

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 определяются самые удачные её решения, отправленные участником. Технические баллы за них суммируются. Полученная сумма умножается на корректировочный коэффициент $\frac{1}{7}$ и округляется до целого. Результат определяет итоговую оценку.

Условия задачи 1

Когда-то 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: $2026_{10} = 38 * 52^1 + 50 * 52^0 = my$. 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: $2026_{10} = 0010.0000.0010.0110_{bcd}$. 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):

запись цифры	<i>N</i>	<i>I</i>	<i>II</i>	<i>III</i>	<i>IV</i>	<i>V</i>	<i>VI</i>	<i>VII</i>
значение	0	1	2	3	4	5	6	7
запись цифры	<i>VIII</i>	<i>IX</i>	<i>X</i>	<i>XI</i>	<i>XII</i>	<i>XIII</i>	<i>XIV</i>	<i>XV</i>
значение	8	9	10	11	12	13	14	15
запись цифры	<i>XVI</i>	<i>XVII</i>	<i>XVIII</i>	<i>XIX</i>	<i>XX</i>	<i>XXI</i>	<i>XXII</i>	<i>XXIII</i>
значение	16	17	18	19	20	21	22	23
запись цифры	<i>XXIV</i>	<i>XXV</i>	<i>XXVI</i>	<i>XXVII</i>	<i>XXVIII</i>	<i>XXIX</i>	<i>XXX</i>	<i>XXXI</i>
значение	24	25	26	27	28	29	30	31
запись цифры	<i>XXXII</i>	<i>XXXIII</i>	<i>XXXIV</i>	<i>XXXV</i>	<i>XXXVI</i>	<i>XXXVII</i>	<i>XXXVIII</i>	<i>XXXIX</i>
значение	32	33	34	35	36	37	38	39
запись цифры	<i>XL</i>	<i>XLI</i>	<i>XLII</i>	<i>XLIII</i>	<i>XLIV</i>	<i>XLV</i>	<i>XLVI</i>	<i>XLVII</i>
значение	40	41	42	43	44	45	46	47
запись цифры	<i>XLVIII</i>	<i>XLIX</i>	<i>L</i>	<i>LI</i>				
значение	48	49	50	51				

Приведем пример записи числа в римско-буквенной системе счисления: $2026_{10} = 38 \cdot 52^1 + 50 \cdot 52^0 = XXXVIII.L$

Коротышки, привычные к изобретениям Знайки, стали пользоваться римско-буквенной системой счисления. Один лишь Незнайка, как всегда, творчески подошёл к делу. Однажды Знайка заглянул в исписанный Незнайкой листок, схватился за голову и пришёл в ужас. Он увидел, что в римско-буквенной записи чисел Незнайка нарушал правила римской системы счисления. Например, записывая в разряд значение 6, Незнайка мог написать *VI* по правилам римской системы или по своей прихоти – *IIIIIX*. Знайка обратился к Незнайке за разъяснениями. Как выяснилось, Незнайке не понравилось, что в римской системе не используют больше одной *I* перед *V* или перед *X*, обозначая уменьшение, например, в *IV* и в *IX*. Незнайка разрешил себе использовать столько *I* перед *V* или перед *X*, сколько ему надо (от 0 до 5 перед *V*: *IIIIIV*, *IIIIIV*, *IIIV*, *IIV*, *IV*, *V*; от 0 до 10 перед *X*: *IIIIIIIIIX*, *IIIIIIIIIX*, *IIIIIIIX*, *IIIIIIIX*, *IIIIIX*, *IIIIIX*, *IIIIIX*, *IIIX*, *IIX*, *IX*, *X*). Точно также он использовал столько *X* перед *L*, сколько ему надо (от 0 до 5 перед *L*: *XXXXXL*, *XXXXXL*, *XXXL*, *XXL*, *XL*, *L*). Никаких других изменений в римско-буквенную систему счисления Незнайка не вносил.

Знайка сразу понял, что модифицированная Незнайкой римско-буквенная система позволяет одно и то же число записать по-разному. Например:

$$2026_{10} = 38 \cdot 52^1 + 50 \cdot 52^0 = XXXVIII.L = XLVIII.L = XXXIIX.L = XLIIIX.L \text{ и т. п.}$$

Помочь разобраться с этой путаницей может только программа.

Требуется написать программу, которая считает последовательность записей чисел в модифицированной Незнайкой римско-буквенной системе. В ответе программа должна выдать записи в буквенной системе счисления самого малого числа и самого большого числа в последовательности.

Формат ввода: В первой строке вводится натуральное число K в десятичной записи: $0 < K < 501$. В последующих K строках содержится последовательность из K записей чисел A_i в модифицированной Незнайкой римско-буквенной системе счисления. В каждой строке содержится запись одного числа. Длина каждой такой записи не более чем 150.

