

Задача 1. Незнайка и Коллатц**Критерий оценивания решений задачи 1**

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

Технические баллы за задачу 1

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

Перевод технических баллов в оценку

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

Условия задачи 1

Незнайка очень любит преобразовывать числа, но еще больше он любит преобразовывать наборы, составленные из чисел. Однажды он придумал случайное на его взгляд преобразование при помощи функции

$$f(n) = \begin{cases} \frac{n}{2}, & n - \text{четное число} \\ 3n + 1, & n - \text{нечетное число} \end{cases}$$

Он решил применять $f(n)$ к каждому числу в наборе (причем одновременно) до тех пор, пока все числа не станут единицами. Но вот беда, для некоторых массивов (например $[1, 2, 3]$) процесс преобразования может заиклиться: $[1, 2, 3] \rightarrow [4, 1, 10] \rightarrow [2, 4, 5] \rightarrow [1, 2, 16] \rightarrow [4, 1, 8] \rightarrow [2, 4, 4] \rightarrow [1, 2, 2] \rightarrow [4, 1, 1] \rightarrow [2, 4, 4] \rightarrow \dots$ и никогда не завершиться. Мимо проходил Знайка, который рассказал, что функция $f(x)$ носит имя Коллатца и по недоказанной гипотезе её применение снова и снова, стартуя с любого числа, всегда приводит к числу 1, после чего возникает цикл $1 \rightarrow 4 \rightarrow 2 \rightarrow 1$.

Полагая, что гипотеза Коллатца верна, помогите Незнайке написать программу, которая определяет, настанет ли момент, когда все элементы массива, к которым одновременно применяют функцию $f(x)$ снова и снова, станут единицами.

Формат ввода: В первой строке вводится натуральное число n : $2 \leq n \leq 10000$. Во второй строке содержится последовательность из n натуральных чисел a_i : $1 \leq a_i \leq 1000$.

Формат вывода: Выводится либо наименьшее из возможных количество применений функции $f(x)$, после которого все элементы массива станут единицами, либо -1, если никакое количество применений $f(x)$ не сделает массив, состоящим только из единиц. Если исходный массив содержит только единицы, то выводится 0.

Ввод примера №1:

10

1 1 1 1 1 1 1 1 1 1

Вывод примера №1:

0

Ввод примера №2:

2

8 1

Вывод примера №2:

3

Ввод примера №3:

3

1 2 3

Вывод примера №3:

-1

Решение

Можно заметить, что существует ровно один цикл, стартовый с 1, и имеющий длину 3 — $1 \rightarrow 4 \rightarrow 2 \rightarrow 1 \rightarrow \dots$. Для всех целых чисел от 1 до 1000 можно заранее рассчитать минимальное количество применений функции $f(x)$, приводящее в 1. Весь массив может превратиться в единицы, если для каждого a_i рассчитанное количество применений даёт одинаковый остаток от

деления на 3. Искомым результатом в таком случае будет максимальное количество применений среди всех a_i . Иначе результатом будет -1 .

