Тетра-злые числа. 11 класс. Заключительный этап

Рассмотрим последовательность чисел, называемых числами тетраначчи: $T_0 = T_1 = T_2 = 0$, $T_3 = 1, T_4 = T_3+T_2+T_1+T_0 = 1$, ..., $T_n = T_{n-1}+T_{n-2}+T_{n-3}+T_{n-4}$ для любого натурального $n \geq 4$. Числа T_4, T_5, ..., T_n, ... можно использовать для записи любого неотрицательного целого числа в так называемой тетраначчиевой системе счисления. Эта система счисления является позиционной и использует цифры 0 и 1. В этой системе ноль записывается как 0_{tetra}, один как 1_{tetra}, два как 10_{tetra}, три как 11_{tetra}. В общем случае неотрицательное целое число $N = d_k * T_{k+4} + d_{k-1} * T_{k+3} + ... + d_0 * T_4 = d_k d_{k-1} ... d_{0_{tetra}}$, где любое d_i является либо 0, либо 1, и среди любых четырёх идущих подряд d_i, d_{i-1}, d_{i-2}, d_{i-3} обязательно есть хотя бы один 0, т. е. $d_i * d_{i-1} * d_{i-2} * d_{i-3} = 0$.

Неотрицательное целое число назовём тетра-злым, если в записи этого числа в тетраначчиевой системе счисления количество единиц чётное. Примеры тетра-злых чисел: $0_{10} = 0_{tetra}$, $3_{10} = 11_{tetra}$, $732_{10} = 1110010000_{tetra}$.

Составьте программу, принимающую на вход в первой строке десятичное число N – положительное натуральное число ($1 \leq N \leq 1000$) – количество чисел, а в последующих N строках – неотрицательные целые числа A_1, A_2, ... , A_N, записанные в десятичной системе, $0 \leq A_i \leq 2^{24} - 1$. Программа подсчитывает количество чисел A_i, являющихся тетра-злыми и выводит это количество в виде неотрицательного целого числа, записанного в десятичной системе без незначащих нулей.

Формат ввода: В 1-ой строке содержится десятичное число N – положительное натуральное число ($1 \leq N \leq 1000$). В последующих N строках содержатся неотрицательные целые числа A_1, A_2, ... , A_N, записанные в десятичной системе, $0 \leq A_i \leq 2^{24} - 1$. Каждое число A_i записывается в отдельной строке.

Формат вывода: Искомое количество тетра-злых чисел выводится в виде неотрицательного целого числа, записанного в десятичной системе без незначащих нулей.

```
Ввод примера №1:
1
1
Вывод примера №1:
0
```

```
Ввод примера №2:
2
0
1
Вывод примера №2:
1
```

```
Ввод примера №3:
4
0
3
732
3
Вывод примера №3:
4
```

Решение

В решении можно запрограммировать следующие подзадачи: а) проверку того, является ли очередное число тетра-злым; б) подсчёт и вывод искомого количества. Перевод очередного числа A_i в тетраначчиевую систему реализуется жадным алгоритмом. Ради эффективности можно заранее подсчитать все числа тетраначчи, которые меньше 2^{24}. Среди этих чисел ищется наибольшее число тетраначчи, не превосходящее A_i. Если его нет, то A_i – ноль. Если оно есть, то обозначим его T_{k+4}, увеличим количество единичных разрядов в тетраначчиевой записи и вычислим разность $A_i - T_{k+4}$. Далее работаем с этой разностью и числом T_{k+3}. Если оно больше разности, то соответствующий разряд в тетраначчиевой записи будет нулевым. Если оно не меньше разности, то снова увеличим количество единичных разрядов в тетраначчиевой записи и вычислим разность $A_i - T_{k+4} - T_{k+3}$. И т. д. Дойдя до нулевой разности, проверяем на чётность количество единичных разрядов. В цикле вводим и проверяем все A_i. Ради эффективности можно, но не обязательно, использовать мемоизацию, так как числа могут повторяться. То есть, можно запоминать результаты выполненных проверок в АВЛ-дереве, где ключами будут проверяемые числа, а связанными значениями – результаты проверок. При мемоизации перед выполнением проверки сначала пробуем найти ответ в АВЛ-дереве. Если поиск неудачен, то выполняем проверку так, как описано выше, и результат вставляем в дерево. Если поиск удачен, то нам повезло, и повторно его считать не придётся.

