Максимальная сумма баллов: 100

Задание 1

Задача 1. count79: Количество вхождений цифры

В задаче рассмотрим две симметричные системы счисления (далее будем писать сокращённо - ССС): троичную ССС и девятеричную ССС. В троичной ССС используются цифры Z, 0, 1. К записи числа в этой системе приписывают снизу окончание $_{S3}$. $Z_{S3}=-1_{10}$, $0_{S3}=0_{10}$, $1_{S3}=1_{10}$. Если имеется запись в троичной ССС $a_n a_{n-1}...a_1 a_{0S3}$, то она обозначает число, равное $a_n*3^n+a_{n-1}*3^{n-1}+...+a_1*3+a_0*1$. Например, $623_{10}=729-81-27+3-1=1*3^6+0*3^5+(-1)*3^4+(-1)*3^3+0*3^2+1*3+(-1)*1=10ZZ01Z_{S3}$. В девятиричной ССС используются цифры W, X, Y, Z, 0, 1, 2, 3, 4. К записи числа в этой системе приписывают снизу окончание $_{S9}$. $W_{S9}=-4_{10}$, $X_{S9}=-3_{10}$, $Y_{S9}=-2_{10}$, $Z_{S9}=-1_{10}$, $0_{S9}=0_{10}$, $1_{S9}=1_{10}$, $2_{S9}=2_{10}$, $3_{S9}=3_{10}$, $4_{S9}=4_{10}$. Если имеется запись в девятеричной ССС $a_n a_{n-1}...a_1 a_{0S9}$, то она обозначает число, равное $a_n*9^n+a_{n-1}*9^{n-1}+...+a_1*9+a_0*1$. Например, $623_{10}=729-81-27+2=1*9^3+(-1)*9^2+(-3)*9+2*1=1ZX2_{S9}$. В симметричных системах счисления перед записью числа не ставят ни плюс, ни минус, и когда число положительное, и когда оно отрицательное.
Составьте программу, которая принимает на вход в первой строке цифру J - одну из цифр девятеричной ССС {W, ..., Z, 0, 1, ..., 4}, во второй строке целое число K, записанное в троичной ССС. В этой записи используются десятичные цифры 0, 1 и заглавная латинская буква Z. Длина записи числа K в троичной ССС не более чем 100000. Программа находит, количество вхождений цифры J в запись числа K, если его перевести в девятеричную ССС. В начале записи числа K могут стоять незначащие нули, которые не следует учитывать при подсчёте количества вхождений J=0. Незначащим является любой нуль, стоящий левее первой ненулевой цифры, или, если K = 0, то все нули, кроме самого правого.

Формат входных данных

В первой строке содержится символ J - цифра девятеричной ССС (одна из {W, ..., Z, 0, 1, ..., 4}).
Во второй строке содержится непустая последовательность символов, являющаяся записью в троичной ССС целого числа K (в этой записи не более чем 100000 символов). В записи числа K используются символы из набора {Z, 0, 1}.

Формат выходных данных

В первой и единственной строке выводится неотрицательное целое число (от 0 до 50000), равное искомому количеству вхождений цифры J в запись числа K в девятеричной ССС.

Баллы: 33.333333

Решение:

```
program DigitQty1011 (input, output);

var

    C, CPREV : char;

    FLAG : boolean;

    J, TEMP, TEMP1 : integer;

    IIID1, IIID2, IIID3, IIID4, IIID5, IIID6 : integer;

    (* разр€ды записи в 3-й с.с.с. *)

    ANSWER1, ANSWER2 : int32;

function XXVIIDigitValue(C : char) : integer;
```

```
begin   (* находим значение разр€да записи в 27-й с.с.с. *)
      if ((C <= '9') and (C >= '0')) then XXVIIDigitValue := ord(C) - ord('0')
      else if ((C <= 'D') and (C >= 'A'))
        then XXVIIDigitValue := ord(C) - ord('A') + 10
      else XXVIIDigitValue := ord(C) - ord('Z') - 1;
end;
function IXDigitValue(C : char) : integer;
begin   (* находим значение разр€да записи в 9-й с.с.с. *)
      if ((C <= '4') and (C >= '0')) then IXDigitValue := ord(C) - ord('0')
      else IXDigitValue := ord(C) - ord('Z') - 1;
end;
procedure XXVIIToIII(XXVII : integer; var IIID1, IIID2, IIID3 : integer);
begin   (* разр€д записи в 27-й с.с.с. переводим в три разр€да записи в 3-й с.с.с. *)
      if XXVII < -4 then IIID1 := -1
      else if XXVII < 5 then IIID1 := 0
      else IIID1 := 1;
      IIID3 := (XXVII + 27) mod 3;
      if (IIID3 = 2) then IIID3 := -1;
      IIID2 := (XXVII - IIID1 * 9 - IIID3) div 3;
end;
function CountJDigit1(C: char) : int32;
(* подсчитать вхождени€ J в один разр€д записи в 27-й с.с.с.*)
var     I : int32;
      IIID1, IIID2, IIID3 : integer;
      (* разр€ды записи в 3-й с.с.с. *)
begin
      XXVIIToIII(XXVIIDigitValue(C), IIID1, IIID2, IIID3);
      I := 0;
      if ((IIID1 = J) and (J <> 0))then inc(I);
      if ((IIID2 * 3 + IIID3) = J) then inc(I);
      CountJDigit1 := I
end;
```

```
function CountJDigit2(C1, C2: char) : int32;
(* подсчитать вхождени€ J в два разр€да записи в 27-й с.с.с. без незначащих 0 *)
var    I : int32;
       IIID1, IIID2, IIID3, IIID4, IIID5, IIID6 : integer;
       (* разр€ды записи в 3-й с.с.с. *)
begin
       XXVIIToIII(XXVIIDigitValue(C1), IIID1, IIID2, IIID3);
       XXVIIToIII(XXVIIDigitValue(C2), IIID4, IIID5, IIID6);
       I := 0;
       if ((IIID1 * 3 + IIID2) = J) then inc(I);
       if ((IIID3 * 3 + IIID4) = J) then inc(I);
       if ((IIID5 * 3 + IIID6) = J) then inc(I);
       CountJDigit2 := I
end;
function CountJDigit3(C1, C2: char) : int32;
(* подсчитать вхождени€ J в два разр€да записи в 27-й с.с.с. с незначащим 0 *)
var    I : int32;
       IIID1, IIID2, IIID3, IIID4, IIID5, IIID6 : integer;
       (* разр€ды записи в 3-й с.с.с. *)
begin
       XXVIIToIII(XXVIIDigitValue(C1), IIID1, IIID2, IIID3);
       XXVIIToIII(XXVIIDigitValue(C2), IIID4, IIID5, IIID6);
       I := 0;
       if (((IIID1 * 3 + IIID2) = J) and (J <> 0)) then inc(I);
       if ((IIID3 * 3 + IIID4) = J) then inc(I);
       if ((IIID5 * 3 + IIID6) = J) then inc(I);
       CountJDigit3 := I
end;


begin
       readln(C);
       J := IXDigitValue(C);
```

```
    C := '0';
    CPREV := '0';
    while ((not EOLn) and (C = '0')) do begin
        read(C);
    end; (* while *)
                ANSWER1 := CountJDigit1(C);
                ANSWER2 := 0;
                FLAG := TRUE; (* признак нечетной длины *)
                if (not EOLn) then begin
        CPREV:= C;
        read(C);
        FLAG := not FLAG;
        ANSWER2 := CountJDigit3(CPREV, C)
    end; (* if *)
    while (not EOLn) do begin
        CPREV:= C;
        read(C);
        FLAG := not FLAG;
        if FLAG then
            ANSWER1 := ANSWER1 + CountJDigit2(CPREV, C)
        else ANSWER2 := ANSWER2 + CountJDigit2(CPREV, C)
    end; (* обработка в цикле пар последних считанных разр€дов записи в 27-й с.с.с. *)
    if FLAG then writeln(ANSWER1) else writeln(ANSWER2);
end.
```

