

Задача 1. Незнайка и римско-буквенная система счисления**Критерий оценивания решений задачи 1**

Общим критерием оценивания решения любой задачи является количество тестов, верно пройденных программой, составленной участником. Сначала решение проверяется на тестах из условия. Если не пройден хотя бы один из них, то решение оценивается в 0 технических баллов. Если тесты из условия пройдены, то решение проверяется на основном наборе тестов. Высчитывается доля (в процентах) верно пройденных тестов из основного набора по отношению к общему количеству тестов в наборе. Полученное количество процентов, округлённое до целого, определяет количество технических баллов за решение.

Технические баллы за задачу 1

За решение задачи начисляется от 0 до 100 технических баллов по критерию, указанному выше.

Перевод технических баллов в оценку

По каждой задаче определяется самое удачное её решение, отправленное участником. Технические баллы за такие решения всех шести задач суммируются. Высчитывается доля (в процентах), которую составила полученная сумма по отношению к 600 техническим баллам. Полученное количество процентов, округлённое до целого, становится оценкой. Таким образом, за решение одной задачи начисляется от 0 до 17 итоговых баллов.

Условия задачи 1

В прошлом году Знайка ввёл в Цветочном городе буквенную систему счисления. Это позиционная система счисления с основанием 52, в которой цифрами служат заглавные и строчные латинские буквы и только они. В ней используется такая таблица цифр и их значений:

цифра	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
значение	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
цифра	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
значение	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51

Приведем пример записи числа в буквенной системе счисления: $2025_{10} = 38 * 52^1 + 49 * 52^0 = mx$

В этом году Знайка решил не останавливаться на достигнутом. Он вычитал в научном журнале, выписанном из Солнечного города, про двоично-десятичный способ записи чисел, при котором каждая цифра десятичной записи числа представляется четырёхбитным двоичным кодом её значения. Пример записи числа двоично-десятичным способом: $2025_{10} = 0010.0000.0010.0101_{bcd}$

Знайке понравилась идея смешивать разные системы счисления и получать новые способы записи чисел, и он пошел по этому пути. Знайка решил каждую цифру своей буквенной системы счисления представить как римское число, и ставить одну точку между записями соседних цифр. Так им была изобретена римско-буквенная система. В ней используется такая таблица (обратите внимание, что в качестве римских цифр используются заглавные латинские буквы):

Решение

В решении стоит использовать массив из 45 элементов со значениями от 0 до 51 как внутреннее представление чисел, записанных 45 разрядами рассматриваемой системы. Для удобства сравнения старшие разряды стоит хранить в начальных элементах массива. В старших разрядах могут быть незначащие нули. Программа будет в цикле считывать запись очередного числа последовательности, осуществлять перевод записи во внутреннее представление, сравнивать текущее число с текущим максимумом и обновлять текущий максимум, если считанное число больше. После окончания ввода найденный максимум будет переводиться в буквенную систему счисления и выводиться как результат.

Код возможного решения

```
program ROMLETTERS11(input, output);
type    number = array [1..45] of byte;
var     K, I : word;
        RESULT, AI : number;
function roman2Decimal(DIGIT : char): byte;
var RESULT : byte;
begin
    case DIGIT of
        'N' : begin RESULT := 0 end;
        'I' : begin RESULT := 1 end;
        'V' : begin RESULT := 5 end;
        'X' : begin RESULT := 10 end;
        'L' : begin RESULT := 50 end;
        else RESULT := 255;
    end;
    roman2Decimal := RESULT
end;
function decimal2Letter(DIGIT : byte): char;
var RESULT : char;
begin
    case DIGIT of
        0..25 : begin RESULT := chr(ord('A') + DIGIT) end;
        26..51 : begin RESULT := chr(ord('a') + DIGIT - 26) end;
        else RESULT := '#';
    end;
    decimal2Letter := RESULT
end;
procedure readRoman(var N : byte);
var BUFFER : array[1..7] of byte;
    I : word; CH : char; D : byte;
begin
    for I := 1 to 7 do BUFFER[I] := 255;
    if not (eoln or eof) then begin
        read(CH);
        I := 1;
        D := roman2Decimal(CH);
        while (D < 52) and (I < 8) do begin
            BUFFER[I] := D;
            if not (eoln or eof) then begin
                read(CH);
                D := roman2Decimal(CH);
                I := I + 1;
            end
        end
    end
```

```

        else D := 255
    end;
    if (I = 1) and (BUFFER[1] > 51) then N := 255
    else begin
        I := 1;
        N := 0;
        while (I < 8) and (BUFFER[I] < 52) do begin
            if (I > 1) and (BUFFER[I-1] < BUFFER[I]) then
                N := N + BUFFER[I] - 2 * BUFFER[I-1]
            else N := N + BUFFER[I];
            I := I + 1;
        end
    end
end
else N := 255
end;
procedure writeNumber(var A : number);
var I, J : word;
begin
    I := 1;
    while (I < 45) and (A[I] = 0) do I := I + 1;
    for J := I to 45 do write(decimal2Letter(A[J]))
end;
procedure readNumber(var A : number);
var CH : byte; I, J : word;
begin for I := 1 to 45 do A[I] := 0;
    readRoman(CH);
    I := 1;
    J := 0;
    while (I <= 45) and (CH < 52) do begin
        J := J + 1; A[J] := CH;
        if not (eoln or eof) then begin
            readRoman(CH);
            I := I + 1
        end else I := 46
    end;
    for I := J downto 1 do A[45 - J + I] := A[I];
    for I := 1 to 45 - J do A[I] := 0;
end;

begin readln(K);
    readNumber(RESULT);
    if eoln and not(eof) then readln;
    for I := 2 to K do begin
        readNumber(AI);
        if eoln and not(eof) then readln;
        if (CompareByte(RESULT, AI, 45) <= 0) then RESULT := AI
    end;
    writeNumber(RESULT)
end.