Код возможного решения, без мемоизации проверок

```
program TETRAEVIL11(input, output);
const   NUMTETRA = 26;
        TETRA: array [1 .. NUMTETRA] of dword = (1, 2, 4, 8, 15, 29, 56, 108,
```

```
        208, 401, 773, 1490, 2872, 5536, 10671, 20569, 39648, 76424, 147312,
        283953, 547337, 1055026, 2033628, 3919944, 7555935, 14564533);
var  N, I, ANSWER : word; A : dword;
function binSearch(A : dword) : byte;
var L, M, H : byte;
begin
    L := 1;
    H := NUMTETRA;
    while L <= H do
    begin
        M := (L + H) div 2;
        if TETRA[M] > A then
        begin
            H := M - 1;
        end
        else if TETRA[M] < A then
        begin
            L := M + 1;
        end
        else
        begin
            break;
        end;
    end;
    if (M < NUMTETRA) AND (TETRA[M + 1] <= A) then binSearch := M + 1
    else if (M > 1) AND (TETRA[M] > A) then binSearch := M - 1
    else binSearch := M
end;
function isTetraEvil(A : dword): byte;
var I, NUM1 : byte;
begin
    if (A = 0) then begin NUM1 := 0 end
    else begin
        I := binSearch(A);
        NUM1 := 1;
        A := A - TETRA[I];
        while (A > 0) AND (I > 1) do begin
            I := I - 1;
            if A >= TETRA[I] then begin
                NUM1 := (NUM1 + 1) mod 2;
                A := A - TETRA[I]
            end
        end;
    end;
    isTetraEvil := (NUM1 + 1) mod 2
end;
begin
    readln(N);
    ANSWER := 0;
    for I := 1 to N do begin
        read(A);
        ANSWER := ANSWER + isTetraEvil(A)
    end;
    write(ANSWER)
end.
```

Октодерево. 11 класс. Заключительный этап

Рассмотрим способ представления трёхмерных кубических сцен, называемый октодеревом:

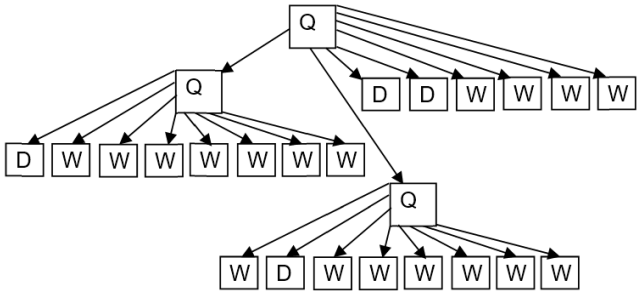
1) Если сцена целиком одноцветная, то есть является кубом, закрашенным одним цветом, то октодерево состоит из одной листовой вершины – вокселя. Линейная запись такого октодерева состоит из одного символа P: W, R, O, Y, G, C, B, V или D (W — белый, R — красный, O — оранжевый, Y — жёлтый, G — зеленый, C — голубой, B — синий, V — фиолетовый или D — черный).

2) Если в сцене есть фрагменты разного цвета, то она делится на 8 равных кубов (верхний левый ближний, верхний правый ближний, нижний левый ближний, нижний правый ближний, верхний левый дальний, верхний правый дальний, нижний левый дальний, нижний правый дальний) и представляется октодеревом, состоящим из корневой вершины и восьми поддеревьев, которые описывают части сцены – полученные при делении кубы. Пусть $T_{\text{ВЛБД}}$ – линейная запись верхнего левого ближнего поддерева, $T_{\text{ВПБД}}$ – линейная запись верхнего правого ближнего поддерева, $T_{\text{НЛБД}}$ – линейная запись нижнего левого ближнего поддерева, $T_{\text{НПБД}}$ – линейная запись нижнего правого ближнего поддерева, $T_{\text{ВЛДД}}$ – линейная запись верхнего левого дальнего поддерева, $T_{\text{ВПДД}}$ – линейная запись верхнего правого дальнего поддерева, $T_{\text{НЛДД}}$ – линейная запись нижнего левого дальнего поддерева, $T_{\text{НПДД}}$ – линейная запись нижнего правого дальнего поддерева; тогда запись всего дерева будет такой: $QT_{\text{ВЛБД}}T_{\text{ВПБД}}T_{\text{НЛБД}}T_{\text{НПБД}}T_{\text{ВЛДД}}T_{\text{ВПДД}}T_{\text{НЛДД}}T_{\text{НПДД}}$

