

Задача 1. Квадродерево

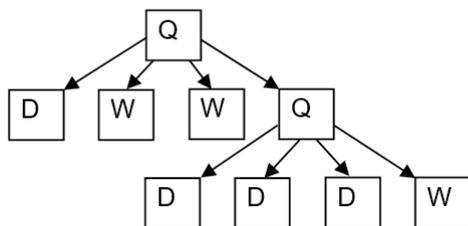
Рассмотрим способ представления плоских квадратных изображений, называемый квадродеревом:

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).

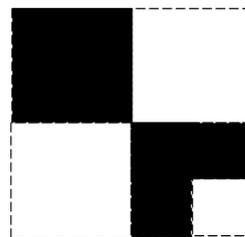
2) Если на изображении участки разного цвета, то оно делится на 4 равных квадрата (верхний левый, верхний правый, нижний левый, нижний правый) и представляется квадродеревом, состоящим из корневой вершины и четырёх поддеревьев, которые описывают части изображения – полученные при делении квадраты. Пусть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

Пример: линейная запись квадродерева: $QDWWQDDDDW$

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) и во 2-ой строке цвет P (W, R, O, Y, G, C, B, V или D). Полагая, что общая площадь записанного изображения равна 1, программа находит площадь части квадродерева T , закрашенной данным цветом P , и выводит результат в виде 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. Во 2-ой строке содержится 1 символ W, R, O, Y, G, C, B, V или D .

Формат 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:

QDWWQDDDDW

W

Вывод примера №1:

0.1001

Ввод примера №2:

R

B

Вывод примера №2:

0.0

Ввод примера №3:

R

R

Вывод примера №3:

1.0

Решение

В решении можно запрограммировать следующие подзадачи: а) считывание записи квадродерева и одновременный расчёт площади частей квадродерева, покрашенных в каждый из 9 цветов; б) считывание цвета P и вывод площади соответствующей части квадродерева. 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 $\frac{1}{4}^{H-1}$.
 При выводе результата 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 *)
       answerarray = array [1..9] of bitscale; (* 9 шкал, по 1 для каждого цвета *)
var S : stack; CH : char; I, J : word; ANSWER : answerarray;
function char2color(C : char) : byte; (* перевод символа в цвет *)
begin case C of
    'Q': char2color := 0; (* пусть Q соответствует 0 *)
    'W': char2color := 1;
    'R': char2color := 2;
    'O': char2color := 3;
    'Y': char2color := 4;
    'G': char2color := 5;
    'C': char2color := 6;
    'B': char2color := 7;
    'V': char2color := 8;
    'D': char2color := 9;
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;
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;
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(ANSWER[I], H-1);
           end (* if *)
         end (* while *)
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 ANSWER[I][J] := 0 end;
         with S do begin
           for I := 1 to STACKSIZE do begin mass[I] := 0 end;
           top := 0
         end;
         readtree(S);
         readln();
         read(CH);
         printscale(ANSWER[char2color(CH)])
end.

```

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. Вычисляется 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.

Сундуки. 7-10 класс

Скупой рыцарь хранит в подвале множество сундуков. Сундуки занумерованы, начиная с единицы. В сундуках лежат монеты. О содержимом каждого сундука есть опись. Опись пустого сундука – пустая строка. Опись непустого сундука состоит из одной или нескольких записей. Каждая запись – это сведения о стопке монет одного достоинства, лежащей в сундуке. В начале записи указано количество монет в стопке – положительное беззнаковое целое число без незначащих нулей. Количество монет в одной стопке не превышает 1000. За количеством следует символ обозначающий достоинство монет в стопке – строчная латинская буква: a означает монету номиналом 1 копейка; b – 5 копеек; c – 10 копеек; d – 50 копеек; e – 1 рубль или 100 копеек; f – 2 рубля; g – 5 рублей; h – 10 рублей; i – 25 рублей. Записи в описи никак не упорядочены, так как скупой рыцарь добавляет стопки монет в сундуки без какой-либо системы. В описи могут быть записи о стопках монет одного и того же достоинства. Например, опись $1a2b1a1h1i1h1i1h1i$ означает, что в сундуке лежат 105 рублей и 12 копеек (две монеты номиналом 1 копейка; две монеты номиналом 5 копеек; три монеты номиналом 10 рублей; три монеты номиналом 25 рублей).

Скупой рыцарь 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 скупой рыцарь выбирает из них такую пару, что сумма номеров сундуков минимальна.

Составьте программу, принимающую на вход в первой строке десятичное число N – положительное натуральное число ($2 \leq N \leq 100$) – количество сундуков, а в последующих N строках – описи сундуков с номерами от 1 до N . Известно, что в каждой описи не более чем 500 символов (цифр и латинских букв). Программа находит номера K и L такие, что $K < L$, абсолютное значение (модуль) разницы количеств денег в K -ом и L -ом сундуках наибольшее и сумма $K + L$ наименьшая. Программа выводит в первой строке найденное число K , а во второй строке – L . Каждое число записывается в десятичной системе без знака.

Формат ввода: В первой строке содержится десятичное число N – количество сундуков ($2 \leq N \leq 100$). В следующих N строках содержатся описи монет из сундуков – последовательности, в которых могут встретиться только строчные латинские буквы a, b, \dots, i и десятичные цифры. Эти последовательности записаны по правилам составления описей сундуков. Длины строк находятся в диапазоне от 0 до 500 включительно.

Формат вывода: В первой строке выводится беззнаковое десятичное натуральное число K . Во второй строке выводится беззнаковое десятичное натуральное число L . Числа K и L таковы, что $K < L$, абсолютное значение (модуль) разницы количеств денег в K -ом и L -ом сундуках наибольшее и сумма $K + L$ наименьшая.

Ввод примера №1:	Ввод примера №2:	Ввод примера №3:
2	3	4
1a2b1a1h1i1h1i1h1i	1i	1b3f2f
1c1d2f1g1e	1i	1h
Вывод примера №1:	1a1i	5e1a5e
1	Вывод примера №2:	1h1b
2	1	Вывод примера №3:
	3	1
		2

Решение

В решении можно запрограммировать следующие подзадачи: 1) считывание очередной описи сундука и представление её в виде массива из 9 элементов, в каждом из которых хранится количество монет соответствующего номинала); 2) нормализацию описи в виде массива, то есть приведение к виду, когда монет номиналом 1 копейка не более чем 4 (остальные конвертируются в 5-тикопеечные монеты), монет номиналом 5 копеек не более 1 (остальные конвертируются в 10-тикопеечные монеты), монет номиналом 10 копеек не более 4 (остальные конвертируются в 50-тикопеечные монеты), монет номиналом 50 копеек не более чем 1 (остальные конвертируются в 1-норублёвые монеты), монет номиналом 1 рубль не более 1 (остальные конвертируются в 2-хрублёвые монеты), монет номиналом 2 рубля не более чем 2 (остальные конвертируются в 5-тирублёвые монеты, при этом либо используется ещё одна 1-норублёвая монета, либо 1 рубль остатка остаётся и количество 1-норублёвых монет прирастает), монет номиналом 5 рублей не более чем 1 (остальные конвертируются в 10-тирублёвые монеты), монет номиналом 10 рублей

не более чем 2 (остальные конвертируются в 25-тирублёвые монеты, при этом либо используется ещё одна 5-тирублёвая монета, либо 5 рублей остатка остаётся и количество 5-тирублёвых монет прирастает); 3) поэлементное сравнение двух нормализованных описей в виде массивов; 4) поиск наибольшего и наименьшего элементов в последовательности описей и вывод их номеров. Для решения достаточно одного прохода по последовательности, в котором совмещены посимвольный ввод описей 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 номеров рекорда и «антирекорда» по возрастанию.

Код возможного решения