```

Задача 2. CodeWeakness

Критерий оценивания решений задачи 2

Общим критерием оценивания решения любой задачи является количество тестов, верно пройденных программой, составленной участником. Сначала решение проверяется на тестах из условия. Если не пройден хотя бы один из них, то решение оценивается в 0 технических баллов. Если тесты из условия пройдены, то решение проверяется на основном наборе тестов. Высчитывается доля (в процентах) верно пройденных тестов из основного набора по отношению к общему количеству тестов в наборе. Полученное количество процентов, округлённое до целого, определяет количество технических баллов за решение.

Технические баллы за задачу 2

За решение задачи начисляется от 0 до 100 технических баллов по критерию, указанному выше.

Перевод технических баллов в оценку

По каждой задаче определяется самое удачное её решение, отправленное участником. Технические баллы за такие решения всех шести задач суммируются. Высчитывается доля (в процентах), которую составила полученная сумма по отношению к 600 техническим баллам. Полученное количество процентов, округлённое до целого, становится оценкой. Таким образом, за решение одной задачи начисляется от 0 до 17 итоговых баллов.

Условия задачи 2

Миша готовится к олимпиаде «Прямоносов» и читает разбор к задаче GHD на популярном сайте CodeWeakness. В задаче для массива A длины n нужно посчитать некоторый параметр НПД (наибольший половинный делитель). Решение задачи в лоб требовало $O(n^2d)$, а оптимизированное решение – $O(nd^2)$, где n – это длина массива A , d – это максимальное значение $d(A[i])$ на элементах массива A , а функция $d(x)$ – это число различных делителей натурального числа x . Например, $d(6) = 4$, так как делители 6: 1, 2, 3, 6. Миша ничего не понял из разбора, но заинтересовался тем, как по параметру n найти наименьшее j , такое, что $d(j)$ будет равно максимальному $d(i)$ для $i = 1, 2, \dots, n$. Например, при $n = 10$ $d(6) = d(8) = d(10) = 4$, и искомым будет число 6. Помогите Мише разобраться с этим вопросом, чтобы он мог продолжить зависать на CodeWeakness.

Формат ввода: Вводится натуральное число n в десятичной записи: $1 \leq n \leq 100000000$.

Формат вывода: Выводится запись в десятичной системе счисления наименьшего j , такого, что $d(j)$ равно максимуму $d(i)$ для $i = 1, 2, \dots, n$.

Ввод примера №1:		Ввод примера №2:		Ввод примера №3:
1		10		100
Вывод примера №1:		Вывод примера №2:		Вывод примера №3:
1		6		60

Решение

Сверхсоставным числом называется такое число, у которого количество делителей больше, чем у любого числа, меньшего, чем оно. То есть, n сверхсоставное, если $d(n) > d(i)$ для $i = 1, 2, \dots, n-1$. Таблица первых 10000 сверхсоставных чисел известна и известны значения функции $d(x)$ для них. Например, эти сведения могут быть найдены на странице про последовательность A002182 в OEIS (On-Line Encyclopedia of Integer Sequences): <https://oeis.org/A002182> В диапазон до 100000000 попадают первые 56 сверхсоставных чисел. Если не пользоваться готовой таблицей, то можно самостоятельно рассчитать значения из нее. Подробности проведения этих расчётов опускаем. Заметим лишь, что они могут занять значительное время.

Считав заданное число, нужно найти ближайшее к нему, но не превосходящее его сверхсоставное число. Сделать это можно бинарным поиском по таблице сверхсоставных чисел. Затем нужно вывести найденное число.

Код возможного решения

```
program CODEWEAKNESS11(input, output);
const NUMHCN = 56;
    HCN: array [1 .. NUMHCN] of dword = (1, 2, 4, 6, 12, 24, 36, 48,
```

```

        60, 120, 180, 240, 360, 720, 840, 1260, 1680, 2520, 5040, 7560,
        10080, 15120, 20160, 25200, 27720, 45360, 50400, 55440, 83160, 110880,
        166320, 221760, 277200, 332640, 498960, 554400, 665280, 720720, 1081080,
        1441440, 2162160, 2882880, 3603600, 4324320, 6486480, 7207200, 8648640,
        10810800, 14414400, 17297280, 21621600, 32432400, 36756720, 43243200,
        61261200, 73513440);
var N, RESULT : dword;

function binSearch(A : dword) : byte;
var L, M, H : byte;
begin
    L := 1;
    H := NUMHCN;
    while L <= H do
    begin
        M := (L + H) div 2;
        if HCN[M] > A then
        begin
            H := M - 1;
        end
        else if HCN[M] < A then
        begin
            L := M + 1;
        end
        else
        begin
            break;
        end;
    end;
    if (M < NUMHCN) AND (HCN[M + 1] <= A) then binSearch := M + 1
    else if (M > 1) AND (HCN[M] > A) then binSearch := M - 1
    else binSearch := M
end;

begin
    read(N);
    RESULT := HCN[binSearch(N)];
    write(RESULT)
end.