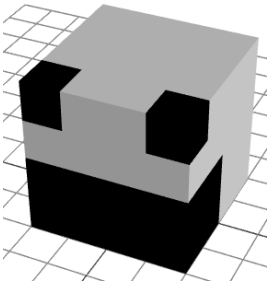
Пример:

линейная запись октодерева: $QQDWWWWWWQWDWWWWWWDDWWWW$

октодерево в виде графа:



сцена, представленная октодеревом (*на рисунке сетка дана для пояснения и в состав сцены не входит; серый цвет соответствует белым кубам, записываемым как W, и используется для контраста с фоном; задняя половина куба вся имеет цвет W; спереди внизу находятся целиком чёрные кубы, записываемые как D*):



В линейной записи октодерева все символы идут подряд. Никаких пробельных, разделительных и других символов в линейной записи октодерева не используется.

Составьте программу, на вход которой в 1-ой строке подаётся линейная запись октодерева T (длина записи не превышает 4681 символ). Программа находит цвет, в который закрашена часть октодерева T, имеющая максимальный объём. Если таких цветов больше чем один, то искомым является тот, которому соответствует символ, стоящий в алфавите ближе всего к началу. Программа в первой строке выводит символ, соответствующий найденному цвету. Далее, во второй строке, полагая, что общий объем записанной октодеревом T сцены равен 1, программа выводит объем части, закрашенной найденным цветом, в виде беззнаковой двоичной дроби без незначащих нулей.

Формат ввода: В 1-ой строке содержится последовательность символов Q, W, R, O, Y, G, C, B, V или D, составленная по правилам линейной записи октодерева. Длина 1-ой строки не превосходит 4681 символ.

Формат вывода: В первой строке вывода содержится единственный символ – один из W, R, O, Y, G, C, B, V или D. Он должен соответствовать цвету, в который закрашена часть октодерева

T с максимальным объемом. Если таких цветов несколько, то из их символов выбирается тот, который предшествует другим в алфавитном порядке. Во второй строке вывода искомый объём части октодерева записывается беззнаковой двоичной дробью без незначащих нулей (незначащим считается ноль, находящийся в разряде дробной части, начиная со 2-го, такой, что в части записи дроби справа от него не встречается ни одна 1). В записи вывода всегда присутствует один разряд в целой части и не менее чем один разряд в дробной. Разделителем между целой и дробной частью является точка. Например, 0 записывается так: 0.0 Пример записи, являющейся неверной из-за наличия двух незначащих нулей: 0.000100.

Ввод примера №1:	Ввод примера №2:	Ввод примера №3:
QQDWWWWWWWQWDWWWWWWDDWWWW	R	QWROYGCBV
Вывод примера №1:	Вывод примера №2:	Вывод примера №3:
W	R	B
0.10111	1.0	0.001

Решение

В решении можно запрограммировать следующие подзадачи: а) считывание записи октодерева и одновременный расчёт объёма частей октодерева, покрашенных в каждый из 9 цветов; б) поиск и вывод argmax, символ которого стоит раньше других по алфавиту. Запись октодерева соответствует его обходу в глубину, поэтому можно использовать такой алгоритм: 1) Помещаем в стек 1. 2) Пока стек не пуст и не достигнут конец строки выполняем тело цикла. 2.1) Считываем очередной символ в записи дерева. 2.2) Берем из стека текущую высоту H. 2.3) Если считали Q, то 8 раз помещаем в стек $H+1$, иначе к объему, закрашенному цветом C, добавляем $\frac{1}{8}^{H-1}$. По завершении расчётов ищем нужный цвет. Перенумеруем цвета так, чтобы их символы шли по алфавиту: B – 1; C – 2; D – 3; G – 4; O – 5; R – 6; V – 7; W – 8; Y – 9. Полагаем, что текущим argmax является чёрный цвет. В цикле от голубого до жёлтого сравниваем текущий максимальный объём с объёмом части, закрашенной в очередной цвет. Если текущий максимум превышен, то меняем argmax на текущий цвет. Когда всё искомое в задаче найдено, выводим результаты. При выводе объёма нужно определить количество значимых разрядов, а затем вывести их.