Формат вывода: В первой строке выводится запись в буквенной системе счисления элемента последовательности A_{min} – минимального среди чисел A_i . Во второй строке выводится запись в буквенной системе счисления элемента последовательности A_{max} – максимального среди чисел A_i . Если $A_{min} = A_{max}$, то дважды выводится запись одного и того же числа. При выводе незначащие нулевые разряды не выводятся. Например, запись нуля в буквенной системе выводится как **A**

3
IIIIIV
XXLIIX.L
N

A
my

2
IIIIIV
XXXXXL

A
A[illegible]

```
zyyyyyyyyyyyyyyyyyyyB  
zyyyyyyyyyyyyyyyyyyyB
```

В решении стоит использовать массив из 75 элементов со значениями от 0 до 51 как внутреннее представление чисел, записанных 75 разрядами рассматриваемой системы. Для удобства сравнения старшие разряды стоит хранить в начальных элементах массива. В старших разрядах могут быть незначимые нули. Программа будет в цикле считывать запись очередного числа последовательности, осуществлять перевод записи во внутреннее представление, сравнивать текущее число с текущим максимумом и с текущим минимумом, обновлять текущий максимум, если считанное число больше, обновлять текущий минимум, если считанное число меньше. После окончания ввода найденный минимум и найденный максимум будут переводиться в буквенную систему счисления и выводиться как результаты.

```

program ROMLETTERS11(input, output);
type    number = array [1..75] of byte;
var     K, I : word;
        RESMIN, RESMAX, 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..25] of byte;
    I, J : word; CH : char; D : byte;
begin
    for I := 1 to 25 do BUFFER[I] := 255;
    if not (eoln or eof) then begin
        read(CH);
        I := 1;
        D := roman2Decimal(CH);
        while (D < 52) and (I < 26) do begin
            BUFFER[I] := D;
            if not (eoln or eof) then begin
                read(CH);
                D := roman2Decimal(CH);
                I := I + 1;
            end
            else D := 255
        end;
        if (I = 1) and (BUFFER[1] > 51) then N := 255
        else begin
            I := 1;
            N := 0;
            while (I < 26) and (BUFFER[I] < 52) do begin
                J := 1;
                N := N + BUFFER[I];
                if (I > J) and (BUFFER[I] = 50) and (BUFFER[I - J] = 10) then
                    while (I > J) and (BUFFER[I - J] = 10) do begin
                        N := N - 20;
                        J := J + 1;
                    end
                else if (I > J) and (BUFFER[I] > 1) and (BUFFER[I - J] = 1) then
                    while (I > J) and (BUFFER[I - J] = 1) do begin
                        N := N - 2;
                        J := J + 1;
                    end;
                I := I + 1;
            end
        end
        end
    else N := 255
end;
procedure writeNumber(var A : number);
var I, J : word;
begin
    I := 1;
    while (I < 75) and (A[I] = 0) do I := I + 1;
    for J := I to 75 do write(decimal2Letter(A[J]))
end;
procedure readNumber(var A : number);
var CH : byte; I, J : word;

```

```

begin for I := 1 to 75 do A[I] := 0;
    readRoman(CH);
    I := 1;
    J := 0;
    while (I <= 75) 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 := 76
    end;
    for I := J downto 1 do A[75 - J + I] := A[I];
    for I := 1 to 75 - J do A[I] := 0;
end;

begin readln(K);
    readNumber(RESMIN);
    RESMAX := RESMIN;
    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(RESMAX, AI, 75) < 0) then RESMAX := AI
        else if (CompareByte(AI, RESMIN, 75) < 0) then RESMIN := AI
    end;
    writeNumber(RESMIN);
    writeln;
    writeNumber(RESMAX)
end.

```

Задача 2. Незнайка и простые бразильские числа

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 определяются самые удачные её решения, отправленные участником. Технические баллы за них суммируются. Полученная сумма умножается на корректировочный коэффициент $\frac{1}{7}$ и округляется до целого. Результат определяет итоговую оценку.

Условия задачи 2

Незнайка бродил по интернету и наткнулся на архив задач Международной иберо-американской олимпиады. Его внимание привлекла задача о простых бразильских числах. Незнайка выяснил, что простое бразильское число – это простое число, которое в некоторой позиционной системе по основанию b , где $b > 1$, записывается тремя или более чем тремя единицами, при этом никаких других цифр кроме единиц в записи числа нет. Например, $7 = 111_2$, $13 = 111_3$, $31 = 11111_2 = 111_5$. Но оказалось, что такую запись могут иметь некоторые составные числа. Например, $21 = 111_4$, $111 = 111_{10}$. Такие числа к простым бразильским числам не относятся.

Незнайка очень увлёкся простыми бразильскими числами. Буквально про каждое встреченное им число он хотел знать, является ли оно простым бразильским числом. Помогите Незнайке и составьте для него программу.

Программа считывает десятичное натуральное ненулевое число N , а затем последовательность из N десятичных натуральных ненулевых чисел $A[i]$. Программа находит все простые бразильские числа в последовательности. Среди них она находит простое бразильское число, которое встречается в последовательности наибольшее количество раз. Если таких чисел оказывается несколько, то программа выбирает из них наибольшее. Найденное число программа выводит. Если простых бразильских чисел в последовательности нет, то программа выводит 0.

