экспрессы

1 секунда 40 очков

В одном городе есть N автобусных остановок и между ними ходит M автобусных линий. Все линии — экспрессы, то есть автобус едет от начальной остановки прямо до конечной без промежуточных остановок. Сеть линий такова, что между любыми двумя остановками есть не более одной линии, но с любой остановки можно доехать до любой другой остановки — прямо или с пересадками. На всех линиях автобусы ездят в обе стороны.

В городе действует единый проездной билет, с которым можно проехать один раз по любой линии. Пассажир хочет доехать от остановки A до остановки B, истратив на это минимальное количество билетов. Написать программу, которая подсчитывает, сколькими способами он может это сделать.

Входные данные. В первой строке входного файла `eb.sis` дано количество остановок N ($2 \leqslant N \leqslant 100$) и количество линий M ($1 \leqslant M \leqslant 1000$). Остановки пронумерованы числами $1 \ldots N$. В каждой из следующих M строк даны номера конечных остановок одной линий a и b ($1 \leqslant a < b \leqslant N$). При этом все эти M строк различные. В последней строке файла даны номера начальной и конечной остановок пассажира A и B ($1 \leqslant A, B \leqslant N$, $A \neq B$).

Выходные данные. В единственную строку текстового файла `eb.val` вывести разделённые пробелом целые числа K и S, которые означают, что для проезда от остановки A до остановки B требуется как минимум K билетов и, истратив K билетов, от остановки A до остановки B можно проехать S различными способами. Можно предполагать, что $S < 2^{31}$.

Пример.

`eb.sis`
```
5 7
1 2
2 3
3 4
4 5
1 3
1 4
2 5
4 2
```

`eb.val`
```
2 3
```

Маршруты с двумя билетами — это $4 \to 1 \to 2$, $4 \to 3 \to 2$ и $4 \to 5 \to 2$. Можно было бы также поехать маршрутом $4 \to 1 \to 3 \to 2$, но в этом случае пришлось бы истратить уже три билета.

Оценивание. В этом задании нахождение числа K даёт 25% очков. Если ваша программа не может найти число S, выведите вместо него число `-1`.