Задание 2.

Задача 2. vampir79: Вампиры

В тексте будем использовать девятеричную симметричную систему счисления (сокращённо ССС), описанную в условиях задачи "Количество вхождений цифры". Рассмотрим целые положительные числа специального вида, которые назовём вампиры$_{S9}$. Каждое число, являющееся вампиром$_{S9}$, обладает всеми следующими свойствами: 1) длина его записи в девятеричной ССС является чётной (рассматривается запись без незначащих нулей в начале); 2) оно является произведением двух своих клыков -- целых положительных множителей, не равных друг другу, у которых длина записи в девятеричной ССС равна половине длины записи в девятеричной ССС вампира$_{S9}$; 3) если запись в девятеричной ССС одного из клыков оканчивается нулём, то у другого клыка последняя цифра в записи в девятеричной ССС не может быть нулём; 4) если выписать друг за другом слитно записи в девятеричной ССС клыков, то можно так переставить цифры в образовавшейся записи, что получится запись в девятеричной ССС вампира$_{S9}$. Например, $1008_{10} = 1340_{S9} = 28_{10} * 36_{10} = 31_{S9} * 40_{S9}$, значит 1008 является вампиром$_{S9}$. Второй пример, $81648_{10} = 134000_{S9} = 252_{10} * 324_{10} = 310_{S9} * 400_{S9}$, значит 81648 не является вампиром$_{S9}$, так как нарушено условие о том, что записи обоих клыков в девятеричной ССС не могут заканчиваться нулём. Третий пример, $77616_{10} = 13Y420_{S9} = 252_{10} * 308_{10} = 310_{S9} * 4Y2_{S9}$, значит 77616 является вампиром$_{S9}$, так как только у одного клыка запись в девятеричной ССС заканчивается нулём.
Составьте программу, которая принимает на вход в первой строке целое положительное число N, а во второй строке последовательность из N целых положительных чисел V_i, разделённых пробелами. Число N не более чем 50000. Числа V_i не более чем 2500000. Программа подсчитывает количество чисел V_i, которые являются вампирами$_{S9}$, и выводит результат подсчёта. Если в последовательности более одного раза встречается одно и то же число, являющееся вампиром$_{S9}$, то при подсчёте учитываются все вхождения такого числа, как если бы это были вхождения разных чисел.

Формат входных данных

В первой строке содержится целое положительное число N — длина последовательности чисел, ($0 < N < 50001$).
Во второй строке содержится N целых положительных чисел V_i, разделённых пробелами ($0 < V_i < 2500001$).

Формат выходных данных

В первой и единственной строке выводится неотрицательное целое число (от 0 до 50000), равное искомому количеству вхождений во входную последовательность чисел, являющихся вампирами$_{S9}$.