Код возможного решения

```
program OCTOTREE11(input, output);
const   STACKSIZE = 4690;
        SCALESIZE = 1800;
type    stack = record mass : array [1..STACKSIZE] of word; top : word end;
        bitscale = array [0..SCALESIZE] of 0..1; (* битовая шкала для двоичной дроби *)
        scalearray = array [1..9] of bitscale; (* 9 шкал, по 1 для каждого цвета *)
        cmpresult = -1..1; (* результат сравнения: -1 - <, 0 - =, 1 - > *)
var  S : stack; C : byte; I, J : word; BITSCALES : scalearray;
function char2color(C : char) : byte; (* перевод символа в цвет *)
begin case C of
        'Q': char2color := 0; (* пусть Q соответствует 0 *)
        'B': char2color := 1;
        'C': char2color := 2;
        'D': char2color := 3;
        'G': char2color := 4;
        'O': char2color := 5;
        'R': char2color := 6;
        'V': char2color := 7;
        'W': char2color := 8;
        'Y': char2color := 9;
      end (* of case *)
end;
function color2char(B : byte) : char; (* перевод цвета в символ *)
begin case B of
        0: color2char := 'Q'; (* Q соответствует 0 *)
        1: color2char := 'B';
```

```
        2: color2char := 'C';
        3: color2char := 'D';
        4: color2char := 'G';
        5: color2char := 'O';
        6: color2char := 'R';
        7: color2char := 'V';
        8: color2char := 'W';
        9: color2char := 'Y';
      end (* of case *)
end;
procedure push(var S : stack; e : word); (* операция ВСТЕК *)
begin   with S do begin
           top := succ(top);
           mass[top] := e
        end
end;
function notempty(var S : stack) : boolean; (* проверка на непустоту *)
begin   notempty := S.top > 0
end;
function look(var S : stack) : word; (* верхушка стека *)
begin   if notempty(S) then look := S.mass[S.top] else look := 0;
end;
procedure pop(var S : stack); (* операция ИЗСТЕКА *)
begin   if notempty(S) then
           with S do begin
              mass[top] := 0;
              top := pred(top);
        end;
end;
procedure add1(var B : bitscale; I : word); (* прибавление 2^(-3*I) к двоичной дроби *)
var CF : boolean; J : word;
begin   if I = 0 then B[0] := 1 else begin
           J := I * 3;
           repeat
              CF := B[J] = 1;
              if CF then B[J] := 0 else B[J]:= 1;
              J := pred(J)
           until (not CF) or (J = 0);
        end
end;
procedure readtree(var S : stack); (* чтение дерева с попутным подсчётом объёмов *)
var     CH : char; H, I, J : word;
begin   push(S, 1);
        while notempty(S) and (not eoln()) do begin
           read(CH);
           H := look(S);
           I := char2color(CH);
           pop(S);
           if (I = 0) then begin
               for J := 1 to 8 do push(S, H + 1)
           end (* then *)
           else begin
               add1(BITSCALES[I], H-1);
           end (* if *)
        end (* while *)
end;
```

```
function comparescales(var SC1, SC2 : bitscale): cmpresult; (* сравнение двух шкал *)
var I, J : word;
begin   I := 0;
        while (SC1[I] = SC2[I]) and (I < SCALESIZE) do I := succ(I);
        if SC1[I] < SC2[I] then comparescales := -1
        else if SC1[I] = SC2[I] then comparescales := 0
        else comparescales := 1
end;
function findcolor (var BITSCALES : scalearray): byte; (* поиск нужного argmax *)
var I, ARGMAX : byte;
begin   ARGMAX := 1;
        for I := 2 to 9 do
           if comparescales(BITSCALES[I], BITSCALES[ARGMAX]) = 1 then ARGMAX := I;
        findcolor := ARGMAX
end;
procedure printscale(var SC : bitscale); (* вывод шкалы как двоичной дроби *)
var I, J : word;
begin   if SC[0] = 1 then write('1.0')
        else begin
           I := SCALESIZE;
           while (SC[I] = 0) and (I > 1) do I := pred(I);
           write('0.');
           for J := 1 to I do write(SC[J])
        end
end;
(* main *)
begin   for I := 1 to 9 do
           for J := 0 to SCALESIZE do begin BITSCALES[I][J] := 0 end;
        with S do begin
           for J := 1 to STACKSIZE do begin mass[J] := 0 end;
           top := 0
        end;
        readtree(S);
        C := findcolor(BITSCALES);
        writeln(color2char(C));
        printscale(BITSCALES[C])
end.
```