Формат ввода: В первой строке содержится десятичное натуральное число N : $0 < N < 2001$. Во второй строке содержится последовательность из N десятичных натуральных чисел $A[i]$: $0 < A[i] < 160001$, $i = 1, \dots, N$.

Формат вывода: Выводится десятичная запись без незначащих нулей простого бразильского числа, встречающегося в последовательности наибольшее количество раз. Если есть несколько искоемых чисел, то выводится наибольшее из них. При отсутствии простых бразильских чисел в последовательности выводится 0.

Ввод примера №1:		Ввод примера №2:		Ввод примера №3:
4		6		1
31 7 31 7		31 13 31 13 7 13		21
Вывод примера №1:		Вывод примера №2:		Вывод примера №3:
31		13		0

Решение задачи 2

В диапазон, заданный в условии задачи, попадает лишь 99 простых бразильских чисел: 7, 13, 31, ..., 158803. Подробнее см. на странице про последовательность A085104 в OEIS (On-Line Encyclopedia of Integer Sequences): <https://oeis.org/A085104>. Для получения эффективной программы следует самостоятельно рассчитать эти числа. Массив из них будет использоваться для быстрой проверки того, является ли число простым бразильским.

В решении используем массив BP из предварительно рассчитанных простых бразильских чисел. Также используем массив C из 99 числовых значений. Каждый элемент массива C будет хранить

количество встреченных вхождений соответствующего простого бразильского числа в последовательность. Массив C инициализируем нулевыми значениями. В цикле будем считывать очередное число $A[i]$. Бинарным поиском по массиву BP будем проверять, является ли число простым бразильским. При успешной проверке получим индекс числа в массиве BP . У элемента массива C с тем же индексом увеличим значение на единицу. После обработки последовательности запустим второй цикл, в котором найдём наибольший argmax массива C . Если максимум массива C равен нулю, то это означает, что простых бразильских чисел в последовательности не было, и следует вывести 0. В другом случае используем найденный argmax как индекс в массиве BP и выводим искомое число.

Код возможного решения задачи 2

```

program BRAZILPRIMES11(input, output);
const NUMBP =99;
    BP: array [1 .. NUMBP] of dword = (7, 13, 31, 43, 73, 127, 157, 211,
    241, 307, 421, 463, 601, 757, 1093, 1123, 1483, 1723, 2551, 2801,
    2971, 3307, 3541, 3907, 4423, 4831, 5113, 5701, 6007, 6163, 6481,
    8011, 8191, 9901, 10303, 11131, 12211, 12433, 13807, 14281, 17293,
    19183, 19531, 20023, 20593, 21757, 22621, 22651, 23563, 24181, 26083,
    26407, 27061, 28057, 28393, 30103, 30941, 31153, 35533, 35911, 37057,
    37831, 41413, 42643, 43891, 46441, 47743, 53593, 55933, 55987, 60271,
    60763, 71023, 74257, 77563, 78121, 82657, 83233, 84391, 86143, 88741,
    95791, 98911, 108571, 110557, 113233, 117307, 118681, 121453, 123553,
    127807, 131071, 136531, 143263, 145543, 147073, 154057, 156421, 158803);
var A, RESULT : dword;
    N, J : word;
    I, ARGM : byte;
    C: array [1 .. NUMBP] of word;
function binSearch(A : dword) : byte;
var L, M, H : byte;
begin
    L := 1;
    H := NUMBP;
    while L <= H do
    begin
        M := (L + H) div 2;
        if BP[M] > A then
        begin
            H := M - 1;
        end
        else if BP[M] < A then
        begin
            L := M + 1;
        end
        else
        begin
            break;
        end;
    end;
    if (M < NUMBP) AND (BP[M + 1] <= A) then binSearch := M + 1
    else if (M > 1) AND (BP[M] > A) then binSearch := M - 1
    else binSearch := M
end;

begin
    for I := 1 to NUMBP do C[I] := 0;