```

program TREASURECHESTS710 (input, output);
type   chests = array ['a'..'i'] of longword;
       answer = record number : word; chest : chests end;
var    CURCHEST, FIRST : chests; N, I : word; CHECK : integer; CURMAX, CURMIN : answer;
procedure readchest(var CHEST : chests); (* считывание описи сундука *)
var    CH, J : char; I, L : word;
begin
  for J := 'a' to 'i' do CHEST[J] := 0;
  if (not EOLn) then begin
    read(CH);
    I := 499;
    L := 0;
    while (I > 0) and (not EOLn) do begin
      case CH of
        'a'..'i': begin CHEST[CH] := CHEST[CH] + L; L := 0 end;
        '0'..'9': L := L * 10 + ord(CH) - ord('0');
      end;
      read(CH);
      I := I - 1;
    end;
    if (EOLn) then
      case CH of
        'a'..'i': CHEST[CH] := CHEST[CH] + L;
        '0'..'9': L := L * 10 + ord(CH) - ord('0'); (* по условию такого не бывает *)
      end;
    for J := 'a' to 'h' do (* нормализация описи сундука *)
      case J of
        'a', 'c': if (CHEST[J] > 4) then begin
          CHEST[succ(J)] := CHEST[succ(J)] + CHEST[J] div 5;
          CHEST[J] := CHEST[J] mod 5;
        end;
        'b', 'd', 'e', 'g': if (CHEST[J] > 1) then begin
          CHEST[succ(J)] := CHEST[succ(J)] + CHEST[J] div 2;
          CHEST[J] := CHEST[J] mod 2;
        end;
      end;
  end;
end;

```

```

    'f', 'h': if (CHEST[J] > 2) then begin
        CHEST[succ(J)] := CHEST[succ(J)] + (CHEST[J] * 2) div 5;
        I := (CHEST[J] * 2) mod 5;
        if odd(I) then
            if (CHEST[pred(J)] > 0) then begin
                CHEST[pred(J)] := CHEST[pred(J)] - 1;
                I := I + 1 end
            else begin
                CHEST[pred(J)] := CHEST[pred(J)] + 1;
                I := I - 1 end;
                CHEST[J] := I div 2;
            end;
        end;
    end
end
end;
function CompareChest(CHEST1, CHEST2: chests) : integer; (* сравнение денег в 2х сундуках *)
var J : char; RESULT : integer;
begin
    RESULT := 0;
    J := 'i';
    while (RESULT = 0) and (J >= 'a') do begin
        if (CHEST1[J] > CHEST2[J]) then
            RESULT := 1
        else if (CHEST1[J] < CHEST2[J]) then
            RESULT := -1;
        J := pred(J)
    end; (* while *)
    CompareChest := RESULT
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 (* обновляем максимум *)
            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 (* обновляем минимум *)
            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
        writeln(CURMIN.number);
        write(CURMAX.number) end
end;

```

```
else begin
    writeln(CURMAX.number);
    write(CURMIN.number)
end (* if *)
```

end.

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. Вычисляется 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.

Задача 3. Лифт

В одном правительственном здании установили инновационный лифт. Известно, что здание имеет как наземные этажи, нумерующиеся 1, 2, ..., так и подземные этажи, нумерующиеся $-1, -2, \dots$. Лифт изначально находится на нулевом этаже, а этаж, на который нужно подняться, кодируется нажатием на кнопки «А» и «В». Нажатие на кнопку «А» прибавляет к текущему запрошенному этажу некоторое фиксированное число (обозначим его A), а нажатие на кнопку «В» умножает текущий запрошенный этаж на некоторое другое фиксированное число (обозначим его B). При этом требуется, чтобы выбор этажа выполнялся за минимальное число нажатий, иначе лифт никуда не поедет. Значения A и B могут быть как положительными, так и отрицательными, но не равны 0. Например, пусть нам требуется попасть на 9 этаж, а значение A равно 1, а B равно 2. Тогда необходимо последовательно нажать на кнопки A, B, B, B, A : $(0 + 1) * 2 * 2 * 2 + 1$.

Формат ввода: На вход поступает число N ($0 < N < 1000$) – число запросов, затем целое число A ($-1000 < A < 1000, A \neq 0$), затем целое число B ($-1000 < B < 1000, B \neq 0$), затем N целых чисел X ($-1000 < X < 1000, X \neq 0$) – запрашиваемый этаж. Запросы не зависят друг друга, то есть каждый раз лифт едет с нулевого этажа.

Формат вывода: Если заданное число получить невозможно, выведите 0. В противном случае выведите последовательность минимальной длины из символов A и B , где A означает операцию сложения с A , а B – умножения на B , которая позволяет получить заданное число из начального значения 0. Каждый символ выводите на отдельной строке. Если подходящих последовательностей несколько, то выведите любую.

Ввод примера №1:

```
1
1
2
9
```

Вывод примера №1:

```
A
B
B
B
A
```

Ответ к задаче 3

Код возможного решения

```
/* -*- mode:c++; c-basic-offset:4 -*- */

#include <stdio.h>
#include <stdint.h>
#include <stdlib.h>

static constexpr int MAP_SIZE = 6000;
static constexpr int ZERO_BASE = 3000;
uint8_t pb[MAP_SIZE];

int queue[MAP_SIZE];
int qf, ql = 0;

int main()
{
    int n;
    scanf("%d", &n);
    int a, b;
    scanf("%d%d", &a, &b);

    pb[ZERO_BASE + a] = 1;
    queue[ql++] = a;
    while (qf != ql) {
```

```

int cur = queue[qf++];

int next = cur + a;
int zb = ZERO_BASE + next;
if (zb >= 0 && zb < MAP_SIZE && !pb[zb]) {
    pb[zb] = 1;
    queue[q1++] = next;
}

next = cur * b;
zb = ZERO_BASE + next;
if (zb >= 0 && zb < MAP_SIZE && !pb[zb]) {
    pb[zb] = 2;
    queue[q1++] = next;
}
}

for (; n; --n) {
    int x;
    scanf("%d", &x);
    if (x <= -1000 || x >= 1000) {
        printf("0\n");
    } else if (!x) {
        // nothing
    } else if (!pb[ZERO_BASE + x]) {
        printf("0\n");
    } else {
        qf = 0;
        while (x) {
            int zb = ZERO_BASE + x;
            queue[qf++] = pb[zb];
            if (pb[zb] == 1) {
                x -= a;
            } else if (pb[zb] == 2) {
                x /= b;
            } else {
                abort();
            }
        }
        while (--qf >= 0) {
            printf("%c\n", '@' + queue[qf]);
        }
    }
}
}

```

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.

Задача 4. Дешифратор

Дана программа, шифрующая файлы. Этой программе в аргументах командной строки передаются три параметра: 1) ключ шифрования – целое беззнаковое 32-битное число в десятичной записи; 2) имя входного файла; 3) имя выходного файла. Например, чтобы зашифровать файл по ссылке <https://ejudge.cs.msu.ru/lom/2023-2024/001.dat> с ключом 1234 запустите программу шифрования следующим образом:

На Linux:

```
./encoder-linux-amd64 1234 001.dat 001.res
```

На MacOS:

```
./encoder-darwin-amd64 1234 001.dat 001.res
```

На Windows:

```
encoder-windows-amd64.exe 1234 001.dat 001.res
```

В результате должен получиться файл по ссылке: <https://ejudge.cs.msu.ru/lom/2023-2024/001.res>
Ваша задача – написать программу, которая декодирует зашифрованные данной программой файлы. Ваша программа будет запускаться с тремя аргументами командной строки: 1) ключ шифрования – целое беззнаковое 32-битное число в десятичной записи; 2) имя входного (зашифрованного) файла; 3) имя выходного (расшифрованного) файла. То есть после работы программы дешифрования должен получаться файл, идентичный исходному файлу для шифрования.

Программу для шифрования можно скачать для следующих систем:

Linux/x64 <https://ejudge.cs.msu.ru/lom/2023-2024/encoder-linux-amd64>

Linux/x86 <https://ejudge.cs.msu.ru/lom/2023-2024/encoder-linux-386>

MacOS/x64 <https://ejudge.cs.msu.ru/lom/2023-2024/encoder-darwin-amd64>

MacOS/M1(M2) <https://ejudge.cs.msu.ru/lom/2023-2024/encoder-darwin-arm64>

Windows <https://ejudge.cs.msu.ru/lom/2023-2024/encoder-windows-amd64.exe>

Windows (32 bit) <https://ejudge.cs.msu.ru/lom/2023-2024/encoder-windows-386.exe>

Внимание! Программа делает что-то похожее на то, что делают вирусы-шифровальщики. Поэтому возможны ложные срабатывания некоторых «антивирусов». Если Вы по каким-либо причинам опасаетесь запускать программу на локальном компьютере, то Вы можете использовать сторонние сервисы, например, такой: <https://bellard.org/jslinux/> и запускать программу в Alpine Linux.

Код возможного решения