Эльфийские сундуки. 11 класс. Заключительный этап

В сокровищнице короля эльфов Лихолесья Трандуила хранится множество сундуков. Сундуки занумерованы, начиная с единицы. В сундуках лежат эльфийские деньги. О каждом сундуке известна денежная сумма, хранящаяся в нём, – неотрицательное целое число. Эльфы пользуются своей собственной системой счисления для записи номеров сундуков и денежных сумм. Число 0 по-эльфийски записывается двумя символами [] – открывающей и закрывающей прямоугольными скобками. Цифра 1 по-эльфийски записывается строчной латинской буквой i. Цифра 2 по-эльфийски записывается двумя символами i(– строчной латинской буквой i и открывающей круглой скобкой. Цифра 3 по-эльфийски записывается тремя символами i((– строчной латинской буквой i и двумя открывающими круглыми скобками. Цифра 4 по-эльфийски записывается заглавной латинской буквой I. Цифра 5 по-эльфийски записывается двумя символами I(– заглавной латинской буквой I и открывающей круглой скобкой. Цифра 6 по-эльфийски записывается тремя символами I((– заглавной латинской буквой I и двумя открывающими круглыми скобками. Цифра 7 по-эльфийски записывается заглавной латинской буквой J. Цифра 8 по-эльфийски записывается двумя символами J) – заглавной латинской буквой J и закрывающей круглой скобкой. Цифра 9 по-эльфийски записывается тремя символами J)) – заглавной латинской буквой J и двумя закрывающими круглыми скобками. Цифра десять (A в позиционных системах счисления с основанием больше десяти) по-эльфийски записывается строчной латинской буквой j. Цифра одиннадцать (B в позиционных системах счисления с основанием больше одиннадцати) по-эльфийски записывается двумя символами j) – строчной латинской буквой j и закрывающей круглой скобкой. Цифра двенадцать (C в позиционных системах счисления с основанием больше двенадцати) по-эльфийски записывается тремя символами j)) – строчной латинской буквой j и двумя закрывающими круглыми скобками. Другие цифры (в том числе цифру 0) эльфы не используют.

Ещё одной особенностью эльфийской системы счисления является то, что их цифры в записи числа следуют в обратном порядке, если сравнивать с обычной позиционной системой счисления. Пусть имеется эльфийская запись числа $d_0d_1...d_{n-1}d_n$, чтобы узнать, какое ненулевое число записано, нужно вычислить: $val(d_0) * 12^0 + val(d_1) * 12^1 + ... + val(d_{n-1}) * 12^{n-1} + val(d_n) * 12^n$. Здесь val обозначает взятие значения цифры и для удобства определено в таблице:

цифра d_i	i	i(	i((	I	I(	I((	J	J)	J))	j	j)	j))
значение $val(d_i)$	1	2	3	4	5	6	7	8	9	10	11	12

Получается, что от двенадцатеричной системы счисления эльфийская система отличается порядком записи цифр числа и тем, что в ней не используется нулевая цифра, но используется цифра двенадцать. Примеры:

$0_{10} = 0_{12}$ =[]

$12_{10} = 10_{12}$ =j))

$2024_{10} = 1208_{12} = 8 * 12^0 + 12 * 12^1 + 1 * 12^2 + 1 * 12^3$ =J)j))ii

Король Трандуил хочет найти в своём подвале два сундука, таких, чтобы разность денежных сумм, лежащих в них, была максимальной. Если в подвале есть несколько искомых пар сундуков, то король Трандуил выбирает из них такую пару, что сумма номеров сундуков максимальна.

Составьте программу, принимающую на вход в первой строке десятичное число N – положительное натуральное число ($2 \le N \le 500$), записанное без знака в десятичной системе счисления – количество сундуков, а в последующих N строках – записи в эльфийской системе денежных сумм, хранящихся в сундуках с номерами от 1 до N. Известно, что в каждой записи денежной суммы не более чем 100 символов (латинских букв и скобок). Программа находит номера K и L такие, что $K < L$, абсолютное значение (модуль) разницы денежных сумм в K-ом и L-ом сундуках наибольшее и сумма $K + L$ наибольшая. Программа выводит в первой строке эльфийскую запись найденного числа K, а во второй строке – эльфийскую запись найденного числа L. Каждое число записывается в эльфийской системе.

Формат ввода: В первой строке содержится десятичное число без знака N — количество сундуков ($2 \le N \le 500$). В следующих N строках содержатся записи в эльфийской системе денежных сумм, хранящихся в сундуках, – последовательности, в которых могут встретиться только строчные и заглавные латинские буквы i, I, j, J и прямоугольные или круглые скобки (открывающие и закрывающие). Эти последовательности записаны по правилам составления записей чисел в эльфийской системе. Длины строк находятся в диапазоне от 1 до 100 включительно.