```

```

readln(N);
for J := 1 to N do begin
    read(A);
    I := binsearch(A);
    if A = BP[I] then C[I] := C[I] + 1
end;
ARGM := NUMBP;
for I := NUMBP-1 downto 1 do begin
    if C[I] > C[ARGM] then ARGM := I;
end;
if C[ARGM] = 0 then RESULT := 0
else RESULT := BP[ARGM];
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 определяются самые удачные её решения, отправленные участником. Технические баллы за них суммируются. Полученная сумма умножается на корректировочный коэффициент $\frac{1}{7}$ и округляется до целого. Результат определяет итоговую оценку.

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 стал квадратом из $N * N$ одинаковых квадратных участков. На каждом участке был выстроен один домик коротышки. Домики были перенумерованы согласно бустрофедоническому способу, названному так из-за схожести формой борозды, оставляемой при вспашке поля плугом, который тянет бык. Коротышки согласились с реновацией, предложенной Знайкой. Один лишь Незнайка решил разузнать, как на самом деле обстоят дела в Солнечном городе. Оказалось, что домики там нумеровались спиральным способом, а вовсе даже не бустрофедоническим. Домик с номером 0 (ноль) располагался в северо-западном углу (на плане это верхний левый угол). От него нумерация шла вдоль вертикальной стороны квадрата по направлению вниз (или против хода часовой стрелки). Достигнув нижнего края, нумерация продолжалась вдоль горизонтальной стороны квадрата по направлению вправо. Достигнув правого края, она следовала вдоль вертикальной стороны квадрата по направлению вверх. Дальнейший её ход замыкал обход периметра квадрата против часовой стрелки. Потом следовал ещё один виток спирали по периметру внутреннего квадрата $(N - 2) * (N - 2)$. И так далее. Например, при $N = 7$ результат нумерации был бы таким:

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

Эти сведения Незнайка тут же сообщил Знайке и стал требовать соблюдения правил нумерации домиков Солнечного города. Знайка обещал рассмотреть новую реновацию – ререновацию, как он её назвал, – только в том случае, если будет надёжный способ узнать номер каждого домика по его расположению. В который раз Незнайке нужна помощь с составлением программы.

Требуется написать программу, которая считывает N – количество участков вдоль одной стороны города. Затем программа считывает I – номер вертикального ряда домиков и J – номер горизонтального ряда домиков. Рядам присвоены номера, начиная с 1, слева направо для вертикальных рядов и сверху вниз – для горизонтальных. На пересечении рядов, с номерами I и J находится домик, номер которого в спиральной нумерации следует найти. Программа находит и выводит результат – искомый номер домика. Например, если $N = 7$, как выше, и $I = 4$, а $J = 2$, то искомым номером будет 38.

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

Формат ввода: В первой строке вводится натуральное число N в десятичной записи: $0 < N < 2^{25} + 1$. Во второй строке вводится натуральное число I : $0 < I < N + 1$. В третьей строке вводится натуральное число J : $0 < J < N + 1$.

Формат вывода: Выводится десятичная запись искомого номера домика согласно спиральной нумерации. При выводе незначащие нулевые разряды не выводятся.

Ввод примера №1:		Ввод примера №2:		Ввод примера №3:
1		2		7
1		2		4
1		1		2
Вывод примера №1:		Вывод примера №2:		Вывод примера №3:
0		3		38

Решение задачи 3

Сначала определяется K – номер витка спирали, во время прохода по которому будет пронумерован домик на пересечении рядов с номерами I и J . Он равен увеличенному на 1 расстоянию от домика до ближайшего края квадрата (вертикального или горизонтального – того, что окажется ближе). В примере №3 $K = \min(I, J, (N + 1 - I), (N + 1 - J)) = 2$. Номер M , назначенный домику в начале витка, определяется как сумма первых $K - 1$ элементов последовательности: $4 * (N - 1), 4 * (N - 3), \dots, 12$ для чётных N или $4 * (N - 1), 4 * (N - 3), \dots, 8$ для нечётных N . Выведем формулу для M . При $K = 1$ $M = 0$. При больших K $M = 4 * (N - 1) - 4 * 2 * 0 + 4 * (N - 1) - 4 * 2 * 1 + \dots + 4 * (N - 1) - 4 * 2 * (K - 2) = 4 * ((N - 1) * (K - 1) - (K - 2) * (K - 1)) = 4 * (K - 1) * (N - K + 1)$. В примере №3 $M = 4 * 1 * 6 = 24$. Имеются 4 случая расположения домика:

- 1) Он находится на начальном участке витка – в вертикальном отрезке, проходимом сверху вниз. Искомый номер находится как $M + (J - K)$.
- 2) Он находится на втором участке витка – в горизонтальном отрезке, проходимом слева направо (в таком случае $N + 1 - J = K$). Искомый номер находится как $M + (N + 1 - 2 * K) + (I - K) = M + N + I - 3 * K + 1$.
- 3) Он находится на третьем участке витка – в вертикальном отрезке, проходимом снизу вверх (в таком случае $N + 1 - I = K$). Искомый номер находится как $M + 3 * (N + 1 - 2 * K) - (J - K) = M + 3 * N - J - 5 * K + 3$.
- 4) Он находится на последнем участке витка – в горизонтальном отрезке, проходимом справа налево (в таком случае $K = J$). В примере №3 как раз такой случай. Искомый номер находится как $M + 4 * (N + 1 - 2 * K) - (I - K) = M + 4 * N - I - 7 * K + 4 = 24 + 28 - 4 - 14 + 4 = 38$.

Код возможного решения задачи 3