Баллы: 33.333333

Решение:

```
program Fangs1011 (input, output);

type    PairOfFangs = record

          first: word;

          second: word;
```

```
    end;
    Check = -1..1;
    Numbers = array [ 1..800] of word;
const
    NUMPAIRS = 265;
    FANGS : array [ 1..NUMPAIRS ] of PairOfFangs = (
      ( first: 28; second: 36),
      ( first: 136; second: 312),
      ( first: 138; second: 298),
      ( first: 154; second: 210),
      ( first: 154; second: 362),
      ( first: 172; second: 204),
      ( first: 244; second: 252),
      ( first: 244; second: 324),
      ( first: 252; second: 308),
      ( first: 252; second: 348),
      ( first: 258; second: 298),
      ( first: 788; second: 3132),
      ( first: 788; second: 3156),
      ( first: 856; second: 2976),
      ( first: 860; second: 2916),
      ( first: 872; second: 2912),
      ( first: 880; second: 2984),
      ( first: 896; second: 3240),
      ( first: 908; second: 2876),
      ( first: 916; second: 3124),
      ( first: 922; second: 3042),
      ( first: 932; second: 2684),
      ( first: 936; second: 2800),
      ( first: 948; second: 3124),
      ( first: 968; second: 2664),
      ( first: 968; second: 2976),
```