```
import base64
import sys
def reverse(x, n):
    result = 0
    for i in range(n):
        if (x >> i) & 1: result |= 1 << (n - 1 - i)
    return result
def transfigure_key(f):
    return ((32 * (((f >> 17) ^ (f >> 16) ^ (f >> 18) ^ (f >> 29)) ^ (f >> 28)) & 1)) ^
    (16 * (((f >> 19) ^ (f >> 15) ^ (f >> 10) ^ (f >> 13)) ^ (f >> 11)) & 1)) ^
    (8 * (((f >> 14) ^ (f >> 27) ^ (f >> 5) ^ (f >> 25)) ^ (f >> 22)) & 1)) ^
    (4 * (((f >> 24) ^ (f >> 1) ^ (f >> 6) ^ (f >> 8)) ^ (f >> 26)) & 1)) ^
    (2 * (((f ^ (f >> 3) ^ (f >> 7) ^ (f >> 31) ^ (f >> 9)) ^ (f >> 2)) & 1)) ^
    ((f >> 21) ^ ((f >> 23) ^ (f >> 12) ^ (f >> 4) ^ (f >> 30) ^ (f >> 20))) & 1);
with open(sys.argv[2], 'rb') as f:
    ciphertext = bytearray(f.read())

xoredKey = int(sys.argv[1]) ^ 0x42953DE6
for i in range(len(ciphertext)):
    ciphertext[i] ^= ((xoredKey >> (8 * (i & 3))) & 0xff)
    ciphertext[i] = reverse(ciphertext[i], 8)
```

```

x = ciphertext[::-1]
alphabet = b'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789-_
ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789-_
ciphertextBase64 = x
#print(ciphertextBase64[:500])
shift = transfigure_key(xoredKey)
#print(shift)
out = ''
for x in ciphertextBase64:
    idx = alphabet.index(x) - shift
    if idx < 0:
        idx += len("ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789-_")
    out += chr(alphabet[idx])
outVal = base64.urlsafe_b64decode(out + '==')
with open(sys.argv[3], 'wb') as g:
    g.write(outVal)

```

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. Вычисляется 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.

Задача 5. Аномальные материалы

Лаборатория аномальных материалов изучает свойства анобтаниума. Из анобтаниума изготовлен штырь длины K см (K – целое положительное число). В торцах штыря и через каждый сантиметр длины штыря размещены датчики температуры (всего $K + 1$ датчик). Эксперимент заключается в последовательности нагревов отрезков штыря $[l; m]$, где l и m – целые положительные числа, обозначающие сантиметры от начала штыря. В результате при каждом нагреве датчики температуры на интервале $[l; m]$ фиксируют повышение температуры на v градусов. В лабораторном журнале фиксируются все нагревы штыря. Начальная температура всего штыря равна 0.

Анобтаниум – уникальный материал с крайне малой теплопроводностью, то есть нагрев отрезка штыря никак не влияет на датчики температуры, не находящиеся на нагреваемом отрезке.

Напишите программу, которая по лабораторному журналу нагреваний штыря найдет сумму показаний температур для заданных отрезков штыря в заданные моменты времени.

Формат ввода: На вход подается целое число K ($1 \leq K \leq 10^7$) – длина штыря. Затем подается целое число N ($0 \leq N \leq 10^5$) – количество записей в журнале экспериментов. Затем вводятся записи журнала экспериментов. Каждая запись представляет собой четыре целых числа (l, r, t, v) , где l ($0 \leq l \leq K$) – левая граница нагреваемого отрезка, r ($0 \leq r \leq K; l \leq r$) – правая граница нагреваемого отрезка, t ($0 \leq t \leq 10^9$) – момент времени от начала эксперимента, когда выполнялось нагревание, v ($-10^9 \leq v \leq 10^9$) – на сколько изменилась температура на отрезке.

К сожалению, записи в журнале перепутались, и они могут идти не в порядке неубывания времени. В один момент времени может выполняться несколько нагреваний, в том числе и на пересекающихся отрезках.

Далее идёт поток запросов о состоянии отрезка. Сначала подаётся целое число M ($0 \leq M \leq 10^5$), за которым следуют M запросов. Каждый запрос представляет собой три числа (l, r, t) , где l ($0 \leq l \leq K$) – левая граница отрезка, r ($0 \leq r \leq K; l \leq r$) – правая граница отрезка, t ($0 \leq t \leq 10^9$) – момент времени.

Формат вывода: Для каждого запроса выведите сумму температур всех датчиков на заданном отрезке в заданный момент времени. Если в указанный момент времени выполнялись нагревания отрезка, то суммирование производится после нагревания (то есть с новыми значениями датчиков температур).

Ввод примера №1:

```
3 12
1 1 1 1
1 1 2 2
1 1 3 3
1 1 4 10
1 1 5 12
1 1 6 11
1 1 7 13
1 2 8 4
2 2 1 100
1 2 10 4
1 2 11 9
1 3 12 10
5
1 1 2
1 2 3
1 3 5
1 3 11
1 3 15
```

Вывод примера №1:

```
3
106
128
186
216
```

Код возможного решения

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

using namespace std;

#define int long long

struct node {
    int lson, rson, val, push, sz;
    node(int _sz) {
        lson = rson = -1;
        val = push = 0;
        sz = _sz;
    }
};

struct segtree {
    vector <node> t;
    int n;
    segtree(int _n) {
        n = _n;
        t.emplace_back(n);
    }
    void edit(int v, int val) {
        t[v].val += t[v].sz * val;
        t[v].push += val;
    }
    void pls(int v, int l, int r) {
        int m = (r + 1) / 2;
        if (t[v].lson == -1) {
            t.emplace_back(m - 1);
            t[v].lson = (int)t.size() - 1;
        }
        if (t[v].rson == -1) {
            t.emplace_back(r - m);
            t[v].rson = (int)t.size() - 1;
        }
    }
    void make_push(int v) {
        if (!t[v].push)
            return;

        int zn = t[v].push;
        t[v].push = 0;

        int l = t[v].lson;
        int r = t[v].rson;

        edit(l, zn);
        edit(r, zn);
    }
    void inc(int v, int l, int r, int ql, int qr, int val) {
        if (ql >= r || l >= qr)
            return;
    }
};
```

```

        if (l >= ql && r <= qr) {
            edit(v, val);
            return;
        }
        int m = (r + 1) / 2;
        pls(v, l, r);
        make_push(v);
        inc(t[v].lson, l, m, ql, qr, val);
        inc(t[v].rson, m, r, ql, qr, val);
        t[v].val = t[t[v].lson].val + t[t[v].rson].val;
    }
    int get(int v, int l, int r, int ql, int qr) {
        if (ql >= r || l >= qr)
            return 0;
        if (l >= ql && r <= qr) {
            return t[v].val;
        }
        int m = (r + 1) / 2;
        pls(v, l, r);
        make_push(v);
        return get(t[v].lson, l, m, ql, qr) + get(t[v].rson, m, r, ql, qr);
    }
    void inc(int l, int r, int val) {
        inc(0, 0, n, l, r, val);
    }
    int get(int l, int r) {
        return get(0, 0, n, l, r);
    }
};
struct query {
    int l, r, i, t, v;
};
bool cmp(query a, query b) {
    if (a.i != b.i)
        return a.i < b.i;
    return a.t < b.t;
}
void solve() {
    int k, n;
    cin >> k >> n;
    vector <query> zxc;
    for (int i = 0; i < n; i++) {
        query tmp;
        cin >> tmp.l >> tmp.r >> tmp.i >> tmp.v;
        tmp.t = 1;
        zxc.push_back(tmp);
    }
    int m;
    cin >> m;
    for (int i = 0; i < m; i++) {
        query tmp;
        cin >> tmp.l >> tmp.r >> tmp.i;
        tmp.t = 2;
        tmp.v = i;
        zxc.push_back(tmp);
    }
}

```

```

sort(zxc.begin(), zxc.end(), cmp);
vector <int> ans(m);
segtree hvick(k + 1);
for (auto& [l, r, i, t, v] : zxc) {
    //cout << l << " " << r << " " << t << " " << v << " " << i << "\n";
    if (t == 1) {
        hvick.inc(l, r + 1, v);
    }
    else {
        ans[v] = hvick.get(l, r + 1);
    }
}
for (auto el : ans) {
    cout << el << "\n";
}
}
signed main()
{
    ios_base::sync_with_stdio(false);
    cin.tie(0);
    cout.tie(0);
    int t = 1;
    while (t--)
        solve();
}

```

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. Вычисляется 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.

Перевод технических баллов в оценку

По каждой задаче определялись самые удачные её решения, отправленные участником. Технические баллы за них суммировались. Вычислялась доля (в процентах), которую составила полученная сумма по отношению к 400 техническим баллам. Полученное количество процентов, округлённое до целого, становилось оценкой.