```

program SPIRAL11(input, output);
var N, I, J, K, M, RESULT: qword;
function MIN4(A1, A2, A3, A4: qword): qword;
begin
    if (A1 > A2) then A1 := A2;
    if (A3 > A4) then A3 := A4;
    if (A1 > A3) then MIN4 := A3 else MIN4 := A1
end;

begin
    readln(N);
    readln(I);

```

```

read(J);
K := MIN4(I, J, (N+1-I), (N+1-J));
M := 4 * (K - 1) * (N - K + 1);
if (K = I) then RESULT := M + J - K
else if (N + 1 - J) = K then RESULT := M + N + I - 3 * K + 1
else if (N + 1 - I) = K then RESULT := M + 3 * N - J - 5 * K + 3
else RESULT := M + 4 * N - I - 7 * K + 4;
write(RESULT)
end.

```

Задача 4. Очередь на Луну

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 определяются самые удачные её решения, отправленные участником. Технические баллы за них суммируются. Полученная сумма умножается на корректировочный коэффициент $\frac{1}{7}$ и округляется до целого. Результат определяет итоговую оценку.

Условия задачи 4

Коротышки из Цветочного города готовятся отправиться в путешествие на Луну. Знайка поручил Незнайке, выстроить всех коротышек в очередь на посадку в космический корабль так, чтобы они были упорядочены по росту: первым должен стоять самый высокий, последним – самый низкий. Однако, пока Незнайка витал в облаках, все перепутались и встали в очередь, кто как хотел. Строгий Знайка решил навести порядок и наказать всех тех, кто встал в очередь не по порядку. Знайка решил выбрать самую длинную цепочку коротышек, которые уже стоят (не обязательно рядом) в правильном порядке (по невозрастанию роста). Всех остальных, кто не входит в эту цепочку, он будет наказывать, заставляя учить лунные правила. Помогите Знайке узнать, каково количество коротышек, которые окажутся вне цепочки и будут наказаны.

Требуется составить программу, которая считывает натуральное число N : $0 < N < 10001$ – общее количество коротышек в очереди, а затем N натуральных чисел $H[i]$: $0 < H[i] < 1000000001$. $H[i]$ – рост коротышки, стоящего на i -ом месте в очереди. Программа находит наименьшее целое число K ($-1 < K < N$) – количество коротышек, которые не входят в цепочку из коротышек, стоящих в очереди на местах с номерами i_1, i_2, \dots, i_{N-K} , где $i_1 < i_2 < \dots < i_{N-K}$, и $H[i_1], H[i_2], \dots, H[i_{N-K}]$ является невозрастающей, т. е. $H[i_j] > H[i_{j+1}]$ или $H[i_j] = H[i_{j+1}]$ для любого j ($0 < j < N - K$).

Формат ввода: В первой строке задано число N : $0 < N < 10001$. Во второй строке заданы N натуральных чисел $H[i]$: $0 < H[i] < 1000000001$.

Формат вывода: Выводится искомое число K : $-1 < K < N$.

Ввод примера №1:		Ввод примера №2:		Ввод примера №3:
6		5		5
6 28 5 5 29 3		1 4 10 8 5		2 2 2 1 1
Вывод примера №1:		Вывод примера №2:		Вывод примера №3:
2		2		0
Пояснение:		Пояснение:		Пояснение:
Цепочка: 6 5 5 3		Цепочка: 10 8 5		Все выстроились верно.
Вне цепочки: 29 28		Вне цепочки: 1 4		

Пояснения к решению задачи 4

Вместо того чтобы искать наименьшее количество наказуемых, будем искать наибольшее количество тех, кого наказывать не нужно. Тогда ответ можно получить как $ans = N - LnIS$, где $LnIS$ - это длина наибольшей невозрастающей подпоследовательности. Вместо того, чтобы вычислять длину наибольшей невозрастающей подпоследовательности можно развернуть массив и свести задачу к вычислению длины наибольшей неубывающей подпоследовательности $LnDS$. При данных ограничениях на $N \leq 10^4$ задачу можно решить с помощью метода динамического программирования. Для этого определим «динамику» $dp[i]$ как минимальное значение, на которое

может заканчиваться неубывающая подпоследовательность длины i . Изначально $dp[0] = -\infty$ а $dp[1] = dp[2] = \dots = \infty$. Рассматриваемая «динамика» является монотонной, а значит в ней можно бинарным поиском по $H[i]$ находить точную верхнюю границу, куда можно внести элемент $H[i]$, чтобы образовать неубывающую подпоследовательность. Для каждого значения массива нужно запустить бинарный поиск по «динамике».

Код возможного решения задачи 4

```
#include <bits/stdc++.h>

using namespace std;

signed
main(void)
{
    int n;
    cin >> n;

    if (n == 0) {
        cout << 0 << endl;
        return 0;
    }

    vector<int> a(n);
    for (auto &i : a) {
        cin >> i;
    }

    reverse(a.begin(), a.end());

    const int INF = 1e9 + 1e8;
    vector<int> d(n + 1, INF);
    d[0] = -INF;

    for (int i = 0; i < n; i++) {
        int l = upper_bound(d.begin(), d.end(), a[i]) - d.begin();
        if (d[l - 1] <= a[i] && a[i] < d[l]) {
            d[l] = a[i];
        }
    }

    int ans = 0;
    for (int l = 0; l <= n; l++) {
        if (d[l] < INF) {
            ans = l;
        }
    }

    int punish = n - ans;
    cout << punish << endl;

