Три-одиозные числа. 7-10 класс. Заключительный этап

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

Неотрицательное целое число назовём три-одиозным, если в записи этого числа в трибоначчиевой системе счисления количество единиц нечётное. Примеры три-одиозных чисел: $1_{10} = 1_{tri}$, $2_{10} = 10_{tri}$, $733_{10} = 10101101101_{tri}$.

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 три-одиозными 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 три-одиозных чисел выводится в виде неотрицательного целого числа, записанного в десятичной системе без незначащих нулей.

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

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

```
Ввод примера №3:
4
1
2
733
2
Вывод примера №3:
4
```

Решение

В решении можно запрограммировать следующие подзадачи: а) проверку того, является ли очередное число три-одиозным; б) подсчёт и вывод искомого количества. Перевод очередного числа A_i в трибоначчиевую систему реализуется жадным алгоритмом. Ради эффективности можно заранее подсчитать все числа трибоначчи, которые меньше 2^{24}. Среди этих чисел ищется наибольшее число трибоначчи, не превосходящее A_i. Если его нет, то A_i – ноль. Если оно есть, то обозначим его T_{k+3}, увеличим количество единичных разрядов в трибоначчиевой записи и вычислим разность $A_i - T_{k+3}$. Далее работаем с этой разностью и числом T_{k+2}. Если оно больше разности, то соответствующий разряд в трибоначчиевой записи будет нулевым. Если оно не меньше разности, то снова увеличим количество единичных разрядов в трибоначчиевой записи и вычислим разность $A_i - T_{k+3} - T_{k+2}$. И т. д. Дойдя до нулевой разности, проверяем на нечётность 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 TRIODIOUS710(input, output);
const   NUMTRIBO = 28;
        TRIBO: array [1 .. NUMTRIBO] of dword = (1, 2, 4, 7, 13, 24, 44, 81, 149,
```

```
        274, 504, 927, 1705, 3136, 5768, 10609, 19513, 35890, 66012, 121415, 223317,
        410744, 755476, 1389537, 2555757, 4700770, 8646064, 15902591);
var  N, I, ANSWER : word; A : dword;
function binSearch(A : dword) : byte;
var L, M, H : byte;
begin
    L := 1;
    H := NUMTRIBO;
    while L <= H do
    begin
        M := (L + H) div 2;
        if TRIBO[M] > A then
        begin
            H := M - 1;
        end
        else if TRIBO[M] < A then
        begin
            L := M + 1;
        end
        else
        begin
            break;
        end;
    end;
    if (M < NUMTRIBO) AND (TRIBO[M + 1] <= A) then binSearch := M + 1
    else if (M > 1) AND (TRIBO[M] > A) then binSearch := M - 1
    else binSearch := M
end;
function isTriOdious(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 - TRIBO[I];
        while (A > 0) AND (I > 1) do begin
            I := I - 1;
            if A >= TRIBO[I] then begin
                NUM1 := (NUM1 + 1) mod 2;
                A := A - TRIBO[I]
            end
        end;
    end;
    isTriOdious := NUM1
end;
begin
    readln(N);
    ANSWER := 0;
    for I := 1 to N do begin
        read(A);
        ANSWER := ANSWER + isTriOdious(A)
    end;
    write(ANSWER)
end.
```

Квадродерево. 7-10 классы. Заключительный этап

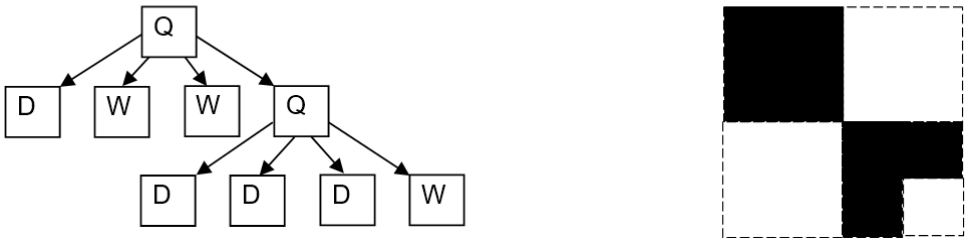
Рассмотрим способ представления плоских квадратных изображений, называемый квадродеревом:

1) Если изображение целиком одноцветное, то оно представляется квадродеревом из одной листовой вершины – пикселя. Линейная запись такого квадродерева 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 на изображении участки разного цвета, то оно делится на 4 равных квадрата (верхний левый, верхний правый, нижний левый, нижний правый) и представляется квадродеревом, состоящим из корневой вершины и четырёх поддеревьев, которые описывают части изображения – полученные при делении квадраты. Пусть $T_{\text{ВЛД}}$ – линейная запись верхнего левого поддерева, $T_{\text{ВПД}}$ – линейная запись верхнего правого поддерева, $T_{\text{НЛД}}$ – линейная запись нижнего левого поддерева, $T_{\text{НПД}}$ – линейная запись нижнего правого поддерева; тогда запись всего дерева будет такой: $QT_{\text{ВЛД}}T_{\text{ВПД}}T_{\text{НЛД}}T_{\text{НПД}}$

Пример: линейная запись квадродерева: $QDWWQDDDW$

квадродерево в виде графа: описываемое изображение:



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

Составьте программу, на вход которой в 1-ой строке подаётся линейная запись квадродерева T (длина записи не превышает 1365 символов). Программа находит цвет, в который закрашена часть квадродерева T, имеющая максимальную площадь. Если таких цветов больше чем один, то искомым является тот, которому соответствует символ, стоящий в алфавите ближе всего к концу. Программа в первой строке выводит символ, соответствующий найденному цвету. Далее, во второй строке, полагая, что общая площадь записанного квадродеревом T изображения равна 1, программа выводит площадь 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 квадродерева. Длина 1-ой строки не превосходит 1365 символов.

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 квадродерева T с максимальной площадью. Если таких цветов несколько, то из их символов выбирается тот, который следует за всеми другими в алфавитном порядке. Во второй строке вывода искомая площадь части квадродерева 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:
QDWWQDDDW	R	QWRYO
Вывод примера №1:	Вывод примера №2:	Вывод примера №3:
W	R	Y
0.1001	1.0	0.01

Решение

В решении можно запрограммировать следующие подзадачи: а) считывание записи квадродерева и одновременный расчёт площади частей квадродерева, покрашенных в каждый из 9 цветов;

б) поиск и вывод argmax, символ которого стоит после всех других по алфавиту. Запись квадродерева 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 4 раза помещаем в стек $H + 1$, иначе к площади, закрашенной цветом C, добавляем $\frac{1}{4}^{H-1}$. 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 жёлтый цвет. В цикле от белого до чёрного (с уменьшением значения счётчика цикла) сравниваем текущую максимальную площадь с площадью 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 площади нужно определить количество значимых разрядов, а затем вывести их.

Код возможного решения