Код возможного решения

```

program COLLATZ11(input, output);
const   ARR_SIZE = 1000;
type    massiv = array [1..ARR_SIZE] of word;
var     N, I, A, M, RESULT : word;
        FLAG : boolean;
        COLLATZ : massiv = (0,1,7,2,5,8,16,3,19,6,14,9,9,17,17,4,12,20,20,7,7,15,15,
10,23,10,111,18,18,18,106,5,26,13,13,21,21,21,34,8,109,8,29,16,16,16,104,11,24,24,
24,11,11,112,112,19,32,19,32,19,19,107,107,6,27,27,27,14,14,14,102,22,115,22,14,22,
22,35,35,9,22,110,110,9,9,30,30,17,30,17,92,17,17,105,105,12,118,25,25,25,25,25,87,
12,38,12,100,113,113,113,69,20,12,33,33,20,20,33,33,20,95,20,46,108,108,108,46,7,
121,28,28,28,28,28,41,15,90,15,41,15,15,103,103,23,116,116,116,23,23,15,15,23,36,
23,85,36,36,36,54,10,98,23,23,111,111,111,67,10,49,10,124,31,31,31,80,18,31,31,31,
18,18,93,93,18,44,18,44,106,106,106,44,13,119,119,119,26,26,26,119,26,18,26,39,26,
26,88,88,13,39,39,39,13,13,101,101,114,26,114,52,114,114,70,70,21,52,13,13,34,34,
34,127,21,83,21,127,34,34,34,52,21,21,96,96,21,21,47,47,109,47,109,65,109,109,47,
47,8,122,122,122,29,29,29,78,29,122,29,21,29,29,42,42,16,29,91,91,16,16,42,42,16,
42,16,60,104,104,104,42,24,29,117,117,117,117,117,55,24,73,24,117,16,16,16,42,24,
37,37,37,24,24,86,86,37,130,37,37,37,37,55,55,11,24,99,99,24,24,24,143,112,50,112,
24,112,112,68,68,11,112,50,50,11,11,125,125,32,125,32,125,32,32,81,81,19,125,32,
32,32,32,32,50,19,45,19,45,94,94,94,45,19,19,45,45,19,19,45,45,107,63,107,58,107,
107,45,45,14,32,120,120,120,120,120,120,27,58,27,76,27,27,120,120,27,19,19,19,27,
27,40,40,27,40,27,133,89,89,89,133,14,133,40,40,40,40,40,32,14,58,14,53,102,102,
102,40,115,27,27,27,115,115,53,53,115,27,115,53,71,71,71,97,22,115,53,53,14,14,14,
40,35,128,35,128,35,35,128,128,22,35,84,84,22,22,128,128,35,35,35,27,35,35,53,53,
22,48,22,22,97,97,97,141,22,48,22,141,48,48,48,97,110,22,48,48,110,110,66,66,110,
61,110,35,48,48,48,61,9,35,123,123,123,123,123,61,30,123,30,123,30,30,79,79,30,30,
123,123,30,30,22,22,30,22,30,48,43,43,43,136,17,43,30,30,92,92,92,43,17,136,17,30,
43,43,43,87,17,43,43,43,17,17,61,61,105,56,105,30,105,105,43,43,25,30,30,30,118,
118,118,30,118,56,118,118,118,56,56,25,74,74,74,25,25,118,118,17,56,17,69,17,
17,43,43,25,131,38,38,38,38,38,69,25,131,25,131,87,87,87,131,38,25,131,131,38,38,
38,38,38,30,38,30,56,56,56,131,12,51,25,25,100,100,100,38,25,144,25,100,25,25,144,
144,113,51,51,51,113,113,25,25,113,51,113,144,69,69,69,95,12,64,113,113,51,51,51,
64,12,64,12,38,126,126,126,38,33,126,126,126,33,33,126,126,33,126,33,64,82,82,82,
170,20,33,126,126,33,33,33,64,33,25,33,25,33,33,51,51,20,46,46,46,20,20,46,46,95,
33,95,139,95,95,46,46,20,139,20,20,46,46,46,95,20,90,20,46,46,46,139,108,20,64,
64,108,108,59,59,108,33,108,152,46,46,46,59,15,33,33,33,121,121,121,152,121,33,121,
59,121,121,121,121,28,121,59,59,28,28,77,77,28,77,28,103,121,121,121,72,28,59,20,
20,20,20,72,28,46,28,134,41,41,41,134,28,41,41,41,28,28,134,134,90,134,90,41,90,
90,134,134,15,28,134,134,41,41,41,85,41,41,41,41,41,41,33,33,15,59,59,59,15,15,54,
54,103,28,103,147,103,103,41,41,116,147,28,28,28,28,28,178,116,147,116,28,54,54,54,
147,116,116,28,28,116,116,54,54,72,147,72,46,72,72,98,98,23,67,116,116,54,54,54,
116,15,67,15,54,15,15,41,41,36,129,129,129,36,36,129,129,36,129,36,67,129,129,129,
116,23,129,36,36,85,85,85,129,23,173,23,85,129,129,129,36,36,36,36,36,36,28,28,
36,28,36,28,54,54,54,129,23,49,49,49,23,23,23,142,98,49,98,36,98,98,142,142,23,98,
49,49,23,23,142,142,49,23,49,36,49,49,98,98,111,93,23,23,49,49,49,49,111);
begin FLAG := false;
      readln(N);
      read(A);
      RESULT := COLLATZ[A];
      M := RESULT mod 3;
      I := 1;
      while (I < N) do begin

```

```
    read(A);
    if (COLLATZ[A] mod 3 <> M) then begin FLAG := true; I := N end
    else if (COLLATZ[A] > RESULT) then RESULT := COLLATZ[A];
    I := I + 1
end;
if FLAG then write(-1)
else write(RESULT)
end.
```