```

Задача 3. Незнайка и реновация

Критерий оценивания решений задачи 3

Общим критерием оценивания решения любой задачи является количество тестов, верно пройденных программой, составленной участником. Сначала решение проверяется на тестах из условия. Если не пройден хотя бы один из них, то решение оценивается в 0 технических баллов. Если тесты из условия пройдены, то решение проверяется на основном наборе тестов. Высчитывается доля (в процентах) верно пройденных тестов из основного набора по отношению к общему количеству тестов в наборе. Полученное количество процентов, округлённое до целого, определяет количество технических баллов за решение.

Технические баллы за задачу 3

За решение задачи начисляется от 0 до 100 технических баллов по критерию, указанному выше.

Перевод технических баллов в оценку

По каждой задаче определяется самое удачное её решение, отправленное участником. Технические баллы за такие решения всех шести задач суммируются. Высчитывается доля (в процентах), которую составила полученная сумма по отношению к 600 техническим баллам. Полученное количество процентов, округлённое до целого, становится оценкой. Таким образом, за решение одной задачи начисляется от 0 до 17 итоговых баллов.

Условия задачи 3

Знайка побывал на экскурсии в Солнечном городе и ему там очень понравилось. Вернувшись домой, он решил провести реновацию Цветочного города – расширить его и полностью перестроить по примеру Солнечного города. По плану Знайки обновлённый Цветочный город станет квадратом из $N \times N$ одинаковых квадратных участков. На каждом участке будет выстроен один домик коротышки. Домики будут перенумерованы согласно бустрофедоническому способу. Название способа дано из-за схожести с формой борозды, оставляемой при вспашке поля плугом, в который запряжён бык.

Домик с номером 0 (ноль) – домик Знайки – расположится в юго-восточном углу (на плане это нижний правый угол). От него нумерация пойдёт вдоль горизонтальной стороны квадрата по направлению влево. Достигнув левого края, нумерация продолжится по соседнему горизонтальному ряду домиков, расположенных севернее, но она пойдёт в обратном направлении – слева-направо. Далее она так и будет продолжаться. Всякий раз, достигнув края, будет происходить переход на соседний ряд, расположенный севернее, а направление будет меняться на противоположное. Например, при $N = 7$ план реновации Цветочного города будет таким:

48	47	46	45	44	43	42
35	36	37	38	39	40	41
34	33	32	31	30	29	28
21	22	23	24	25	26	27
20	19	18	17	16	15	14
7	8	9	10	11	12	13
6	5	4	3	2	1	0

Коротышки сразу же согласились с реновацией, предложенной Знайкой, полагая, что ему виднее. Один лишь Незнайка призадумался о том, как ему теперь ходить в гости к своему другу Гуньке. Раньше ему достаточно было идти по улице Колокольчиков до перекрестка с улицей Маргариток, свернуть на неё и продолжать идти ещё пару кварталов. Единственной надеждой Незнайки было то, что кто-то придёт ему на помощь и составит программу, которая сумеет ориентироваться в плане нового Цветочного города.

Требуется написать программу, которая считывает N – количество участков вдоль одной стороны города. Затем программа считывает A и B – номера домиков, где A – номер домика Незнайки, а B – номер домика Гуньки. Номера A и B могут совпадать, так как не исключено, что после реновации Незнайка и Гунька будут жить в одном и том же домике. Программа находит и выводит результаты – номера рядов домиков, на пересечении которых стоит домик A , номера рядов домиков, на пересечении которых стоит домик B , а также «манхэттенское расстояние»

между домиками с номерами A и B . «Манхэттенское расстояние» рассчитывается по формуле: $D_{AB} = |I_A - I_B| + |J_A - J_B|$, где I_A – номер горизонтального ряда домиков, в котором находится домик A , J_A – номер вертикального ряда домиков, в котором находится домик A ; I_B – номер горизонтального ряда домиков, в котором находится домик B , J_B – номер вертикального ряда домиков, в котором находится домик B . Горизонтальные ряды домиков нумеруются с 1 до N сверху вниз. Вертикальные ряды домиков нумеруются с 1 до N слева направо. Например, если Цветочный город представляет собой квадрат $7 * 7$, как выше, и $A = 8$, а $B = 40$, то $I_A = 6$, $J_A = 2$, $I_B = 2$, $J_B = 6$ и $D_{AB} = |6 - 2| + |2 - 6| = 8$. Один из возможных путей от домика A к домику B , имеющий длину 8 показан на плане ниже:

48	47	46	45	44	43	42
35	36	37	38	39	40	41
34	33	32	31	30	29	28
21	22	23	24	25	26	27
20	19	18	17	16	15	14
7	8	9	10	11	12	13
6	5	4	3	2	1	0

Формат ввода: В первой строке вводится натуральное число N в десятичной записи: $1 \leq N \leq 2^{32} - 1$. Во второй строке вводятся неотрицательные целые числа A и B , разделённые пробелом: $0 \leq A \leq (N^2 - 1)$, B : $0 \leq B \leq (N^2 - 1)$.