    return 0;
}
```

Задача 5. Лунная прогулка

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 определяются самые удачные её решения, отправленные участником. Технические баллы за них суммируются. Полученная сумма умножается на корректировочный коэффициент $\frac{1}{7}$ и округляется до целого. Результат определяет итоговую оценку.

Условия задачи 5

На Луне, куда прилетел Незнайка, поверхность покрыта лунными горами. Каждая гора имеет три характеристики: 1) x – координата долготы (восток-запад), 2) y – координата широты (север-юг), 3) h – высота горы над лунной равниной.

Незнайка любит покорять вершины, но так как гор много, он придумал себе специальные ограничения, чтобы не запутаться. Он может стартовать с любой горы. Дальше он хочет прыгать с горы на гору, но только если выполняются такие правила:

1. Новая гора должна быть правее (восточнее) текущей: $x_2 > x_1$.
2. Новая гора должна быть выше на карте (севернее) текущей: $y_2 > y_1$.
3. Новая гора должна быть ниже той, с которой прыгает Незнайка: $h_2 < h_1$.

После прыжка Незнайка останавливается на новой горе и снова может прыгать дальше, если есть подходящая цель. Помогите Незнайке узнать, какое максимальное количество гор он сможет посетить за одну такую прогулку?

Формат ввода: Первая строка содержит одно целое число n ($1 \leq n \leq 5000$) – количество гор. В следующих n строках содержатся по три целых числа в каждой строке: $0 \leq x_i, y_i, h_i \leq 10^9$ – координаты и высота i горы.

Формат вывода: Выводится одно целое число – максимальное количество гор, которые Незнайка может посетить за одну прогулку.

Ввод примера №1:

```
5
0 0 10
1 2 9
2 1 8
3 3 7
4 4 6
```

Вывод примера №1:

```
4
```

Ввод примера №2:

```
6
0 0 5
1 1 6
2 2 4
3 3 3
4 4 2
5 5 1
```

Вывод примера №2:

5

Ввод примера №3:

10

0 0 5

1 1 4

2 2 3

3 3 10

4 4 9

5 5 8

0 0 4

0 100 15

6 6 7

7 7 7

Вывод примера №3:

4

Решение задачи 5

Любой допустимый прыжок во время прогулки всегда увеличивает x , значит после сортировки по возрастанию x любая ранее посещенная гора обязательно стоит раньше текущей горы. Отсортируем все горы по возрастанию x . Пусть $dp[i]$ – максимальное число гор в маршруте, оканчивающемся в i -й горе в этом порядке. Тогда $dp[i] = 1 + \max(dp[j])$ по всем $j < i$, для которых одновременно $(x[j] < x[i]) \text{ and } (y[j] < y[i]) \text{ and } (h[j] > h[i])$. Ответом тогда будет $\max(dp[i])$.

Код возможного решения задачи 5

```
#include <bits/stdc++.h>
using namespace std;

struct Mountain {
    long long x, y, h;
};

bool cmp(const Mountain &A, const Mountain &B) {
    if (A.x != B.x) return A.x < B.x;
    if (A.y != B.y) return A.y < B.y;
    return A.h < B.h;
}

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);

    int n;
    cin >> n;

    vector<Mountain> a(n);
    for (int i = 0; i < n; ++i) {
        cin >> a[i].x >> a[i].y >> a[i].h;
    }

    sort(a.begin(), a.end(), cmp);

    vector<int> dp(n, 1);
    int ans = 1;
```

```

for (int i = 0; i < n; ++i) {
    for (int j = 0; j < i; ++j) {
        if (a[j].x < a[i].x &&
            a[j].y < a[i].y &&
            a[j].h > a[i].h) {
            dp[i] = max(dp[i], dp[j] + 1);
        }
    }
    ans = max(ans, dp[i]);
}

cout << ans << '\n';
return 0;
}

```


Задача 6. Морской бой

Критерий оценивания решений задачи 6

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 6

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

Перевод технических баллов в оценку

По каждой задаче определяются самые удачные её решения, отправленные участником. Технические баллы за них суммируются. Полученная сумма умножается на корректировочный коэффициент $\frac{1}{7}$ и округляется до целого. Результат определяет итоговую оценку.

Условия задачи 6

Во время одного из заседаний АО «Общество гигантских растений» Незнайка рассказал Миге про игру «Морской бой». В игре на прямоугольном поле расставляются один четырёхпалубный корабль, два трёхпалубных корабля, три двухпалубных корабля и четыре однопалубных корабля. Но, рассказывая об игре, Незнайка забыл, что корабли не могут соприкасаться, и в его версии правил корабли могут располагаться на поле как угодно. Корабли одного типа неразличимы, поэтому если поменять местами два корабля одного типа, получится та же самая расстановка.

Однажды, рассматривая тетрадку, в которой он рисовал поля для игры, Незнайка увидел, что границы между кораблями стёрлись и остался только силуэт конфигурации, то есть для каждой клетки поля известно только, был ли там корабль или нет.

Напишите программу, которая по заданному полю, на котором стёрты границы между кораблями, определит количество различных расстановок кораблей. Все клетки кораблей должны размещаться только на занятых клетках поля, и любая занятая клетка поля должна принадлежать какому-либо кораблю. Каждый корабль может размещаться на поле как горизонтально, так и вертикально.

Например, если дана конфигурация