Задача 2. Незнайка и авиакоррест

Критерий оценивания решений задачи 2

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

Технические баллы за задачу 2

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

Перевод технических баллов в оценку

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

Условия задачи 2

Пусть дан массив $a = [a_1, a_2, \dots, a_n]$, имеющий длину n , и содержащий различные неотрицательные целые числа. К массиву можно применять операцию *shift*, записываемую как $shift(a, i_1, i_2, \dots, i_k)$, при чём $i_j \neq i_m$ при любых $j \neq m$. Данная запись говорит, что в массиве a i_1 -й элемент будет переставлен на i_2 -ю позицию, i_2 -й элемент – на i_3 -ю позицию и т. д., а i_k -й элемент – на i_1 -ю позицию. Например, если к массиву $a = [9, 3, 1, 7, 5]$ применить операцию $shift(a, 1, 5, 3)$, то получится отсортированный по возрастанию массив $[1, 3, 5, 7, 9]$. Длительность выполнения одной операции *shift* измеряется в секундах и равна удвоенному количеству индексов i_j в ней, к которому добавлена одна дополнительная секунда для подготовки выполнения операции. Таким образом для $shift(a, 5, 1, 3)$ потребуется $7 = 2 * 3 + 1$ секунд.

Требуется написать программу, которая найдет минимальное время, за которое массив будет отсортирован по возрастанию с помощью применений операции *shift*. Иные операции применять к массиву нет возможности.

Формат ввода: В первой строке вводится натуральное число n : $2 \leq n \leq 1000$. Во второй строке содержится последовательность из n различных неотрицательных целых чисел a_i : $0 \leq a_i \leq 10^9$.

Формат вывода: Выводится минимальное количество секунд – время для сортировки по возрастанию массива с помощью применений операции *shift*. Если исходный массив уже отсортирован по возрастанию, то выводится 0. Иначе, если массив невозможно отсортировать, применяя операцию *shift*, то выводится -1.

Ввод примера №1:

10
1 2 3 4 5 6 7 8 9 10

Вывод примера №1:

0

Ввод примера №2:

5
9 3 1 7 5

Вывод примера №2:

7

Ввод примера №3:

6
5 1 6 4 2 3

Вывод примера №3:

12

Решение

Нам дан массив $a[n]$. Пусть $s[n]$ – это его отсортированная версия. Так как все элементы по условию различны, для s можно использовать структуру упорядоченного множества `std::set` или отсортировать элементы массива $a[n]$ одним из алгоритмов сортировки. Заведём также массив `bool used[n]`, в котором будем хранить информацию об обработанных элементах.

Основная идея алгоритма заключается в том, что для каждого a_i можно по массиву s узнать его отсортированную позицию $next$. Это можно сделать бинарным поиском за $O(\log(n))$ или линейным поиском за $O(n)$. Для элемента a_{next} нужно выполнять эти же действия (т. е. найти его позицию в отсортированном массиве), пока мы не встретим элемент, с которого все началось (т. е. тот, который в отсортированном массиве стоит на i -ой позиции). При этом нужно не забывать пометить все элементы, которые встречаются, в массиве `used[n]`. Мы получим цикл (циклическую перестановку), для которого применение операции *shift* отсортирует пройденные элементы. Если

length - длина цикла больше чем 1, то требуется к ответу добавить $length * 2 + 1$ и перейти на следующий элемент массива, который не был обработан.

Итоговая сложность алгоритма зависит от способов сортировки и поиска элемента. Если поиск линейный или сортировка квадратичная, то сложность будет $O(n^2)$. Если использовать быструю сортировку и бинарный поиск, то сложность будет $O(n \log(n))$.