Формат вывода: В первой строке выводятся десятичные записи найденных натуральных чисел I_A , J_A , разделённые пробелом (номера горизонтального и вертикального рядов, на пересечении которых находится домик A). Во второй строке выводятся десятичные записи найденных натуральных чисел I_B , J_B , разделённые пробелом (номера горизонтального и вертикального рядов, на пересечении которых находится домик B). В третьей строке выводится десятичная запись найденного неотрицательного целого числа D_{AB} . При выводе незначащие нулевые разряды не выводятся.

Ввод примера №1:		Ввод примера №2:		Ввод примера №3:
1		2		7
0 0		0 3		8 40
Вывод примера №1:		Вывод примера №2:		Вывод примера №3:
1 1		2 2		6 2
1 1		1 2		2 6
0		1		8

Решение

Чтобы по номеру домика найти номер горизонтального ряда (строки), в котором он расположен, нужно поделить нацело его номер на N и вычесть полученное частное из N . Если этот номер чётный и N нечётно, то номер вертикального ряда находится как увеличенный на 1 остаток от деления нацело на N номера домика. Аналогично при чётном N и нечётном номере горизонтального ряда. Если горизонтальный ряд нечётный и N нечётно, то остаток от деления нацело на N номера домика надо вычесть из N . Аналогично при чётном N и нечётном номере горизонтального ряда. Найдя номера рядов для домика A и домика B остаётся подставить их в формулу для D_{AB} и найти результат.

Код возможного решения

```

program BOUSTROPHEDON11(input, output);
var N, IA, IB, JA, JB, A, B, D: qword;
begin
    readln(N);
    read(A);
    read(B);
    IA := N - A div N;

```



```

IB := N - B div N;
if odd(IA) = odd(N) then  JA := N - A mod N
else JA := A mod N + 1;
if odd(IB) = odd(N) then  JB := N - B mod N
else JB := B mod N + 1;
if (IA >= IB) then D := IA - IB
else D := IB - IA;
if (JA >= JB) then D := D + JA - JB
else D := D + JB - JA;
writeln(IA,' ',JA);
writeln(IB,' ',JB);
write(D)
end.

```

Задача 4. Радио Судного Дня

Критерий оценивания решений задачи 4

Общим критерием оценивания решения любой задачи является количество тестов, верно пройденных программой, составленной участником. Сначала решение проверяется на тестах из условия. Если не пройден хотя бы один из них, то решение оценивается в 0 технических баллов. Если тесты из условия пройдены, то решение проверяется на основном наборе тестов. Высчитывается доля (в процентах) верно пройденных тестов из основного набора по отношению к общему количеству тестов в наборе. Полученное количество процентов, округлённое до целого, определяет количество технических баллов за решение.

Технические баллы за задачу 4

За решение задачи начисляется от 0 до 100 технических баллов по критерию, указанному выше.

Перевод технических баллов в оценку

По каждой задаче определяется самое удачное её решение, отправленное участником. Технические баллы за такие решения всех шести задач суммируются. Высчитывается доля (в процентах), которую составила полученная сумма по отношению к 600 техническим баллам. Полученное количество процентов, округлённое до целого, становится оценкой. Таким образом, за решение одной задачи начисляется от 0 до 17 итоговых баллов.

Условия задачи 4

В далёком будущем человечество больше не имеет спутниковой связи. Единственная надежда — старые передатчики «Радио Судного Дня», которые всё ещё периодически посылают зашифрованные сигналы в эфир. Каждый передатчик транслирует некую строку из заглавных латинских букв $A-Z$, повторяя её бесконечно. Например, если строка передатчика равна AB , то в эфире слышится $ABABABAB\dots$

Учёные зафиксировали два разных сигнала — от двух станций: строки S и T . Теперь нужно выяснить: может ли существовать единый сигнал, который оба передатчика могли бы генерировать синхронно (возможно, с разной скоростью повтора)?

Нужно составить программу, которая по двум заданным строкам S и T попытается найти наикратчайшую строку U , такую, что её повторением можно получить и S , и T . Если такую строку удастся найти, то программа должна её вывести. Если такой строки не существует, то программа должна вывести $NO\ SIGNAL$.

Формат ввода: В первой строке вводится непустая последовательность заглавных латинских букв — строка S . Во второй строке вводится непустая последовательность заглавных латинских букв — строка T . Длина каждой из этих строк не превышает 1000000.