(first: 968; second: 3048),

(first: 980; second: 2660),

(first: 980; second: 2948),

(first: 984; second: 2608),

(first: 986; second: 3138),

(first: 1018; second: 2658),

(first: 1020; second: 2564),

(first: 1032; second: 2800),

(first: 1044; second: 2516),

(first: 1050; second: 2554),

(first: 1062; second: 2582),

(first: 1088; second: 2408),

(first: 1106; second: 3106),

(first: 1108; second: 2252),

(first: 1108; second: 2892),

(first: 1108; second: 2988),

(first: 1110; second: 3022),

(first: 1114; second: 2602),

(first: 1114; second: 2674),

(first: 1120; second: 2152),

(first: 1126; second: 2326),

(first: 1126; second: 2542),

(first: 1138; second: 2314),

(first: 1138; second: 2458),

(first: 1158; second: 3022),

(first: 1168; second: 2960),

(first: 1170; second: 2386),

(first: 1170; second: 2442),

(first: 1178; second: 2530),

(first: 1196; second: 3268),

(first: 1206; second: 3022),

(first: 1212; second: 2836),

(first: 1216; second: 2816),

(first: 1220; second: 2308),

(first: 1222; second: 2310),

(first: 1222; second: 2526),

(first: 1222; second: 2910),

(first: 1224; second: 2688),

(first: 1224; second: 2896),

(first: 1226; second: 2682),

(first: 1246; second: 3142),

(first: 1252; second: 2236),

(first: 1260; second: 2908),

(first: 1270; second: 2430),

(first: 1270; second: 2622),

(first: 1270; second: 2798),

(first: 1270; second: 2942),

(first: 1278; second: 2566),

(first: 1282; second: 2890),

(first: 1292; second: 1876),

(first: 1292; second: 2836),

(first: 1292; second: 2844),

(first: 1294; second: 2958),

(first: 1300; second: 2236),

(first: 1304; second: 1896),

(first: 1304; second: 3184),

(first: 1304; second: 3192),

(first: 1324; second: 2732),

(first: 1324; second: 2916),

(first: 1324; second: 2964),

(first: 1324; second: 2980),

(first: 1330; second: 2674),

(first: 1342; second: 3278),

(first: 1372; second: 3268),

(first: 1378; second: 3258),

(first: 1380; second: 2436),

(first: 1382; second: 1998),

(first: 1386; second: 2482),

(first: 1410; second: 3170),

(first: 1420; second: 1756),

(first: 1422; second: 1950),

(first: 1426; second: 2242),

(first: 1436; second: 3268),

(first: 1438; second: 2598),

(first: 1438; second: 3214),

(first: 1458; second: 2242),

(first: 1458; second: 2274),

(first: 1466; second: 2858),

(first: 1472; second: 3032),

(first: 1482; second: 1874),

(first: 1500; second: 2332),

(first: 1502; second: 1870),

(first: 1508; second: 2748),

(first: 1510; second: 2238),

(first: 1512; second: 2080),

(first: 1514; second: 2034),

(first: 1518; second: 2422),

(first: 1522; second: 1826),

(first: 1522; second: 2834),

(first: 1526; second: 2846),

(first: 1528; second: 1864),

(first: 1528; second: 2664),

(first: 1534; second: 2238),

(first: 1540; second: 3196),

(first: 1542; second: 2574),

(first: 1548; second: 2812),

(first: 1548; second: 2908),

(first: 1554; second: 2138),

(first: 1560; second: 3160),

(first: 1566; second: 2230),

(first: 1566; second: 2494),

(first: 1566; second: 2654),

(first: 1566; second: 2806),

(first: 1566; second: 2838),

(first: 1584; second: 2848),

(first: 1600; second: 2240),

(first: 1602; second: 2442),

(first: 1602; second: 2922),

(first: 1638; second: 2998),

(first: 1648; second: 2000),

(first: 1664; second: 2680),

(first: 1666; second: 1746),

(first: 1676; second: 2332),

(first: 1692; second: 2908),

(first: 1694; second: 2926),

(first: 1698; second: 2962),

(first: 1702; second: 2750),

(first: 1702; second: 2950),

(first: 1702; second: 2958),

(first: 1706; second: 2986),

(first: 1710; second: 2894),

(first: 1710; second: 2910),

(first: 1712; second: 1944),

(first: 1734; second: 2230),

(first: 1744; second: 2112),

(first: 1756; second: 1940),

(first: 1758; second: 2230),

(first: 1776; second: 2440),

(first: 1780; second: 2980),

(first: 1782; second: 2998),

(first: 1790; second: 3174),

(first: 1852; second: 1868),

(first: 1864; second: 2960),

(first: 1870; second: 3078),

(first: 1882; second: 3058),

(first: 1882; second: 3178),

(first: 1882; second: 3242),

(first: 1912; second: 2112),

(first: 1914; second: 2026),

(first: 1942; second: 2566),

(first: 1954; second: 2066),

(first: 1954; second: 3178),

(first: 1968; second: 2512),

(first: 1980; second: 3084),

(first: 1990; second: 3246),

(first: 1998; second: 3142),

(first: 2000; second: 3024),

(first: 2008; second: 2600),

(first: 2024; second: 3160),

(first: 2026; second: 3258),

(first: 2034; second: 2114),

(first: 2036; second: 3052),

(first: 2072; second: 2760),

(first: 2080; second: 3200),

(first: 2086; second: 3022),

(first: 2106; second: 3002),

(first: 2110; second: 2358),

(first: 2112; second: 2968),

(first: 2116; second: 2348),

(first: 2122; second: 2682),

(first: 2126; second: 2998),

(first: 2160; second: 2288),

(first: 2182; second: 2286),

(first: 2188; second: 2196),

(first: 2188; second: 2268),

(first: 2188; second: 2276),

(first: 2188; second: 2916),

(first: 2194; second: 3018),

(first: 2196; second: 3012),

(first: 2196; second: 3156),

(first: 2196; second: 3228),

(first: 2198; second: 2558),

(first: 2202; second: 2962),

(first: 2206; second: 2766),

(first: 2206; second: 2998),

(first: 2224; second: 2976),

(first: 2226; second: 2250),

(first: 2228; second: 2692),

(first: 2238; second: 3014),

(first: 2240; second: 2576),

(first: 2240; second: 2944),

(first: 2246; second: 2782),

(first: 2252; second: 2836),

(first: 2254; second: 2790),

(first: 2268; second: 2764),

(first: 2268; second: 2796),

(first: 2268; second: 2972),

(first: 2268; second: 3108),

(first: 2294; second: 2718),

(first: 2296; second: 2736),

(first: 2310; second: 3006),

(first: 2312; second: 2680),

(first: 2316; second: 2844),

(first: 2320; second: 2584),

(first: 2320; second: 2928),

(first: 2320; second: 3024),

(first: 2322; second: 3010),

(first: 2326; second: 2678),

(first: 2334; second: 3006),

(first: 2336; second: 2512),

(first: 2336; second: 3256),

(first: 2350; second: 2574),

(first: 2350; second: 2614),

(first: 2350; second: 2814),

(first: 2350; second: 2966),

(first: 2350; second: 2974),

(first: 2352; second: 2944),

(first: 2356; second: 2908),

(first: 2356; second: 3036),

(first: 2358; second: 2766),

(first: 2368; second: 2576),

(first: 2378; second: 2978),

(first: 2382; second: 3006),

(first: 2386; second: 2586),

(first: 2394; second: 2674),

(first: 2402; second: 2746),

(first: 2420; second: 2620),

(first: 2420; second: 2692),

(first: 2434; second: 2922),

(first: 2436; second: 2772),

(first: 2436; second: 2916),

(first: 2458; second: 2890),

(first: 2466; second: 2674),

(first: 2470; second: 2854),

```
        ( first: 2502; second: 2854),
        ( first: 2502; second: 3174),
        ( first: 2524; second: 2676),
        ( first: 2580; second: 2764),
        ( first: 2584; second: 2592),
        ( first: 2634; second: 2938),
        ( first: 2690; second: 3042),
        ( first: 2772; second: 3116),
        ( first: 2778; second: 3050),
        ( first: 2806; second: 3102),
        ( first: 2898; second: 3018),
        ( first: 3014; second: 3038),
        ( first: 3076; second: 3196),
        ( first: 3110; second: 3158),
        ( first: 14020; second: 14300));
var
    N, I, J, ANSWER : word;
    F : Numbers;
function S9Length(F1 : word): word;
(* количество разрядов в записи числа в ссс9*)
var
    ANSWER: word;
begin
    case F1 of
        0..4: ANSWER := 1;
        5..40: ANSWER := 2;
        41..364: ANSWER := 3;
        365..3280: ANSWER := 4;
        else ANSWER := 5;
    end; (* case *)
    S9Length := ANSWER;
end; (* S9Length *)
```

```
function CmpFangs(F1, F2: word; var PAIR: PairOfFangs): Check;
(* сравнение пары чисел с парой клыков *)
var
      ANSWER: Check;
begin
      with PAIR do begin
            if (F1 < first) then ANSWER := -1
            else if (F1 = first) then
                  if (F2 < second) then ANSWER := -1
                  else if (F2 = second) then ANSWER := 0
                  else ANSWER := 1
            else ANSWER := 1
      end; (* with *)
      CmpFangs := ANSWER;
end; (* CmpFangs *)
function BinarySearch(F1, F2: word): boolean;
(* бинарный поиск в массиве пар клыков *)
var
   L, M, H : word;
          ANSWER : boolean;
begin
   L := 1;
   H := NUMPAIRS;
   ANSWER := FALSE;
   while ((not ANSWER) and (L <= H)) do
   begin
      M := (L + H) div 2;
      case CmpFangs(F1, F2, FANGS[M]) of
          -1: H := M - 1;
           1: L := M + 1;
          else ANSWER := TRUE;
      end; (* case *)
```

```
   end; (* while *)
          BinarySearch := ANSWER
end; (* BinarySearch *)
procedure Sort(L, R: word);
(* сортировка *)
    var
      I, J, X, Y: word;
    begin
      I := L;
      J := R;
      X := F[(L + R) div 2];
      repeat
        while (F[I] < X) do inc(I);
        while (X < F[J]) do dec(J);
        if (I <= J) then begin
            Y := F[I];
            F[I] := F[J];
            F[J] := Y;
            inc(I);
            dec(J);
        end; (* if *)
      until (I > J);
      if (L < J) then Sort(L, J);
      if (I < R) then Sort(I, R);
end; (* Sort *)
begin
      readln(N);
      for I := 1 to N do read(F[I]);
      for I := N+1 to 800 do F[I] := 0;
      Sort(1, N);
      ANSWER := 0;
      for I := 1 to N-1 do begin
```

```
        for J := I+1 to N do
          if ((S9Length(F[I]) = S9Length(F[J])) and BinarySearch(F[I], F[J]))
          then inc(ANSWER)
      end; (* for *)
      writeln(ANSWER)
end.
```

Задание 3.