Код возможного решения

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

using namespace std;
int main() {
    int n;
    cin >> n;
    vector<int> a(n);
    for (int i = 0; i < n; ++i) cin >> a[i];
    vector<bool> used(n, false);
    set<int> s(a.begin(), a.end());
    int sort_cost = 0;
    for (int i = 0; i < n; ++i) {
        if (used[i]) continue;
        int cycle_start = i;
        used[cycle_start] = true;
        int runner = distance(s.begin(), s.lower_bound(a[cycle_start]));
        int cycle_size = 1;
        while (runner != cycle_start) {
            used[runner] = true;
            runner = distance(s.begin(), s.lower_bound(a[runner]));
            ++cycle_size;
        }
        if (cycle_size > 1) sort_cost += 2 * cycle_size + 1;
    }
    cout << sort_cost << endl;
}
```

Задача 3. Незнайка и компас

Критерий оценивания решений задачи 3

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

Технические баллы за задачу 3

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

Перевод технических баллов в оценку

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

Условия задачи 3

Волшебник подарил Незнайке компас. Поначалу он не знал, что с ним делать, но потом стал брать компас на прогулки и записывать его показания (азимуты) на разные примечательные объекты. Способы записи азимутов Незнайке не были известны, поэтому он изобрёл свой собственный способ.

Записывая азимут, Незнайка сначала указывает заглавной латинской буквой направление, от которого потом отсчитывает угол: N – север, E – восток, S – юг, W – запад. Потом Незнайка записывает величину угла – целое неотрицательное количество градусов в диапазоне от 0 включительно до 360 включительно. В конце записи Незнайка указывает еще одну заглавную латинскую букву (N, E, S или W) чтобы отметить то, был ли отложен угол по ходу часовой стрелки или против хода часовой стрелки. Так, если запись азимута начинается на букву N, то буква E в конце записи соответствует ходу часовой стрелки, W – обратному ходу, а буквы S и N не ставятся в конце. Если запись азимута начинается на букву S, то буква W в конце записи соответствует ходу часовой стрелки, E – обратному ходу, а буквы N и S не ставятся в конце. Если запись азимута начинается на букву E, то буква S в конце записи соответствует ходу часовой стрелки, N – обратному ходу, а буквы W и E не ставятся в конце. Если запись азимута начинается на букву W, то буква N в конце записи соответствует ходу часовой стрелки, S – обратному ходу, а буквы E и W не ставятся в конце.

Ознакомившись с записями Незнайки, Знайка схватился за голову и пришёл в ужас. Он сразу понял, что такой способ позволяет одно и то же показание компаса записать по-разному. Например, направление на северо-восток (45 градусов от направления на север, отложенные по ходу часовой стрелки) соответствует восьми записям: N45E, N315W, E45N, E315S, S135E, S225W, W225S, W135N. Например, направление на север (0 градусов от направления на север, отложенные по ходу часовой стрелки) соответствует десяти записям: N0E, N0W, N360E, N360W, E90N, E270S, S180E, S180W, W90N, W270S.

Требуется написать программу, которая считает последовательность записей азимутов и найдет в ней показание, встречающееся чаще всего, учитывая то, что записи одинаковых показаний могут быть разными. Если искомым показаний больше чем одно, то среди них следует выбрать то, которое имеет наименьший угол от направления на север, отложенный по ходу часовой стрелки. Для однозначности будем считать, что само направление на север имеет угол в 0 градусов (а не 360), отложенный от самого себя. В ответе программа должна выдать номер элемента последовательности с записью выбранного показания, находящегося ближе всего к началу последовательности. Нумерация элементов последовательности начинается с единицы.

Формат ввода: В первой строке вводится натуральное число K : $1 \leq K \leq 2000$. Во второй строке содержится последовательность из K записей A_i . Каждая запись A_i начинается и заканчивается заглавной латинской буквой (N, E, S, W). Между буквами находится неотрицательное целое число от 0 включительно до 360 включительно, записанное в десятичной системе. Каждая запись

A_i сделана по правилам, изложенным во втором абзаце условия. Между соседними записями в последовательности стоит пробел.

Формат вывода: Выводится номер r элемента последовательности записей A_r , такого что, записанное в A_r показание встречается в последовательности чаще всего и среди всех чаще всего встречающихся показаний оно имеет наименьший угол от направления на север, отложенный по ходу часовой стрелки, и A_r является записью такого показания, стоящей ближе всего к началу последовательности.

Ввод примера №1:

10

N1E N0W N360E N360W E90N E270S S180E S180W W90N W270S

Вывод примера №1:

2

Ввод примера №2:

8

N45E N315W E45N E315S S180E S180W W90N W270S

Вывод примера №2:

5

Ввод примера №3:

1

N359E

Вывод примера №3:

1

Решение

В решении стоит использовать массив из 360 элементов с индексами от 0 до 359. В элемент с индексом i будет записываться количество встреченных в считанной части последовательности вхождений записи показания, соответствующего углу i градусов от направления на север, отложенному по ходу часовой стрелки, а также номер первого вхождения в последовательность. Программа будет в цикле считывать очередной элемент последовательности, определять угол i , делать приращение на единицу элемента массива с индексом i , при надобности, вносить номер вхождения. В том же цикле программа будет подсчитывать максимальное количество вхождений повторяющихся показаний. После окончания ввода, искомый номер ищется проходом по массиву от его начала до первого встреченного элемента с максимальным количеством вхождений и взятием хранящегося в нём номера первого вхождения.

Код возможного решения