Формат вывода: Если существуют строки U , такие, что строка S является повторением строки U k раз и строка T является повторением строки U m раз (k и m натуральные и неотрицательные), то следует выбрать среди возможных строк U строку минимальной длины и вывести её. Иначе следует вывести строку $NO\ SIGNAL$.

Ввод примера №1:		Ввод примера №2:		Ввод примера №3:
ABAB		ABCABC		AAAA
AB		AB		AA
Вывод примера №1:		Вывод примера №2:		Вывод примера №3:
AB		NO SIGNAL		A
Пояснение №1:		Пояснение №2:		Пояснение №3:
$S = "AB" \times 2$		U не существует		$S = "A" \times 4$
$T = "AB" \times 1$				$T = "A" \times 2$
$U = "AB"$				$U = "A"$

Решение

Эффективное решение может базироваться на использовании Z-функции (Z-function).

Код возможного решения

```

#include <bits/stdc++.h>
#define int long long
using namespace std;
vector<int> zfunc(string& s) {
    int n = s.size();
    vector<int> z(n);
    int l = 0, r = 0;
    for (int i = 1; i < n; i++) {
        if (i < r) {
            z[i] = min(r - i, z[i - l]);
        }
        while (i + z[i] < n && s[z[i]] == s[i + z[i]]) {
            z[i]++;
        }
        if (i + z[i] > r) {
            l = i;
            r = i + z[i];
        }
    }
    return z;
}

int period(string& s) {
    auto z = zfunc(s);
    int n = s.size();
    int i = 1;
    for (i = 1; i < n; i++) {
        if (n % i != 0) {
            continue;
        }
        if (i + z[i] == n) {
            break;
        }
    }
    return i;
}

signed main(void) {
    string S, T;
    cin >> S >> T;
    int ps = period(S);
    int ts = period(T);
    if (S.substr(0, ps) != T.substr(0, ts)) {
        cout << "NO SIGNAL" << endl;
        return 0;
    }
    cout << S.substr(0, ps);
    cout << endl;
    return 0;
}

```

Задача 5. Вещества

Критерий оценивания решений задачи 5

Общим критерием оценивания решения любой задачи является количество тестов, верно пройденных программой, составленной участником. Сначала решение проверяется на тестах из условия. Если не пройден хотя бы один из них, то решение оценивается в 0 технических баллов. Если тесты из условия пройдены, то решение проверяется на основном наборе тестов. Высчитывается доля (в процентах) верно пройденных тестов из основного набора по отношению к общему количеству тестов в наборе. Полученное количество процентов, округлённое до целого, определяет количество технических баллов за решение.

Технические баллы за задачу 5

За решение задачи начисляется от 0 до 100 технических баллов по критерию, указанному выше.

Перевод технических баллов в оценку

По каждой задаче определяется самое удачное её решение, отправленное участником. Технические баллы за такие решения всех шести задач суммируются. Высчитывается доля (в процентах), которую составила полученная сумма по отношению к 600 техническим баллам. Полученное количество процентов, округлённое до целого, становится оценкой. Таким образом, за решение одной задачи начисляется от 0 до 17 итоговых баллов.

Условия задачи 5

Доктор Пилюлкин изготавливает в лаборатории разные лекарственные вещества. Доктор нашел древний трактат Николаса Фламелья и хочет узнать, атомы каких веществ упоминаются в трактате.

Молекулы состоят из атомов. Атом – это либо одна заглавная латинская буква, либо заглавная латинская буква, за которой идут одна или более строчных латинских букв. Примеры обозначения атомов: H, El, Elerium. Если после обозначения атома записано положительное целое число, оно обозначает число атомов этого типа. Молекула состоит из атомов с указанием числа повторений. Примеры молекул: H₂O, C₂H₅OLi. Круглые или квадратные скобки могут использоваться в записи молекулы для группировки, и тогда число повторений относится ко всей группе атомов в скобках. Например, [(CH₂)₁₀HSi]₂. Пробелы в записи не допускаются.

Формулы в трактате могут никак не отделяться от остального текста, например, текст thatH₂gas содержит формулу молекулы H₂. В тексте выделяется формула максимально возможной длины, то есть текст thatH₂O₂Sigas! содержит формулу H₂O₂Sigas. Но формула должна быть синтаксически корректной, в противном случае она в тексте игнорируется. Например, текст [(CH₂)₁₀work не должен рассматриваться как формула вообще, даже для корректной подформулы (CH₂). Но в тексте [(CH₂)₁₀workH₂] должна быть выделена формула H₂. Другими словами, первая ошибка при распознавании формулы приводит к прекращению распознавания текущей формулы и возврату в начальное состояние распознавания, в котором может начаться новая формула.

Формат ввода: На стандартном потоке ввода задаётся текст трактата. Текст завершается признаком конца файла.

Формат вывода: Для каждого атома в формулах, распознанных в тексте, выведите суммарное количество атомов этого типа. Атомы должны выводиться в лексикографическом порядке. Между атомом и числом повторений должен выводиться ровно один пробел.

Ввод примера №1:

Water H₂O dissolves sugar C₁₂H₂₂O₁₁.