Задача 3. distance: Социальная дистанция

В одном университете озаботились соблюдением социальной дистанции в студенческой столовой. Зал столовой можно представить как прямоугольную сетку

размера N по вертикали на M по горизонтали, в каждой клетке которой может сесть посетитель.
В момент времени s_i, когда приходит посетитель администратор узнает момент времени f_i, когда посетитель уйдет и выбирает свободное место исходя из требований:

- место должно быть свободным
- место посетителя максимально удалено от занятых мест в зале,
- если таких мест несколько, то выбираются места с минимальной координатой y по вертикали
- если таких мест несколько, то выбираются места с минимальной координатой x по горизонтали

Расстояние между точками (y_i, x_i) и (y_j, x_j) вычисляется как $L = abs(x_i - x_j) + abs(y_i-y_j)$
Вам требуется написать программу, которая для каждого посетителя определяет место, куда ему сесть.

Формат входных данных

В первой строке задаются размеры зала N, M ($1 \le N, M \le 100$) и количество посетителей K ($1 \le K \le 5000$).
В следующих K строках информация о посетителях s_i, f_i - время прихода и ухода. Времена до 10^9, гарантируется что все времена различные и $s_i < f_i$. Посетители упорядочены по s_i.

Формат выходных данных

Для каждого посетителя выведите координаты y, x ($1 \le y \le N$, $1 \le x \le M$). Если свободных мест в зале нет, выведите -1 -1 для этого посетителя

Примеры

Баллы: 33.33333

Решение:

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

```
#include <algorithm>
#include <cmath>
#include <map>
#include <cstdio>

#define pb push_back

using namespace std;

typedef pair<int, int> ii;
typedef vector<int> vi;

inline int sqr(int x) { return x * x; }
inline int abs(int x) { return x > 0 ? x : -x; }

const int inf = 1000 * 1000 * 1000;

int n, m;

map<int, ii> wh;

int ri, rj;

/*inline int distMin(int x, int x1, int x2) {
	if (x >= x1 && x <= x2) return 0;
	return min(abs(x - x1), abs(x - x2));
}*/

inline int distMin(int x, int y, int x1, int y1, int x2, int y2) {
	return
		max({abs(x-x1) + abs(y-y1),
		   abs(x-x2) + abs(y-y2),
```

```
                    abs(x-x1) + abs(y-y2),
                    abs(x-x2) + abs(y-y1)}
                );
}

// 4 похожих друг на друга квадратичных спуска
bool checkInt(int dist, int x1, int y1, int x2, int y2) {
        if (x2 < x1) return false;
        if (y2 < y1) return false;

        bool U = false;

        for (const auto & [t, p] : wh) {
                if (distMin(p.first, p.second, x1, y1, x2, y2) < dist)
                        return false;
                if (p.first == x1 && p.second == y1) U = true;
        }

        if (x1 == x2 && y1 == y2) {
                if (U) {
                        return false;
                }
                        ri = x1, rj = y1;
                return true;
        }

        int mx = (x1 + x2) / 2;
        int my = (y1 + y2) / 2;
        if (checkInt(dist, x1, y1, mx, my)) return true;
        if (checkInt(dist, x1, my + 1, mx, y2)) return true;
        if (checkInt(dist, mx + 1, y1, x2, my)) return true;
        if (checkInt(dist, mx + 1, my + 1, x2, y2)) return true;
```

```
        return false;
}

bool checkIntFull(int dist, int x1, int y1, int x2, int y2) {
        if (x2 < x1) return false;
        if (y2 < y1) return false;
        if (x1 > ri || x1 == ri && y1 > rj) return false;

        bool U = false;

        for (auto & [t, p] : wh) {
                if (distMin(p.first, p.second, x1, y1, x2, y2) < dist)
                        return false;

                if (p.first == x1 && p.second == y1) U = true;
        }

        if (x1 == x2 && y1 == y2) {
                if (U) {
                        return false;
                }
                if (x1 < ri || x1 == ri && y1 < rj)
                        ri = x1, rj = y1;
                return true;
        }

        int mx = (x1 + x2) / 2;
        int my = (y1 + y2) / 2;
        bool res = false;
        if (checkIntFull(dist, x1, y1, mx, my)) res = true;
        if (checkIntFull(dist, x1, my + 1, mx, y2)) res = true;
```

```
        if (checkIntFull(dist, mx + 1, y1, x2, my)) res = true;
        if (checkIntFull(dist, mx + 1, my + 1, x2, y2)) res = true;

        return res;
}

vector<vector<vector<int>>> v3(204, vector<vector<int>>(104, vector<int>(104,0)));

bool checkNewInt(int dist, int n,  int m, int &pi, int &pj) {
   for (int i = 0; i < n; ++i) {
        int s = 0;
        for (int j = 0; j < m; ++j) {
          s += v3[dist][i][j];
          if (s == 0) {
                pi = i;
                pj = j;
            return true;
          }
        }
   }
   return false;
}

void print(const vector<vector<int>> &v, int n, int m) {
        for (const auto& vv: v) {
                int mm = m + 1;
                for (int e : vv) {
                   cout << e << " ";
                   mm--;
                   if (mm == 0) break;
                }
                n--;
```

```
                cout << endl;
                if (n<0) break;
        }
}

int main() {
        int k;
        cin >> n >> m >> k;

    std::vector<std::pair<int, int>> v;

        for (int i = 1; i <= k; ++i) {
            int s, f;
            cin >> s >> f;
            v.push_back(std::make_pair(s, -i));
            v.push_back(std::make_pair(f, i));
    }

    std::sort(v.begin(), v.end());

    int maxd = n + m + 3;

        for (int kk = 0; kk < 2*k; ++kk) {
                if (v[kk].second < 0) { // enter
                        int down = 0, up = maxd;
                        int pi;
                        int pj;
                        while (up - down > 1) {
                                int t = (up + down) / 2;
                                if (!checkNewInt(t, n, m, pi, pj))
                                        up = t;
                                else
```

```
                down = t;
            }
            ri = 100500, rj = 100500;
            // ищем самую левую верхнюю точку
            checkNewInt(down, n, m, ri, rj);
            if (ri >= 100500 || rj >= 100500) {
               cout << "-1 -1"  << endl;
        } else {
               cout << ri + 1 << " " << rj + 1 << "\n";
               for (int dd = 0; dd < maxd; ++dd) {
                     for (int i=0; i < n; ++i) {
                        int l = abs(i - ri);
                        int ddd = dd - l;
          if (ddd<0) continue;
                        int lrj = rj - ddd;
                        int rrj = rj + ddd + 1;
                        if (lrj < 0) {
                              lrj = 0;
                        }
                        if (rrj > m) {
                              rrj = m;
                        }

                        v3[dd][i][lrj]++;
                        v3[dd][i][rrj]--;
                     }
               }
            }
            wh[-v[kk].second] = ii(ri, rj);
      } else {
            // удаляем
            auto [ri, rj] = wh[v[kk].second];
```