```
program COMPAS11(input, output);
type    azimuth = record count, position : word end;
      massiv = array [0..359] of azimuth;
var  N, I, A, RESULT : word;
      COMPAS : massiv;
function readAzimuth() : word;
var CH, CH1 : char; FLAG : boolean; A, R : word;
begin
  read(CH1);
  case CH1 of
    'N' : R := 0;
    'E' : R := 90;
    'S' : R := 180;
    'W' : R := 270;
  end;
  FLAG := false;
  A := 0;
  repeat
    read(CH);
    case CH of
```

```

        'N' : begin if (CH1 = 'E') then A := 360 - A; FLAG := true end;
        'E' : begin if (CH1 = 'S') then A := 360 - A; FLAG := true end;
        'S' : begin if (CH1 = 'W') then A := 360 - A; FLAG := true end;
        'W' : begin if (CH1 = 'N') then A := 360 - A; FLAG := true end;
        '0'..'9' : A := A * 10 + ord(CH) - ord('0');
    end;
until FLAG;
if not (eoln or eof) then read(CH);
readAzimuth := (R + A) mod 360
end;

begin for I := 0 to 359 do
    with COMPAS[I] do begin
        count := 0;
        position := 0
    end;
    readln(N);
    RESULT := 0;
    for I := 1 to N do begin
        A := readAzimuth;
        with COMPAS[A] do begin
            count := count + 1;
            if (count > RESULT) then RESULT := count;
            if (position = 0) then position := I
        end
    end;
end;
I := 0;
while (I < 359) AND (COMPAS[I].count < RESULT) do I := I + 1;
RESULT := COMPAS[I].position;
write(RESULT)
end.

```


Задача 4. Незнайка и буквенная система счисления

Критерий оценивания решений задачи 4

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

Технические баллы за задачу 4

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

Перевод технических баллов в оценку

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

Условия задачи 4

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: $2024_{10} = 38 * 52^1 + 48 * 52^0 = tw$. Все стали пользоваться буквенной системой счисления, как положено. Один лишь Незнайка, как всегда, творчески подошёл к делу. Однажды Знайка заглянул в исписанный Незнайкой листок, схватился за голову и пришёл в ужас. В записи чисел помимо обычных латинских букв Незнайка задействовал как цифры буквы с крышками ($\hat{A} \dots \hat{Z}$ и $\hat{a} \dots \hat{z}$), буквы с волной ($\tilde{A} \dots \tilde{Z}$ и $\tilde{a} \dots \tilde{z}$) и буквы с нижними подчёркиваниями ($\underline{A} \dots \underline{Z}$ и $\underline{a} \dots \underline{z}$). Знайка обратился к Незнайке за разъяснениями. Выяснилось, что Незнайка модифицировал буквенную систему счисления, разрешив переполнять в ней разряды, т. е. записывать в них цифры со значениями, превышающими 51. При этом основание системы счисления он оставил прежним – 52. Незнайка дополнил таблицу цифр и их значений:

цифра	\hat{A}	\hat{B}	\hat{C}	\hat{D}	\hat{E}	\hat{F}	\hat{G}	\hat{H}	\hat{I}	\hat{J}	\hat{K}	\hat{L}	\hat{M}	\hat{N}	\hat{O}	\hat{P}	\hat{Q}	\hat{R}	\hat{S}	\hat{T}	\hat{U}	\hat{V}	\hat{W}	\hat{X}	\hat{Y}	\hat{Z}
значение	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77
цифра	\hat{a}	\hat{b}	\hat{c}	\hat{d}	\hat{e}	\hat{f}	\hat{g}	\hat{h}	\hat{i}	\hat{j}	\hat{k}	\hat{l}	\hat{m}	\hat{n}	\hat{o}	\hat{p}	\hat{q}	\hat{r}	\hat{s}	\hat{t}	\hat{u}	\hat{v}	\hat{w}	\hat{x}	\hat{y}	\hat{z}
значение	78	79	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99	100	101	102	103
цифра	\tilde{A}	\tilde{B}	\tilde{C}	\tilde{D}	\tilde{E}	\tilde{F}	\tilde{G}	\tilde{H}	\tilde{I}	\tilde{J}	\tilde{K}	\tilde{L}	\tilde{M}	\tilde{N}	\tilde{O}	\tilde{P}	\tilde{Q}	\tilde{R}	\tilde{S}	\tilde{T}	\tilde{U}	\tilde{V}	\tilde{W}	\tilde{X}	\tilde{Y}	\tilde{Z}