Вывод примера №1:

C 12

H 24

O 12

Water 1

Код возможного решения

```

use std::{collections::BTreeMap, error::Error, fmt::{Display, Write}, io::Read};

#[derive(Debug, Clone)]
#[allow(dead_code)]
enum Token<'a> {
    End,
    Open(u8),
    Close(u8),
    Equals,
    Atom(&'a str),
    Num(u64),
    Plus,
    Bracket(&'a str, Box<Token<'a>>, &'a str),
    Counted(Box<Token<'a>>, u64),
    Molecule(Vec<Box<Token<'a>>>, &'a str),
    Component(u64, Box<Token<'a>>),
    Side(Vec<Box<Token<'a>>>),
    Equation(Box<Token<'a>>, Box<Token<'a>>),
}

impl<'a> Display for Token<'a> {
    fn fmt(&self, f: &mut std::fmt::Formatter<'_,_>) -> std::fmt::Result {
        match self {
            Token::Open(c) => f.write_char(*c as char),
            Token::Close(c) => f.write_char(*c as char),
            Token::Equals => f.write_char('='),
            Token::Atom(s) => f.write_str(*s),
            Token::Num(x) => f.write_fmt(format_args!("{}", *x)),
            Token::Plus => f.write_char('+'),
            Token::Bracket(ob, t, cb) => {
                f.write_str(*ob)?;
                t.fmt(f)?;
                f.write_str(*cb)
            },
            Token::Counted(t, cnt) => {
                t.fmt(f)?;
                if *cnt > 1 {
                    f.write_fmt(format_args!("{}", *cnt))?;
                }
                Ok(())
            },
            Token::Molecule(_, s) => {
                f.write_str(*s)?;
                /*
                for i in 0..v.len() {
                    v[i].fmt(f)?;
                }
                */
                Ok(())
            },
            Token::Component(cnt, t) => {
                if *cnt > 1 {
                    f.write_fmt(format_args!("{}", *cnt))?;
                }
                t.fmt(f)?;
                Ok(())
            }
        }
    }
}

```

```

    },
    Token::Side(v) => {
        for i in 0..v.len() {
            if i > 0 {
                f.write_char('+')?;
            }
            v[i].fmt(f)?;
        }
        Ok(())
    },
    Token::Equation(lhs, rhs) => {
        lhs.fmt(f)?;
        f.write_char('=')?;
        rhs.fmt(f)
    },
    _ => panic!(),
}

}

#[derive(Debug, Clone)]
struct ParseError {
    msg: &'static str,
    pos: usize,
}

impl ParseError {
    fn new(msg: &'static str, pos: usize) -> Self {
        Self { msg, pos }
    }
}

impl Display for ParseError {
    fn fmt(&self, f: &mut std::fmt::Formatter<'_,>) -> std::fmt::Result {
        f.write_fmt(format_args!("parse error {} at pos {}", self.msg, self.pos))
    }
}

impl Error for ParseError {}

#[derive(Debug, Clone)]
#[allow(unused)]
struct MissingWeightError {
    atom: String
}

#[allow(unused)]
impl MissingWeightError {
    fn new(atom: &str) -> Self {
        Self { atom: atom.to_string() }
    }
}

impl Display for MissingWeightError {
    fn fmt(&self, f: &mut std::fmt::Formatter<'_,>) -> std::fmt::Result {
        f.write_fmt(format_args!("missing weight for atom {}", self.atom))
    }
}

impl Error for MissingWeightError {}

```

```

struct Parser<'a> {
    s: &'a [u8],
    len: usize,
    i: usize,
    t: Token<'a>,
    token_i: usize,
}

#[allow(unused)]
impl<'a> Parser<'a> {
    fn new(s: &'a str) -> Self {
        Self {
            s: s.as_bytes(),
            len: s.len(),
            i: 0,
            t: Token::End,
            token_i: 0,
        }
    }

    fn next_token(&mut self) -> Result<(), Box<dyn Error>> {
        self.token_i = self.i;
        if self.i >= self.len {
            self.t = Token::End;
        } else if self.s[self.i] >= b'0' && self.s[self.i] <= b'9' {
            let mut val = 0;
            while self.i < self.len && self.s[self.i] >= b'0' && self.s[self.i] <= b'9' {
                val = val * 10 + (self.s[self.i] - b'0') as u64;
                self.i += 1;
            }
            self.t = Token::Num(val);
        } else if self.s[self.i] >= b'A' && self.s[self.i] <= b'Z' {
            let beg = self.i;
            self.i += 1;
            while self.i < self.len && self.s[self.i] >= b'a' && self.s[self.i] <= b'z' {
                self.i += 1;
            }
            self.t = Token::Atom(std::str::from_utf8(&self.s[beg..self.i]).unwrap());
        } else if self.s[self.i] == b'(' || self.s[self.i] == b'[' {
            self.t = Token::Open(self.s[self.i]);
            self.i += 1;
        } else if self.s[self.i] == b')' || self.s[self.i] == b']' {
            self.t = Token::Close(self.s[self.i]);
            self.i += 1;
        } else if self.s[self.i] == b'=' {
            self.t = Token::Equals;
            self.i += 1;
        } else if self.s[self.i] == b'+' {
            self.t = Token::Plus;
            self.i += 1;
        } else {
            self.t = Token::Plus;
            self.i += 1;
            // return Err(Box::new(ParseError::new("invalid char", self.i)));
        }
    }

    Ok(())
}