```
program QUADTREE710(input, output);
const   STACKSIZE = 1370;
        SCALESIZE = 690;
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^(-2*I) к двоичной дроби *)
var CF : boolean; J : word;
begin   if I = 0 then B[0] := 1 else begin
           J := I * 2;
           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 площадей *)
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 4 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 := 9;
        for I := 8 downto 1 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.
```

Гномьи сундуки. 7-10 классы. Заключительный этап

В сокровищнице короля гномов Кхазад-Дума Дурина хранится множество сундуков. Сундуки занумерованы, начиная с единицы. В сундуках лежат гномьи деньги. О каждом сундуке известна денежная сумма, хранящаяся в нём, – неотрицательное целое число. Гномы пользуются своей собственной системой счисления для записи номеров сундуков и денежных сумм. Число 0 по-гномьи записывается двумя символами () – открывающей и закрывающей круглыми скобками. Цифра 1 по-гномьи записывается закрывающей угловой скобкой (или знаком «больше») >. Цифра 2 по-гномьи записывается двумя символами >! – закрывающей угловой скобкой > и восклицательным знаком. Цифра 3 по-гномьи записывается тремя символами >!! – закрывающей угловой скобкой > и двумя восклицательными знаками. Цифра 4 по-гномьи записывается четырьмя символами >!!! – закрывающей угловой скобкой > и тремя восклицательными знаками. Цифра 5 по-гномьи записывается двумя символами >? – закрывающей угловой скобкой > и вопросительным знаком. Цифра 6 по-гномьи записывается открывающей угловой скобкой (или знаком «меньше») <. Цифра 7 по-гномьи записывается двумя символами <! – открывающей угловой скобкой < и восклицательным знаком. Цифра 8 по-гномьи записывается тремя символами <!! – открывающей угловой скобкой < и двумя восклицательными знаками. Цифра 9 по-гномьи записывается четырьмя символами <!!! – открывающей угловой скобкой < и тремя восклицательными знаками. Цифра десять (A в позиционных системах счисления с основанием больше десяти) по-гномьи записывается двумя символами <? – открывающей угловой скобкой < и вопросительным знаком. Другие цифры (в том числе цифру 0) гномы не используют.

Ещё одной особенностью гномьей системы счисления является то, что их цифры в записи числа следуют в обратном порядке, если сравнивать с обычной позиционной системой счисления. Пусть имеется гномья запись числа $d_0d_1...d_{n-1}d_n$, чтобы узнать, какое ненулевое число записано, нужно вычислить: $val(d_0)*10^0+val(d_1)*10^1+...+val(d_{n-1})*10^{n-1}+val(d_n)*10^n$. Здесь val обозначает взятие значения цифры и для удобства определено в таблице:

цифра d_i	>	>!	>!!	>!!!	>?	<	<!	<!!	<!!!	<?
значение $val(d_i)$	1	2	3	4	5	6	7	8	9	10

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

0_{10} =()

10_{10} =<?

$2024_{10} = 4*10^0+2*10^1+10*10^2+1*10^3$ =>!!!>!<?>

Король Дурин 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 Дурин выбирает из них такую пару, что сумма номеров сундуков минимальна.

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 гномьей системе денежных сумм, хранящихся в сундуках с номерами от 1 до N. Известно, что в каждой записи денежной суммы не более чем 100 символов (угловых или круглых скобок, восклицательных или вопросительных знаков). 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 $K + L$ наименьшая. Программа выводит в первой строке гномью запись найденного числа K, а во второй строке – гномью запись найденного числа L. Каждое число записывается в гномьей 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 гномьей системе денежных сумм, хранящихся в сундуках, – последовательности, в которых могут встретиться только угловые или круглые скобки <, >, (,) и восклицательные или вопросительные знаки. Эти последовательности записаны по правилам составления записей чисел в гномьей системе. Длины строк находятся в диапазоне от 1 до 100 включительно.

Формат вывода: В первой строке выводится гномья запись найденного числа K. Во второй строке выводится гномья 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 $K + L$ наименьшая.

`Ввод примера №1:` `Ввод примера №2:` `Ввод примера №3:`