Формат вывода: В первой строке выводится эльфийская запись найденного числа K. Во второй

строке выводится эльфийская запись найденного числа L. Числа K и L таковы, что $K < L$, абсолютное значение (модуль) разницы денежных сумм в K-ом и L-ом сундуках наибольшее и сумма $K + L$ наибольшая.

```
Ввод примера №1:
2
[]
J)j))ii
Вывод примера №1:
i
i(
```

```
Ввод примера №2:
3
J)j))ii
J)j))ii
J)j))ii
Вывод примера №2:
i(
i((
```

```
Ввод примера №3:
4
J)j))ii
j))
j))
[]
Вывод примера №3:
i
I
```

Решение

В решении можно запрограммировать следующие подзадачи: 1) считывание очередной денежной суммы в эльфийской записи и представление её в виде строки специального вида, в которой эльфийские цифры заменены на латинские строчные буквы a, ..., l, перевернут порядок цифр и позиции старших незначащих разрядов заполнены нулями; 2) поэлементное сравнение двух таких строк (может выполняться функцией из стандартной библиотеки); 3) поиск наибольшего и наименьшего элементов в последовательности денежных сумм; 4) вывод номеров искомых элементов в эльфийской записи. Для решения достаточно одного прохода по последовательности, в котором совмещены посимвольный ввод денежных сумм и их обработка. Следует хранить текущий рекорд (максимум среди всех денежных сумм, которые программа успела считать) и текущий «антирекорд» (минимум среди всех денежных сумм, которые программа успела считать). Очередная сумма после считывания сравнивается с рекордом. Если рекорд не больше её, то она становится рекордом. Также очередная сумма сравнивается с «антирекордом». Если очередная сумма не больше «антирекорда», то она становится «антирекордом». Но нужно учитывать случай, когда во всех сундуках одинаковые суммы. Чтобы в этом случае не получить одинаковые номера у рекорда и «антирекорда», нужно при обработке N-ой описи не обновлять номер «антирекорда», если номер рекорда уже равен N. По окончании обработки выводятся эльфийские записи номеров рекорда и «антирекорда» по их возрастанию. Перевод в эльфийскую запись схож с переводом в двенадцатеричную систему, но отличается другим порядком цифр записи числа и тем, что не используется нулевая цифра, но используется цифра двенадцать.

Код возможного решения

```
program ELVISHCHESTS11 (input, output);
const CHESTLENGHT = 100;
type    chests = array [1..CHESTLENGHT] of char;
        answer = record number : word; chest : chests end;
var  CURCHEST : chests; N, I : word; CHECK : integer; CURMAX, CURMIN : answer;
procedure readChest(var CHEST : chests); (* ввод эльфийской записи *)
var   TEMP: chests;  CH : char; I, J : word;
begin
    for J := CHESTLENGHT downto 1 do CHEST[J] := '0';
    I := CHESTLENGHT;
    while (I > 0) and (not EOLn) do begin
        read(CH);
        case CH of
            '[': begin read(CH); break end;
            'i': begin CHEST[I] := 'a'; I := I - 1 end;
            'I': begin CHEST[I] := 'd'; I := I - 1 end;
            'J': begin CHEST[I] := 'g'; I := I - 1 end;
            'j': begin CHEST[I] := 'j'; I := I - 1 end;
            '(', ')': CHEST[I+1] := succ(CHEST[I+1]);
        end;
    end
```

```
end;
function compareChest(CHEST1, CHEST2: chests) : integer;
var     I : integer;
begin   I := CompareChar(CHEST1, CHEST2, CHESTLENGHT);
        if I > 0 then compareChest := 1
        else if I = 0 then compareChest := 0
        else compareChest := -1
end;
procedure writeElvish(I : integer); (* вывод эльфийской записи *)
begin
    if (I = 0) then write('[]')
    else while (I > 0) do
        case I mod 12 of
            1: begin write('i'); I := I div 12 end;
            2: begin write('i('); I := I div 12 end;
            3: begin write('i(('); I := I div 12 end;
            4: begin write('I'); I := I div 12 end;
            5: begin write('I('); I := I div 12 end;
            6: begin write('I(('); I := I div 12 end;
            7: begin write('J'); I := I div 12 end;
            8: begin write('J)'); I := I div 12 end;
            9: begin write('J))'); I := I div 12 end;
            10: begin write('j'); I := I div 12 end;
            11: begin write('j)'); I := I div 12 end;
            0: begin write('j))'); I := (I - 12) div 12 end;
        end;
end;
begin   readln(N);
        with CURMAX do begin
            number := 1;
            readChest(chest);
        end;
        CURMIN := CURMAX;
        for I := 2 to N do begin
            readln;
            readChest(CURCHEST);
            CHECK := compareChest(CURMAX.chest, CURCHEST);
            if (CHECK <= 0) then begin
                with CURMAX do begin
                   number := I;
                   chest := CURCHEST
            end end; (* if *)
            CHECK := compareChest(CURCHEST, CURMIN.chest);
            if (CHECK <= 0) and (CURMAX.number <> N) then
                with CURMIN do begin
                   number := I;
                   chest := CURCHEST
            end; (* if *)
        end; (* for *)
        if (CURMAX.number > CURMIN.number) then begin
            writeElvish(CURMIN.number);
            writeln;
            writeElvish(CURMAX.number) end
        else begin
            writeElvish(CURMAX.number);
            writeln;
            writeElvish(CURMIN.number)
        end (* if *)
end.
```