```

```

}

fn counted(&mut self) -> Result<Token<'a>, Box<dyn Error>> {
    let b: Box<Token<'a>>>;
    match self.t {
        Token::Atom(_) => {
            b = Box::new(self.t.clone());
            self.next_token()?;
        },
        Token::Open(oc) => {
            let ob = std::str::from_utf8(&self.s[self.i-1..self.i]).unwrap();
            self.next_token()?;
            let m = self.molecule()?;
            if let Token::Close(cc) = self.t {
                if (oc == b'(' && cc == b')') || (oc == b'[' && cc == b']') {
                    let cb = std::str::from_utf8(&self.s[self.i-1..self.i]).unwrap();
                    self.next_token()?;
                    b = Box::new(Token::Bracket(ob, Box::new(m), cb));
                } else {
                    return Err(Box::new(ParseError::new("bracket mismatch", self.i)));
                }
            } else {
                return Err(Box::new(ParseError::new("expect closing bracket", self.i)));
            }
        },
        _ => {
            return Err(Box::new(ParseError::new("unexpected token", self.i)));
        },
    }
    let mut cnt = 1;
    if let Token::Num(cc) = self.t {
        cnt = cc;
        self.next_token()?;
    }
    Ok(Token::Counted(b, cnt))
}

fn molecule(&mut self) -> Result<Token<'a>, Box<dyn Error>> {
    let mut v: Vec<Box<Token<'a>>> = Vec::new();
    let beg = self.token_i;
    loop {
        match self.t {
            Token::Open(_) | Token::Atom(_) => v.push(Box::new(self.counted()?)),
            _ => break,
        }
    }
    let end = self.token_i;
    Ok(Token::Molecule(v, std::str::from_utf8(&self.s[beg..end]).unwrap()))
}

fn component(&mut self) -> Result<Token<'a>, Box<dyn Error>> {
    let mut koef = 1;
    if let Token::Num(kk) = self.t {
        koef = kk;
        self.next_token()?;
    }
    let m = self.molecule()?;
    Ok(Token::Component(koef, Box::new(m)))
}

```



```

}
fn side(&mut self) -> Result<Token<'a>, Box<dyn Error>> {
    let mut comps: Vec<Box<Token<'a>>> = Vec::new();
    comps.push(Box::new(self.component()?));
    while let Token::Plus = self.t {
        self.next_token()?;
        comps.push(Box::new(self.component()?));
    }
    Ok(Token::Side(comps))
}
fn equation(&mut self) -> Result<Token<'a>, Box<dyn Error>> {
    let s1 = self.side()?;
    if let Token::Equals = self.t {
        self.next_token()?;
        let s2 = self.side()?;
        Ok(Token::Equation(Box::new(s1), Box::new(s2)))
    } else {
        Err(Box::new(ParseError::new("= expected", self.i)))
    }
}
fn parse(&mut self) -> Result<Token<'a>, Box<dyn Error>> {
    self.next_token()?;
    let root = self.molecule()?;
    if let Token::End = self.t {
        Ok(root)
    } else {
        Err(Box::new(ParseError::new("EOF expected", self.i)))
    }
}
fn parse_fragment(&mut self) -> Result<Token<'a>, Box<dyn Error>> {
    self.molecule()
}

fn count_atoms_rec(&mut self, node: &Token<'a>, mult: u64, cnt: &mut BTreeMap<&'a str, u64>)
    match node {
        Token::Atom(s) => {
            *cnt.entry(*s).or_default() += mult;
        },
        Token::Bracket(_, t, _) => self.count_atoms_rec(t.as_ref(), mult, cnt),
        Token::Counted(t, v) => self.count_atoms_rec(t.as_ref(), mult * *v, cnt),
        Token::Molecule(v, _) => {
            for i in 0..v.len() {
                self.count_atoms_rec(v[i].as_ref(), mult, cnt);
            }
        },
        Token::Component(x, t) => {
            self.count_atoms_rec(t.as_ref(), mult * *x, cnt);
        },
        Token::Side(v) => {
            for i in 0..v.len() {
                self.count_atoms_rec(v[i].as_ref(), mult, cnt);
            }
        },
        _ => panic!(),
    }
}