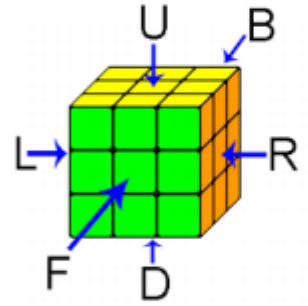
```
                        wh.erase(v[kk].second);
                        for (int dd = 0; dd < maxd; ++dd) {
                                for (int i=0; i < n; ++i) {
                                        int l = abs(i - ri);
                                        int ddd = dd - l;
                                        if (ddd < 0) continue;
                                        int lrj = rj - ddd;
                                        int rrj = rj + ddd + 1;
                                        if (lrj < 0) {
                                                lrj = 0;
                                        }
                                        if (rrj > m) {
                                                rrj = m;
                                        }

                                        v3[dd][i][lrj]--;
                                        v3[dd][i][rrj]++;
                                }
                        }
                }
/*              if (kk == 0) {
                        print(v3[3], n, m);
                        cout << "---\n";
                        print(v3[4], n, m);
                }*/
        }
        return 0;
}
```

Задание 4:

Задача 4. cube: Кубик

Рассмотрим кубик рубика размера 3x3x3. Кубик состоит из 26 кубиков-деталей, хотя бы одна грань которых находится на поверхности кубика 3x3x3. У угловых кубиков-деталей на поверхность кубика-рубика выходят 3 грани, у реберных кубиков-деталей на поверхность кубика-рубика выходят 2 грани, у центральных кубиков-деталей - 1 грань.
В центре фронтальной грани (на поверхности) кубика-рубика находится нанороботт. Нанороботт изначально смотрит в направлении вверх. Нанороботт может исполнять команду 'F', что означает перемещение на ⅓ размера кубика рубика в направлении своего движения, то есть за один шаг он попадает в центр смежного квадратика на поверхности кубика рубика. При необходимости нанороботт переходит на смежную грань кубика-рубика, всегда оставаясь на его поверхности.
Кроме того нанороботт может исполнять команды 'L' - повернуть на 90 градусов налево, 'R' - повернуть на 90 градусов вправо, и 'S' - остановиться. Команда 'S' гарантированно находится в конце последовательности команд.



Фронтальная грань обозначена "F".

Формат входных данных

На стандартный поток ввода подается последовательность команд без пробелов, заканчивающаяся командой 'S'.

Формат выходных данных

На стандартный поток вывода напечатайте количество кубиков-деталей (от 1 до 26), которые были посещены нанороботтом.