4

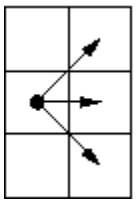
Problem 4: 4-11: Грибы отсюда

Prob Id:	`4`
Full score:	`100`
Input file name:	`input.txt` or standard input
Time limit:	`1 s`
Memory RSS limit:	`256M`
Stack limit:	`256M`

Problem 4: Грибы отсюда

Гарри, Рон и Гермиона занимаются сбором грибов в тайной комнате. Тайная комната представляет собой прямоугольник из клеток размером R строк на C столбцов, в каждой клетке которого находится некоторое количество грибов.

Гарри, Рон и Гермиона начинают движение с некоторых (возможно совпадающих) позиций в первом столбце. Каждый персонаж на каждом ходу переходит либо на клетку справа от текущей клетки, либо на клетку по диагонали справа сверху от текущей клетки, либо на клетку по диагонали справа снизу от текущей клетки. Несколько персонажей могут находиться в одной клетке. Закончить движение они должны в каких-то позициях (возможно совпадающих) в последнем столбце.



Когда персонаж проходит через клетку, он собирает все грибы, которые в ней находятся, но когда несколько персонажей находятся в одной клетке, они как-то делят все грибы между собой, то есть (естественно), грибы собрать можно только один раз.

Напишите программу, подсчитывающую максимальное количество грибов, которые могут собрать друзья, пройдя через комнату.

Input format

В первой строке входных данных находятся два целых числа R ($1 \le R \le 20$) и C ($2 \le C \le 200$) — число строк и столбцов в прямоугольнике, описывающем комнату. Во второй строке входных данных находятся три целых числа r_1, r_2, r_3 — номера строк в самом левом столбце, задающие клетки, в которых в начальный момент времени находятся ребята ($0 \le r_1, r_2, r_3 < R$).

Далее записываются R * C целых чисел $f_{i,j}$ ($0 \le f_{i,j} \le 100$), задающих количество грибов в каждой клетке. Прямоугольник комнаты описывается по строкам, то есть сначала идут C чисел, описывающих строку 0 прямоугольника, а последними идут C чисел, описывающих строку $R-1$ прямоугольника. Числа могут разделяться произвольным числом пробельных символов, то есть переводов строк, пробелов и табуляций.

Output format

Выведите целое число — максимальное количество грибов, которых могу собрать ребята, пройдя через комнату.

Examples

Input

```
5 10
0 2 4
0 0 0 0  0 0 0 0 10 0
1 1 1 1  1 1 1 1 1  1
2 2 2 2  2 2 2 2 2  2
3 3 3 3  3 3 3 3 3  3
0 0 0 10 0 0 0 0 0  0
```

Output

```
74
```

---

5 Problem 5: 5-11: Транспарант

Prob Id:	`5`
Full score:	`100`
Input file name:	`input.txt` or standard input
Time limit:	`1 s`
Memory RSS limit:	`256M`
Stack limit:	`256M`

Problem 5: Транспарант

На четвёртый день плавания на теплоходе Скрябин Остапу было поручено нарисовать агитационный транспарант. В этот день теплоход остановился в Чебоксарах, где, к удаче Остапа, он увидел уже готовый транспарант, висящий на пристани. Остап решил под покровом темноты вырезать из висящего транспаранта один сплошной кусок, который незаметно занести на теплоход, и потом спокойно разрезать этот кусок по отдельным буквам, чтобы сложить полученные буквы в нужный ему агитационный транспарант. Остап хочет вырезать кусок минимальной длины из готового транспаранта, но на котором будет все буквы, необходимые для его транспаранта.