```

```

fn is_molecule_start(c: u8) -> bool {
    c == b'[' || c == b'(' || (c >= b'A' && c <= b'Z')
}

fn skip_chars(&mut self) {
    while self.i < self.len && !Self::is_molecule_start(self.s[self.i]) {
        self.i += 1;
    }
}

fn is_eof(&mut self) -> bool {
    self.i == self.len
}

fn drop_token(&mut self) {
    self.i = self.token_i;
}

fn count_atoms(&mut self, node: &Token<'a>, cnt: &mut BTreeMap<&'a str, u64>) {
    match node {
        Token::Side(_) => {
            self.count_atoms_rec(node, 1, cnt);
        }
        Token::Molecule(_, _) => {
            self.count_atoms_rec(node, 1, cnt);
        }
        _ => panic!(),
    }
}

}

fn main() -> Result<(), Box<dyn std::error::Error>> {
    let mut buf = String::new();
    _ = std::io::stdin().read_to_string(&mut buf)?;
    let mut p = Parser::new(&buf);
    let mut cnt: BTreeMap<&str, u64> = BTreeMap::new();

    while !p.is_eof() {
        p.skip_chars();
        if p.is_eof() {
            break;
        }
        p.next_token().unwrap();
        let res = p.parse_fragment();
        p.drop_token();
        if let Ok(t) = res {
            p.count_atoms(&t, &mut cnt);
        }
    }

    for (&k, &v) in &cnt {
        println!("{}", k, v);
    }

    Ok(())
}

```

}

Задача 6. В далёкой галактике

Критерий оценивания решений задачи 6

Давным-давно, в далёкой-далёкой галактике... опубликовали обновлённую звёздную карту. На этой карте звёздная система планеты Корусант (Coruscant) помещена в точку начала координат, а положения остальных звёзд заданы в координатах объекта на небесной сфере и расстояниях в единицах измерения звёздного расстояния (световые года или парсеки) от Корусант.

Для заданного маршрута найдите его длину в световых годах. Предполагается, что в далёкой-далёкой галактике используются те же самые единицы измерения, что на Земле.

Формат ввода: На стандартном потоке ввода задаётся звёздный каталог. Каждая звезда описывается своим именем, склонением, прямым восхождением и расстоянием. Каждое описание находится на отдельной строке. Строка — завершает звёздный каталог. Далее на одной строке записывается список звёздных систем на маршруте.

Формат вывода: На стандартный поток вывода выведите длину маршрута в световых годах. Число выводите в формате с 10 значащими цифрами.

```
Tatooine -16°42'58.02" 06h45m8.92s 2.64pc
Naboo -09°27'29.7312" 03h32m55.84496s 10.4751y
Hoth +89°15'50.8" 02h31m49.09s 136.90pc
--
Tatooine Naboo
Вывод примера №1:
7.837631935
```

Код возможного решения

```
#include <iostream>
#include <string>
#include <cstdio>
#include <cctype>
#include <cstring>
#include <cmath>
#include <map>

using namespace std;

constexpr double PI = 3.1415926535897932384626433832795028841971;
constexpr double DECL_SEC_TO_RAD = .000004848136811095359935899141;
constexpr double DECL_MIN_TO_RAD = .000290888208665721596153948461;
constexpr double DECL_GRAD_TO_RAD = .017453292519943295769236907684;
constexpr double RA_SEC_TO_RAD = .000072722052166430399038487115;
constexpr double RA_MIN_TO_RAD = .004363323129985823942309226921;
constexpr double RA_HOUR_TO_RAD = .261799387799149436538553615273;
constexpr double PC_TO_LY = 3.261563777167433584426624871949;

struct Star
{
    string name;
    double x{}, y{}, z{};
};

int main()
{
    string buf;
    map<string, Star> stars;
```

```

stars.insert({"Coruscant", Star{"Coruscant", 0, 0, 0}});
while (getline(cin, buf)) {
    size_t l = buf.length();
    while (l > 0 && isspace((unsigned char) buf[l-1])) --l;
    buf.resize(l);
    if (buf == "--") {
        break;
    }
    char *name = nullptr;
    char *unit = nullptr;
    double dg{}, dm{}, ds{}, rah{}, ram{}, ras{}, dist{};
    sscanf(buf.c_str(), "%ms %lf°%lf' %lf" %lfh%lfm%lfs %lf%ms", &name, &dg, &dm, &ds, &rah,
    if (!strcmp(unit, "pc")) {
        dist *= PC_TO_LY;
    }
    free(unit); unit = nullptr;
    double ra = rah * RA_HOUR_TO_RAD + ram * RA_MIN_TO_RAD + ras * RA_SEC_TO_RAD;
    double decl = 0;
    if (dg >= 0) {
        decl = dg * DECL_GRAD_TO_RAD + dm * DECL_MIN_TO_RAD + ds * DECL_SEC_TO_RAD;
    } else {
        decl = dg * DECL_GRAD_TO_RAD - dm * DECL_MIN_TO_RAD - ds * DECL_SEC_TO_RAD;
    }
    double z = sin(decl) * dist;
    double prj = cos(decl);
    double x = prj * cos(ra) * dist;
    double y = prj * sin(ra) * dist;
    stars.insert({ name, Star{ name, x, y, z } });
    free(name); name = NULL;
}
string pos;
cin >> pos;
double dist = 0;
auto cur = stars[pos];
while (cin >> pos) {
    auto next = stars[pos];
    auto dx = cur.x - next.x;
    auto dy = cur.y - next.y;
    auto dz = cur.z - next.z;
    dist += hypot(dx, dy, dz);
    cur = next;
}
printf("%.10g\n", dist);
}

```