Баллы: 33.33333

Решение:

```
#include <iostream>
#include <vector>
#include <math.h>
#include <algorithm>
#include <set>
#include <map>
#include <string>
using namespace std;
#define ll long long
struct detail{
   int gr, kl;
```

```
};


int main()
{
    /*
    char col;
    string s;
    cin >> col >> s;
    long long ch = 0;
    for (int i = 0; i < s.size(); i++){
        if(s[i] == 'Z'){
            ch -= (ll) pow(3, s.size() - i - 1);
        }
        if(s[i] == 49){
            ch += (ll )pow(3, s.size() - i - 1);
        }
    }
    ch = abs(ch);
    int k = 0;
    while(pow(9, k) < ch){
        k++;
    }
    long long start = pow(9, k), razn = start - ch;
    cout << 729 - 2*81 + 2*27 + 2*1;
    k --;*/
```

```
char f;
string s;
cin >> s;
int k = 1;
bool cub[6][9];
for (int i = 0; i < 6; i++){
   for (int j = 0; j < 9; j++){
      cub[i][j] = false;
   }
}

detail coord;
string napr = "u";
coord.gr = 1;
coord.kl = 5;
cub[coord.gr - 1][coord.kl - 1] = true;
long long ans = 0;
for (int i = 0; i < s.size(); i++) {
   if (s[i] == 'L') {
      if (napr == "u") {
         napr = "l";
         continue;
      }
      if (napr == "l") {
         napr = "d";
         continue;
      }
      if (napr == "d") {
         napr = "r";
         continue;
      }
      if (napr == "r") {
```

```
            napr = "u";
            continue;
        }
    }
    if (s[i] == 'R') {
        if (napr == "u") {
            napr = "r";
            continue;
        }
        if (napr == "l") {
            napr = "u";
            continue;
        }
        if (napr == "d") {
            napr = "l";
            continue;
        }
        if (napr == "r") {
            napr = "d";
            continue;
        }
    }
    if (s[i] == 'F') {
        if (napr == "u") {
            coord.kl -= 3;
            //cout << coord.gr << " " << coord.kl << endl;
            if (coord.kl > 0) {
                cub[coord.gr - 1][coord.kl - 1] = true;
            }
            if (coord.kl < 1) {
                coord.kl += 9;
                if (coord.gr == 1 || coord.gr == 2 || coord.gr == 4) {
```

```
            coord.gr = 3;
            cub[coord.gr - 1][coord.kl - 1] = true;
            continue;
        }
        if (coord.gr == 3) {
            coord.gr = 6;
            cub[coord.gr - 1][coord.kl - 1] = true;
            continue;
        }
        if (coord.gr == 6) {
            coord.gr = 5;
            cub[coord.gr - 1][coord.kl - 1] = true;
            continue;
        }
        if (coord.gr == 5) {
            coord.gr = 1;
            cub[coord.gr - 1][coord.kl - 1] = true;
            continue;
        }
    }
}

if (napr == "d") {
    coord.kl += 3;
    if (coord.kl < 9) {
        cub[coord.gr - 1][coord.kl - 1] = true;
    }
    if (coord.kl > 9) {
        coord.kl -= 9;
        if (coord.gr == 1 || coord.gr == 2 || coord.gr == 4) {
            coord.gr = 5;
            cub[coord.gr - 1][coord.kl - 1] = true;
```

```
            continue;
        }
        if (coord.gr == 3) {
            coord.gr = 1;
            cub[coord.gr - 1][coord.kl - 1] = true;
            continue;
        }
        if (coord.gr == 6) {
            coord.gr = 3;
            cub[coord.gr - 1][coord.kl - 1] = true;
            continue;
        }
        if (coord.gr == 5) {
            coord.gr = 6;
            cub[coord.gr - 1][coord.kl - 1] = true;
            continue;
        }
    }
}

if (napr == "r") {
    coord.kl += 1;
    if ((coord.kl - 2) / 3 == (coord.kl - 1) / 3) {
        cub[coord.gr - 1][coord.kl - 1] = true;
    }
    if ((coord.kl - 2) / 3 != (coord.kl - 1) / 3) {
        coord.kl -= 3;
        if (coord.gr == 1 || coord.gr == 3 || coord.gr == 5) {
            coord.gr = 4;
            cub[coord.gr - 1][coord.kl - 1] = true;
            continue;
        }
```

```
        if (coord.gr == 4) {
            coord.gr = 6;
            cub[coord.gr - 1][coord.kl - 1] = true;
            continue;
        }
        if (coord.gr == 6) {
            coord.gr = 2;
            cub[coord.gr - 1][coord.kl - 1] = true;
            continue;
        }
        if (coord.gr == 2) {
            coord.gr = 1;
            cub[coord.gr - 1][coord.kl - 1] = true;
            continue;
        }
    }
}
if (napr == "l") {
    coord.kl -= 1;
    if ((coord.kl - 2) / 3 == (coord.kl - 1) / 3) {
        cub[coord.gr - 1][coord.kl - 1] = true;
    }
    if ((coord.kl - 2) / 3 != (coord.kl - 1) / 3) {
        coord.kl += 3;
        if (coord.gr == 1 || coord.gr == 3 || coord.gr == 5) {
            coord.gr = 2;
            cub[coord.gr - 1][coord.kl - 1] = true;
            continue;
        }
        if (coord.gr == 4) {
            coord.gr = 1;
            cub[coord.gr - 1][coord.kl - 1] = true;
```

```
                continue;
            }
            if (coord.gr == 6) {
                coord.gr = 4;
                cub[coord.gr - 1][coord.kl - 1] = true;
                continue;
            }
            if (coord.gr == 2) {
                coord.gr = 6;
                cub[coord.gr - 1][coord.kl - 1] = true;
                continue;
            }
        }
    }

  }
}

vector <int> right {1, 2, 3, 4, 5, 6, 7, 8, 9,
                    10, 11, 1, 12, 13, 4, 14, 15, 7,
                    10, 16, 17, 11, 18, 19, 1, 2, 3,
                    1, 19, 17, 6, 20, 21, 9, 22, 23,
                     7, 8, 9, 15, 24, 22, 14, 25, 23,
                    14, 25, 23, 12, 26, 21, 10, 16, 17};

for (int i = 0; i < 6; i++){
   for(int j = 0; j < 9; j++){
      if (cub[i][j] && right[i * 9 + j] != 0){
         ans++;
         int n = right[i * 9 + j];
         for (int l = 0; l < 54; l++){
            if(right[l] == n && n > 0){
```

```
                right[l] = 0;
                cub[(l - j) / 9][j] = false;
            }
        }

    }

  }
}

cout << ans;
set <pair<int, int>> check;
long long ans = 0;
for (int i = 0; i < 6; i++){
   for (int j = 0; j < 9; j++){
      pair <int, int> p;
      p.first = i;
      p.second = j;
      if(cub[i][j] == true && check.count(p) == 0){
         ans++;
         check.insert(p);
         if (i == 0 && i == 1 && i == 3) {
            if (j % 3 == 0) {
               p.first = 2;
               p.second = j + 6;
               check.insert(p);
            }
         }
         if (i == 2) {
            if (j % 3 == 0) {
               p.first = 5;
               p.second = j + 6;
```

```
                check.insert(p);
            }
        }
        if (i == 5) {
            if (j % 3 == 0) {
                p.first = 4;
                p.second = j + 6;
                check.insert(p);
            }
        }
        if (i == 4) {
            if (j % 3 == 0) {
                p.first = 0;
                p.second = j + 6;
                check.insert(p);
            }
        }

      }
    }
  }
  cout << coord.gr << " " << coord.kl << " "<< napr << endl;
}

*/
   /* string s;
    cin >> s;
    vector <char> ss(0);
    for (int i = 0; i < s.size(); i++){
       ss.push_back(s[i]);
    }
    sort(ss.begin(), ss.end());
```

```
vector <pair<char, char>> a(s.size());
for (int i = 0; i < s.size(); i++){
   a[i].first = s[i];
   a[i].second = ss[i];

}
sort(a.begin(), a.end());
map <char, int> has, was;
for (int i = 0; i < s.size(); i++){
   has[s[i]]++;
}
string ans = "#";
ans += a[0].second;
was[a[0].second]++;
was[a[0].first]++;
for (int i = 0; i < s.size(); i++){
   for (int j = 0; j < s.size(); j++){
      if(a[j].first == ans[ans.size() - 1] && was[a[j].second] < has[a[j].second]){
         was[a[j].second]++;
         ans += a[j].second;
      }
   }
}
ans.erase(ans.begin());
cout << ans;
```