Input format

В первой строке ввода записана строка текста — уже готовый транспарант. Во второй строке ввода записана строка текста — требуемый Остапу транспарант. Длины строк не превышают 50000 символов.

В строке допускаются любые символы с ASCII-кодами 33 - 126.

Output format

Выведите строку, являющуюся сегментом (то есть фрагментом, идущим подряд) первой строки ввода, которая содержит все требуемые буквы для второй строки ввода, и имеет минимальную длину. Если таких строк несколько, выведите любую. Если такой строки не существует, выведите пустую строку.

Examples

Input

```
cacacbcac
ab
```

Output

```
acb
```

Input

```
cacacbcac
bab
```

Output

Notes

ejudge 3.12.0+ (GIT 7bbe05192) (2024-02-05 23:59:28).

Результат работы на втором тесте — пустая строка.

6

Problem 6: 6-11: Оптимизация

Prob Id:	`6`
Full score:	`100`
Input file name:	`input.txt` or standard input
Time limit:	`5 s`
Memory RSS limit:	`256M`
Stack limit:	`256M`

Problem 6: Оптимизация

Трус, Балбес и Бывалый решили сделать олимпиаду по программированию. Балбес делал тесты для задачи A, и у него получилось 500 тестов; Трус делал тесты для задачи B, и у него получилось 800 тестов; Бывалый делал тесты для задачи C (самой сложной), и у него получилось 20 тестов.

Посылка по задаче считается успешной, если она прошла все тесты. Если хотя бы один из тестов не пройден, посылка считается неуспешной.

После олимпиады троица решила оценить эффективность набора тестов по каждой задаче. Так, неформально, тест к задаче считается "неэффективным", если его можно убрать из набора тестов, и результаты всех запусков по данной задаче не изменятся. Более строго, кластером тестов для некоторой задачи называется непустое подмножество тестов, такое что каждая неуспешная посылка либо проходит все тесты из кластера, либо не проходит ни одного теста из кластера. В этом случае не важно, по какой именно причине тест не пройден, они могут быть разными у разных посылок. Если все посылки по задаче были успешными, то кластером являются все тесты к задаче.

Количество тестов к задаче можно уменьшить без потери качества, если разбить все тесты к задаче на минимальное количество кластеров и оставить по одному тесту из каждого кластера. Каждый тест должен попадать в один и только один кластер. Могут существовать кластеры из одного теста.

Напишите программу, которая для протокола тестирования посылок для каждой задачи в олимпиаде определит, сколько в этой задаче было тестов и на сколько кластеров разбиваются тесты к этой задаче.

Input format

На вход программы подаются данные в формате CSV. Информация о каждой посылке находится на отдельной строке, а поля данных в посылке отделяются друг от друга символом "точка с запятой" (`;`). В полях данных присутствуют только символы с ASCII-кодами 32-126, кроме символа "точка с запятой". Символы "кавычка" и "апостроф" не несут никакого специального смысла.

Первое поле в каждой записи — это непустая строка, идентифицирующая задачу. Длина идентификатора задачи не превышает 64 символа. За ней идут поля с результатами запуска на тестах, где `OK` означает, что тест пройден, а любая другая строка, в том числе и пустая, что тест не пройден. У каждой задачи есть хотя бы один тест.

Гарантируется, что все успешные посылки по некоторой задаче содержат информацию об одном и том же числе тестов; а неуспешные посылки по данной задаче содержат не большее число тестов. В неуспешной посылке может быть меньшее число тестов, чем у успешных, но хотя бы один из тестов будет явно задан как неуспешный. Опущенный "хвост" тестов у неуспешной посылки рассматривается как неуспешный.

Разные задачи могут иметь разное число тестов.

Количество задач в олимпиаде не превышает 1000. Количество тестов к каждой задаче не превышает 1000. Количество посылок по каждой задаче не превышает 10000.

Output format

Для каждой задачи в олимпиаде на отдельной строке выведите два числа: количество тестов в задаче и количество выявленных кластеров тестов. Задачи должны выводиться в лексикографическом порядке их идентификаторов.

Эта задача — оптимизационная. Оценка за задачу зависит от качества оптимизации. Чем меньше кластеров, на которые удалось разбить тесты к задаче, тем выше балл за тест.

Examples

Input

```
A;OK;OK;OK
B;OK;OK;OK;OK
A;OK;RT;RT
B;OK;RT;OK
A;OK;WA;RT
B;OK;WA;WA;PE
```

Output

```
3 2
4 3
```