```
.....  
.#####.  
.....
```

то существует 3 способа разместить один четырёхпалубный и два однопалубных корабля как показано ниже.

444411

144441

114444

Формат ввода: В первой строке вводятся целые числа $N_1 : (0 \leq N_1 \leq 4)$, $N_2 : (0 \leq N_2 \leq 3)$, $N_3 : (0 \leq N_3 \leq 2)$, $N_4 : (0 \leq N_4 \leq 1)$, $R : (0 < R \leq 20)$ и $C : (0 < C \leq 20)$. N_1 – это число однопалубных кораблей, N_2 – это число двухпалубных кораблей, N_3 – это число трёхпалубных кораблей, и N_4 – это число четырёхпалубных кораблей. Затем вводятся R строк, каждая из которых состоит из C символов, не считая конца строки. Символ # («решетка») обозначает клетку с кораблём, а символ . («точка») – пустую клетку. В задаваемой конфигурации поля могут использоваться меньше кораблей, чем в полной игре.

Формат вывода: Выводится одно целое число – количество различных конфигураций кораблей с заданным во вводе заполнением клеток поля. Корабли одного типа (например, однопалубные) неразличимы.

Ввод примера №1:

2 0 0 1 3 8

.....

.#####.

.....

Вывод примера №1:

3

Код возможного решения задачи 6

```
#include <iostream>
#include <string>
#include <vector>
#include <array>

using namespace std;

int rows;
int cols;
int poscount = 0;
array<int, 4> cnt;
vector<string> table;

void solve(int kind, int curr, int curc)
{
    char ship;
    int k = 0;
    for (; k < 4; ++k) {
        if (cnt[k] > 0) break;
    }
    if (k == 4) {
        ++poscount;
        return;
    }
    ++curc;
    if (curc == cols) {
        curc = 0;
        ++curr;
    }
    if (k != kind) {
        curr = 0;
        curc = 0;
    }
    for (int r = curr; r < rows; ++r) {
        for (int c = curc; c < cols; ++c) {
            curc = 0;
            if (table[r][c] != '#') continue;

            if (k == 0) {
                // 4 deck
                if (c + 3 < cols && table[r][c+1] == '#'
                    && table[r][c+2] == '#' && table[r][c+3] == '#') {
                    ship = '4';
                    table[r][c] = ship;
                    table[r][c+1] = ship;
                    table[r][c+2] = ship;
                    table[r][c+3] = ship;
                }
            }
        }
    }
}
```

```

        --cnt[k];
        solve(k, r, c);
        ++cnt[k];
        table[r][c+3] = '#';
        table[r][c+2] = '#';
        table[r][c+1] = '#';
        table[r][c] = '#';
    }
    if (r + 3 < rows && table[r+1][c] == '#'
        && table[r+2][c] == '#' && table[r+3][c] == '#') {
        ship = '4';
        table[r][c] = ship;
        table[r+1][c] = ship;
        table[r+2][c] = ship;
        table[r+3][c] = ship;
        --cnt[k];
        solve(k, r, c);
        ++cnt[k];
        table[r][c] = '#';
        table[r+1][c] = '#';
        table[r+2][c] = '#';
        table[r+3][c] = '#';
    }
} else if (k == 1) {
    // 3 deck
    if (c + 2 < cols && table[r][c+1] == '#'
        && table[r][c+2] == '#') {
        ship = '3';
        table[r][c] = ship;
        table[r][c+1] = ship;
        table[r][c+2] = ship;
        --cnt[k];
        solve(k, r, c);
        ++cnt[k];
        table[r][c+2] = '#';
        table[r][c+1] = '#';
        table[r][c] = '#';
    }
    if (r + 2 < rows && table[r+1][c] == '#'
        && table[r+2][c] == '#') {
        ship = '3';
        table[r][c] = ship;
        table[r+1][c] = ship;
        table[r+2][c] = ship;
        --cnt[k];
        solve(k, r, c);
        ++cnt[k];
        table[r][c] = '#';
        table[r+1][c] = '#';
        table[r+2][c] = '#';
    }
} else if (k == 2) {
    // 2 deck
    if (c + 1 < cols && table[r][c+1] == '#') {
        ship = '2';
        table[r][c] = ship;

```