```
2
()
>!!!>!<?>
Вывод примера №1:
>
>!
```

```
3
<?
<?
>!!!>!<?>
Вывод примера №2:
>
>!!
```

```
4
>!!!>!<?>
<?
<?
()
Вывод примера №3:
>
>!!!
```

Решение

В решении можно запрограммировать следующие подзадачи: 1) считывание очередной денежной суммы в гномьей записи и представление её в виде строки специального вида, в которой гномьи цифры заменены на латинские строчные буквы a, ..., j, 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 гномьей 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 описей, которые программа успела считать) и текущий «антирекорд» (минимум среди всех описей, которые программа успела считать). Очередная опись после считывания сравнивается с рекордом. Если рекорд меньше её, то она становится рекордом. Также очередная опись сравнивается с «антирекордом». Если очередная опись больше «антирекорда», то она становится «антирекордом». Но нужно учитывать случай, когда в первом и во втором сундуках одинаковые суммы денег. Чтобы в этом случае не получить одинаковые номера у рекорда и «антирекорда», нужно при обработке 2-ой описи обновлять номер «антирекорда», если рекорд и «антирекорд» равны. Но далее может выясниться, что во всех других сундуках суммы денег не меньше, и хотя бы в одном из них сумма больше. По условию, в этом случае номер «антирекорда» должен быть 1, а не 2. То есть, сдвиг номера «антирекорда» с 1 на 2 был напрасным. Поэтому по окончании обработки делается проверка на то, что номер рекорда больше 1, и что сумма денег в первом сундуке совпадает с «антирекордом». Если результат проверки истинен, то номеру «антирекорда» присваивается 1. Завершает работу вывод гномьих записей номеров рекорда и «антирекорда» по возрастанию. Перевод в гномью запись схож с переводом в десятичную систему, но отличается другим порядком цифр записи числа и тем, что не используется нулевая цифра, но используется цифра десять.

Код возможного решения

```
program DWARVENCHESTS710 (input, output);
const CHESTLENGHT = 100;
type    chests = array [1..CHESTLENGHT] of char;
        answer = record number : word; chest : chests end;
var  CURCHEST, FIRST : chests; N, I : word; CHECK : integer; CURMAX, CURMIN : answer;
procedure readChest(var CHEST : chests);
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;
            '>': begin CHEST[I] := 'a'; I := I - 1 end;
            '<': begin CHEST[I] := 'f'; I := I - 1 end;
            '!': begin CHEST[I+1] := succ(CHEST[I + 1]) end;
            '?': begin CHEST[I+1] := chr(ord(CHEST[I + 1]) + 4) end;
        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 writeDwarven(I : integer);
begin
    if (I = 0) then write('()')
    else while (I > 0) do
        case I mod 10 of
            1: begin write('>'); I := I div 10 end;
            2: begin write('>!'); I := I div 10 end;
            3: begin write('>!!'); I := I div 10 end;
            4: begin write('>!!!'); I := I div 10 end;
            5: begin write('>?'); I := I div 10 end;
            6: begin write('<'); I := I div 10 end;
            7: begin write('<!'); I := I div 10 end;
            8: begin write('<!!'); I := I div 10 end;
            9: begin write('<!!!'); I := I div 10 end;
            0: begin write('<?'); I := (I - 10) div 10 end;
        end;
end;
begin   readln(N);
        with CURMAX do begin
            number := 1;
            readChest(chest);
        end;
        CURMIN := CURMAX;
        FIRST := CURMAX.chest;
        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) or (I = 2) and (CHECK = 0) then
                with CURMIN do begin
                   number := I;
                   chest := CURCHEST
            end; (* if *)
        end; (* for *)
        if (CURMAX.number > 1) and (compareChest(FIRST, CURMIN.chest) = 0) then
            CURMIN.number := 1;
        if (CURMAX.number > CURMIN.number) then begin
            writeDwarven(CURMIN.number);
            writeln;
            writeDwarven(CURMAX.number) end
        else begin
            writeDwarven(CURMAX.number);
            writeln;
```

```
            writeDwarven(CURMIN.number)
        end (* if *)
end.
```

4

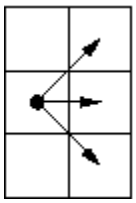
Problem 4: 4-710: Грибы отсюда

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: 5-710: 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.