значение	104	105	106	107	108	109	110	111	112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127	128	129
цифра	\tilde{a}	\tilde{b}	\tilde{c}	\tilde{d}	\tilde{e}	\tilde{f}	\tilde{g}	\tilde{h}	\tilde{i}	\tilde{j}	\tilde{k}	\tilde{l}	\tilde{m}	\tilde{n}	\tilde{o}	\tilde{p}	\tilde{q}	\tilde{r}	\tilde{s}	\tilde{t}	\tilde{u}	\tilde{v}	\tilde{w}	\tilde{x}	\tilde{y}	\tilde{z}
значение	130	131	132	133	134	135	136	137	138	139	140	141	142	143	144	145	146	147	148	149	150	151	152	153	154	155
цифра	\underline{A}	\underline{B}	\underline{C}	\underline{D}	\underline{E}	\underline{F}	\underline{G}	\underline{H}	\underline{I}	\underline{J}	\underline{K}	\underline{L}	\underline{M}	\underline{N}	\underline{O}	\underline{P}	\underline{Q}	\underline{R}	\underline{S}	\underline{T}	\underline{U}	\underline{V}	\underline{W}	\underline{X}	\underline{Y}	\underline{Z}
значение	156	157	158	159	160	161	162	163	164	165	166	167	168	169	170	171	172	173	174	175	176	177	178	179	180	181
цифра	\underline{a}	\underline{b}	\underline{c}	\underline{d}	\underline{e}	\underline{f}	\underline{g}	\underline{h}	\underline{i}	\underline{j}	\underline{k}	\underline{l}	\underline{m}	\underline{n}	\underline{o}	\underline{p}	\underline{q}	\underline{r}	\underline{s}	\underline{t}	\underline{u}	\underline{v}	\underline{w}	\underline{x}	\underline{y}	\underline{z}
значение	182	183	184	185	186	187	188	189	190	191	192	193	194	195	196	197	198	199	200	201	202	203	204	205	206	207

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

$$2024_{10} = 38 * 52^1 + 48 * 52^0 = mw = 37 * 52^1 + 100 * 52^0 = l\hat{w} = 36 * 52^1 + 152 * 52^0 = k\tilde{w} = 35 * 52^1 + 204 * 52^0 = j\underline{w}$$

Требуется написать программу, которая считает запись числа, а затем последовательность записей чисел, при этом все вводимые числа записаны в модифицированной Незнайкой системе счисления. Во вводе буква с крышкой задаётся двумя символами, например, A^\wedge означает \hat{A} . Буква с волной также задаётся двумя символами, например, B^\sim означает \tilde{B} . Буква с подчёркиванием также задаётся двумя символами, например, $C_\underline{}$ означает \underline{C} . В ответе программа должна выдать номер последнего вхождения первого числа в последовательность, с учётом того, что одно и то же число может быть записано разными способами. Нумерация элементов последовательности начинается с единицы. Если вхождений нет, то программа выдаёт результат 0.

Формат ввода: В первой строке вводится запись числа N в модифицированной Незнайкой системе счисления. В записи используются заглавные и/или строчные латинские буквы, крышка \wedge , волна \sim , подчёркивание $\underline{}$. Длина первой строки не более чем 50. Во второй строке вводится натуральное число K в десятичной записи: $1 \leq K \leq 500$. В третьей строке содержится последовательность из K записей чисел A_i в системе счисления, модифицированной Незнайкой. Длина каждой записи A_i не более чем 50. Между соседними записями в последовательности стоит пробел.

Формат вывода: Выводится номер r элемента последовательности записей A_r , такого, что записанное в A_r число равно числу N , записанному в первой строке, и A_r является записью числа N , стоящей ближе всего к концу последовательности. При отсутствии вхождений выводится 0.

Ввод примера №1:

```
jw_
10
В mw D lw^ F jw_ H kw~ J K
```

Вывод примера №1:

```
8
Ввод примера №2:
```

```
jv_
5
ju_ lw^ mu jw_ ku~
```

Вывод примера №2:

```
0
Ввод примера №3:
abcdrfghijklmnopqrstuvwxyzABCDEFGHJKLMNOPQRSTUVWXYZ
1
abcdrfghijklmnopqrstuvwxyzABCDEFGHJKLMNOPQRSTUVWXYZ
```

Вывод примера №3:

```
1
```

Решение

В решении стоит использовать массив из 50 элементов со значениями от 0 до 210 как внутреннее представление чисел, записанных 50 разрядами рассматриваемой системы. 210 выбрано верхней границей диапазона, так как потребуется запас для осуществления нормализации числа. Для удобства сравнения старшие разряды стоит хранить в начальных элементах массива. В старших разрядах могут быть незначащие нули. Программа будет в цикле считывать очередной элемент последовательности, осуществлять его нормализацию (т. е. переводить во внутреннее представление в виде массива со значениями разрядов от 0 до 51), сравнивать нормализованный массив первого числа и нормализованный массив текущего элемента, при равенстве обновлять номер последнего вхождения. После окончания ввода выводится искомый номер, если вхождения были. Иначе выводится 0.

При нормализации считанного числа следует начинать с младшего разряда, оставлять в нём остаток от деления нацело значения записанной в разряде цифры на 52, а частное прибавлять к следующему по старшинству разряду. Такие действия проделываются со всеми прочитанными

разрядами. Если их меньше 50, то оставшиеся старшие разряды заполняются нулями.

Код возможного решения

```
program LETTERS11(input, output);
type    number = array [1..50] of byte;
var  K, I, RESULT : word;
      A1, AI : number;
procedure readNumber(var A : number);
var CH : char; I, J : word;
begin for I := 1 to 50 do A[I] := 0;
      read(CH);
      I := 0;
      J := 0;
      while (I < 50) and (CH <> ' ') do begin
        case CH of
          'A'..'Z' : begin J := J + 1; A[J] := ord(CH) - ord('A') end;
          'a'..'z' : begin J := J + 1; A[J] := ord(CH) - ord('a') + 26 end;
          '^' : A[J] := A[J] + 52;
          '~' : A[J] := A[J] + 104;
          '_' : A[J] := A[J] + 156;
        end;
        if not (eoln or eof) then begin
          read(CH);
          I := I + 1
        end else I := 50
      end;
      for I := J downto 1 do A[50 - J + I] := A[I];
      for I := 1 to 50 - J do A[I] := 0;
      for I := 50 downto 2 do begin
        A[I - 1] := A[I - 1] + A[I] div 52;
        A[I] := A[I] mod 52
      end;
end;

begin readNumber(A1);
      readln(K);
      RESULT := 0;
      for I := 1 to K do begin
        readNumber(AI);
        if (CompareByte(A1, AI, 50) = 0) then RESULT := I
      end;
      write(RESULT)
end.
```

Задача 5. Незнайка и урожай

Критерий оценивания решений задачи 5

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

Технические баллы за задачу 5

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

Перевод технических баллов в оценку

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

Условия задачи 5

Незнайка собирает урожай на прямоугольном поле размером $M \times N$ клеток. Каждая клетка содержит $a_{i,j}$ единиц продукции. Незнайка выбирает на поле два прямоугольника из клеток и собирает всю продукцию во всех клетках, принадлежащих прямоугольникам. Если прямоугольники пересекаются, то в общих клетках прямоугольников продукция собирается два раза. Два прямоугольника не должны совпадать полностью. Продукция прямоугольника равна сумме произведений всех составляющих его клеток. Продукция складывается в машину, минимальная грузоподъемность которой равна A , а максимальная грузоподъемность — B .

Найдите количество способов, которым можно выбрать два прямоугольника таким образом, что сумма продукции в двух прямоугольниках была не меньше A и не больше B .

Формат ввода: В первой строке входных данных задаются четыре целых числа M, N, A, B : $0 < M, 0 < N, M * N \leq 1600, |A| \leq 2000000000, |B| \leq 2000000000, A \leq B$. Далее идут M строк по N чисел в каждой, задающие матрицу a , при этом каждый элемент матрицы $a_{i,j}$: $a_{i,j} \leq 2000000000$.

Формат вывода: Выведите одно число: количество подходящих пар прямоугольников.

Ввод примера №1:

```
4 3 2 2
1 1 1
1 1 1
1 1 1
1 1 1
```

Вывод примера №1:

66

Код возможного решения

```
#include <iostream>
#include <cstdlib>
#include <vector>
#include <print>
#include <algorithm>

using namespace std;

int main()
{
    size_t m, n;
    int64_t a, b;
    cin >> m >> n >> a >> b;
```

```

vector<vector<int64_t>> matr(m, vector<int64_t>(n));
for (size_t r = 0; r < m; ++r) {
for (size_t c = 0; c < n; ++c) {
    cin >> matr[r][c];
}
}