```

        table[r][c+1] = ship;
        --cnt[k];
        solve(k, r, c);
        ++cnt[k];
        table[r][c+1] = '#';
        table[r][c] = '#';
    }
    if (r + 1 < rows && table[r+1][c] == '#') {
        ship = '2';
        table[r][c] = ship;
        table[r+1][c] = ship;
        --cnt[k];
        solve(k, r, c);
        ++cnt[k];
        table[r][c] = '#';
        table[r+1][c] = '#';
    }
} else {
    abort();
}
}
}

int main()
{
    int n1, n2, n3, n4;

    cin >> n1 >> n2 >> n3 >> n4;
    cin >> rows >> cols;
    if (n1 < 0 || n1 > 4 || n2 < 0 || n2 > 3 || n3 < 0 || n3 > 2 || n4 < 0 || n4 > 1)
        abort();
    if (rows < 1 || rows > 20 || cols < 1 || cols > 20) abort();

    string buf;
    getline(cin, buf);

    int hashcount = 0;
    for (int i = 0; i < rows; ++i) {
        getline(cin, buf);
        if (cin.eof()) abort();
        if (int(buf.length()) != cols) abort();
        for (char c : buf) hashcount += (c == '#');
        table.push_back(buf);
    }
    getline(cin, buf);
    if (!cin.eof()) abort();
    if (hashcount != n1 + 2 * n2 + 3 * n3 + 4 * n4) abort();

    cnt = array<int, 4>{ n4, n3, n2, 0 };
    solve(-1, 0, -1);

    cout << poscount << endl;
}

```

Задача 7. Космические гонки

Критерий оценивания решений задачи 7

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 7

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

Перевод технических баллов в оценку

По каждой задаче определяются самые удачные её решения, отправленные участником. Технические баллы за них суммируются. Полученная сумма умножается на корректировочный коэффициент $\frac{1}{7}$ и округляется до целого. Результат определяет итоговую оценку.

Условия задачи 7

Винтик и Шпунтик решили организовать гонки на ракетах. В гонках решили принять участие почти все коротышки, включая Незнайку и Селёдочку. В квалификационном старте побеждает тот коротышка, чья ракета наберёт самую большую скорость. Наблюдательный пункт Винтика расположен в точке Лагранжа L1 системы Земля-Луна. Для замера скорости Винтик делает два сферических снимка пространства вокруг точки наблюдения, а затем вычисляет перемещение ракет участников. Тот коротышка, чья ракета пролетела наибольшее расстояние, побеждает в квалификационном старте. Если ракета присутствует на одном снимке из двух, она дисквалифицируется из гонки.

Формат ввода: На стандартном потоке ввода задаются два сферических снимка пространства вокруг точки наблюдения. Снимки разделяются строкой ——. Каждая ракета описывается своим именем, склонением, прямым восхождением и расстоянием в км. Каждое описание находится на отдельной строке.

Формат вывода: На стандартный поток вывода напечатайте максимальное перемещение ракеты в километрах. Число выводите в формате с 10 значащими цифрами. Если во входных данных нет ракет, либо все ракеты дисквалифицированы, то выведите число -1.

Ввод примера №1:

```
Незнайка -16°42'58.02" 06h45m8.92s 1.66532e6
Селёдочка -09°27'29.7312" 03h32m55.84496s 1043
Фуксия +89°15'50.8" 02h31m49.09s 1000001.45
--
Незнайка -16°42'03" 06h40m16s 1.66e6
Селёдочка -05°18'44" 03h32m55.84496s 1200
```

Вывод примера №1:

34339.0462

Код возможного решения задачи 7

```
#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;

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

    Star() = default;
    Star(string nn, double xx, double yy, double zz) :
        name(std::move(nn)),
        x{xx},
        y{yy},
        z{zz}
    {
    }
    Star(const std::string &buf);
};

Star::Star(const std::string &buf)
{
    char *name = nullptr;
    double dg{}, dm{}, ds{}, rah{}, ram{}, ras{}, dist{};
    sscanf(buf.c_str(), "%ms %lf°%lf' %lf" %lfh%lfm%lfs %lf",
        &name, &dg, &dm, &ds, &rah, &ram, &ras, &dist);
    Star::name = std::string(name);
    double ra = rah * RA_HOUR_TO_RAD + ram * RA_MIN_TO_RAD + ras * RA_SEC_TO_RAD;
    double decl = 0;
    if (signbit(dg)) {
        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;
    }
    z = sin(decl) * dist;
    double prj = cos(decl);
    x = prj * cos(ra) * dist;
    y = prj * sin(ra) * dist;
}

int main()
{
    string buf;
    map<string, Star> stars1;
    map<string, Star> stars2;

    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;
        }
        Star ss(buf);
        stars1.insert({ ss.name, ss });
    }

    while (getline(cin, buf)) {
        size_t l = buf.length();
        while (l > 0 && isspace((unsigned char) buf[l-1])) --l;
        buf.resize(l);
        Star ss(buf);
        stars2.insert({ ss.name, ss });
    }

    double dist = -1;
    for (const auto &[n, s1] : stars1) {
        if (auto it = stars2.find(n); it != stars2.end()) {
            auto dx = s1.x - it->second.x;
            auto dy = s1.y - it->second.y;
            auto dz = s1.z - it->second.z;
            auto dd = hypot(dx, dy, dz);
            if (dist < 0 || dd > dist) {
                dist = dd;
            }
        }
    }
    printf("%.10g\n", dist);
}

```