vector<vector<int64_t>> ps(m, vector<int64_t>(n));
ps[0][0] = matr[0][0];
for (size_t r = 1; r < m; ++r) {
ps[r][0] = ps[r-1][0] + matr[r][0];
}
for (size_t c = 1; c < n; ++c) {
ps[0][c] = ps[0][c-1] + matr[0][c];
}
for (size_t r = 1; r < m; ++r) {
for (size_t c = 1; c < n; ++c) {
    ps[r][c] = ps[r-1][c] + ps[r][c-1] - ps[r-1][c-1] + matr[r][c];
}
}

vector<int64_t> sums;
sums.reserve(n * n * m * m);
for (size_t r2 = 0; r2 < m; ++r2) {
for (size_t c2 = 0; c2 < n; ++c2) {
    for (size_t r1 = 0; r1 <= r2; ++r1) {
        for (size_t c1 = 0; c1 <= c2; ++c1) {
            int64_t s = 0;
            if (r1 == 0 && c1 == 0) {
                s = ps[r2][c2];
            } else if (r1 == 0) {
                s = ps[r2][c2] - ps[r2][c1 - 1];
            } else if (c1 == 0) {
                s = ps[r2][c2] - ps[r1 - 1][c2];
            } else {
                s = ps[r2][c2] - ps[r1 - 1][c2] - ps[r2][c1 - 1] + ps[r1-1][c1-1];
            }
            sums.push_back(s);
        }
    }
}
}

sort(sums.begin(), sums.end());

int64_t res = 0;
for (size_t i = 0; i < sums.size(); ++i) {
    auto lb = lower_bound(sums.begin(), sums.end(), a - sums[i]);
    if (lb == sums.end()) continue;
    size_t lind = size_t(lb - sums.begin());

    auto rb = upper_bound(sums.begin(), sums.end(), b - sums[i]);
    size_t rind = sums.size();
    if (rb != sums.end()) {
        rind = rb - sums.begin();
    }
}

```

```
    if (i < lind) {  
        res += rind - lind;  
    } else if (i < rind) {  
        res += rind - i - 1;  
    }  
    }  
  
    println("{} ", res);  
}
```

Задача 6. Незнайка и король

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

Условия задачи 6

Незнайка раскладывает пасьянс с картами номинала валет, дама, король и туз, обозначаемыми заглавными кириллическими буквами В, Д, К и Т соответственно. Масть карты не имеет значения. У Незнайки неограниченное количество карт всех номиналов.

Пасьянс раскладывается по следующим правилам. На первом шаге выкладывается один валет. На i -м шаге берутся карты с $i - 1$ -го шага, справа к ним добавляется туз, а далее справа добавляются в обратном порядке сдвинутые карты с $i - 1$ -го шага. Под сдвигом понимается преобразование Валет→Дама, Дама→Король, Король→Туз, Туз→Валет.

Таким образом, мы будем по шагам получать следующую последовательность:

В
ВТД
ВТДТКВД
ВТДТКВДТКДТВКВД

Найдите, какая карта будет находиться в n -м члене этой последовательности на k -й позиции. Члены последовательности нумеруются с 1, позиция карты в члене последовательности нумеруется с 0.

Формат ввода: На вход программе подаются два целых числа: n и k : $1 \leq n \leq 60$, $0 \leq k$, k — существующая позиция в n -м члене последовательности.

Формат вывода: Выведите одну заглавную кириллическую букву в кодировке UTF-8.

Ввод примера №1:

3 4

Вывод примера №1:

К

Код возможного решения

```
use std::io::Read;

struct Solution {
}

impl Solution {
    pub fn find_kth_bit(n: u64, k: u64) -> char {
        if n == 1 {
            return '0';
        }
        let mp = (1 << (n-1)) - 1;
        if k == mp {
            return '3';
        }
    }
}
```

```

    }
    if k < mp {
        return Self::find_kth_bit(n-1, k);
    }
    match Self::find_kth_bit(n - 1, 2 * mp - k) {
        '0' => '1',
        '1' => '2',
        '2' => '3',
        '3' => '0',
        _ => panic!(),
    }
}

pub fn find_str(n: u64, k: u64) -> String {
    match Solution::find_kth_bit(n, k) {
        '0' => "B",
        '1' => "Д",
        '2' => "K",
        '3' => "T",
        _ => panic!()
    }.to_string()
}

}

fn main() -> Result<(), Box<dyn std::error::Error>> {
    let mut buf = String::new();
    std::io::stdin().read_to_string(&mut buf)?;
    let mut iter = buf.split_whitespace();
    let n: u64 = iter.next().ok_or("")?.parse()?;
    let k: u64 = iter.next().ok_or("")?.parse()?;
    println!("{}", Solution::find_str(n, k));
    Ok(())
}

```