}

Задание 5:

Задача 5. poker79: Покер

В одном элитном апарт-отеле было решено провести турнир по спортивному покеру на игровых автоматах. Для проведения турнира на все игровые автоматы загружается одинаковая последовательность конфигураций, и игроки соревнуются в том, кто выиграет больше партий. Игровые конфигурации обозначаются символами цифр '0'-'9' и строчными латинскими буквами 'a'-'z'. Специальная конфигурация, обозначаемая '#', вызывает перезагрузку автомата. Конфигурация '#' встречается ровно один раз в конце последовательности игровых конфигураций.
Игровой автомат исполняет последовательность конфигураций циклически, то есть дойдя до конфигурации '#' начинает исполнение с начала последовательности конфигураций.
Для загрузки конфигураций в автомат они кодируются следующим образом. Пусть длина последовательности игровых конфигураций равна N. Тогда, добавляя к ним конфигурацию '#', получим последовательность длины N + 1. Возьмем N + 1 начальных позиций в последовательности (от первой и до N + 1) и для каждой начальной позиции возьмем N + 1 следующих за ней конфигураций. Получим N + 1 последовательностей каждую длины N + 1.
Например, если начальная последовательность из 4 конфигураций равна:
1231

добавляя к ней '#' и беря все возможные начальные позиции, получим 5 последовательностей:
1231#
231#1
31#12
1#123
#1231

отсортируем эти последовательности лексикографически:
#1231
1#123
1231#
231#1
31#12

Мы получили матрицу из N + 1 строк и N + 1 столбцов. Теперь возьмем в этой матрице последний столбец: 13#12. Это будет закодированная последовательность конфигураций.
Ваша задача — написать программу, которая раскодирует закодированную последовательность конфигураций.

Формат входных данных

На стандартном потоке ввода задается одна строка длины не более 250 символов — закодированная последовательность конфигураций. Гарантируется, что конфигурация '#' встречается в последовательности ровно один раз.

Формат выходных данных

На стандартный поток вывода напечатайте декодированную последовательность конфигураций в виде строки. Конфигурацию '#' не печатайте.

Баллы: 33.33333

Решение:

```
#define _CRT_SECURE_NO_WARNINGS
#include <iostream>
#include <stdio.h>
#include <stdlib.h>
#include <algorithm>
#include <math.h>
#include <string.h>
#include <string>
#include <vector>
#include <stack>
#include <queue>
#include <set>
#include <map>
#include <unordered_map>
#include <chrono>
#include <time.h>
#include <random>
using namespace std;

#define int long long
typedef long long ll;
typedef unsigned long long ull;
typedef pair<int, int> pii;
typedef pair<ll, ll> pll;
typedef double db;
const int INF = 1000000007;
const ll LLINF = 1000000000000000007LL;
const double EPS = 1e-9;



int a[100001];
```

```
pair<int, int> as[100001];
int res[100001];
int t[100001];



signed main() {
   ios_base::sync_with_stdio(0); cin.tie(0); cout.tie(0);
   int n;
   cin >> n;
   int x = -1;

   for (int i = 0; i < n + 1; ++i) {
      cin >> a[i];
      as[i] = { a[i], i };
      if (a[i] == 0) {
         x = i;
      }
   }
   sort(as, as + n + 1);
   for (int i = 0; i <= n; ++i) {
      t[i] = as[i].second;
   }

   int j = t[x];
   for (int i = 0; i <= n; ++i) {
      res[i] = a[j];
      j = t[j];
   }
   bool fl = 0;
   for (int i = 0; i < n; ++i) {
      cout << res[i] << ' ';
   }
```

```
    cout << "\n";
    return 0;
}
```

Задание 6:

Задача 6. update: Обновление

Для обновления программного обеспечения в одной p2p сети на центральный узел с номером 0 был загружен апдейт состоящий из K частей по 1 мегабайту. В сети N узлов, соединенные каждый с каждым так, что между парой узлов за одну секунду в каждом направлении может передаваться одна часть размера 1 мегабайт. Каждый узел может одновременно принимать и передавать данные. Когда какая-то часть обновления загружается полностью в узел, эта часть постоянно сохраняется в памяти узла. Каждый узел включается и начинает прием/передачу обновления в момент времени t_i от момента загрузки на центральный узел. Вам требуется определить момент времени, когда все части обновления будут загружены на все узлы сети.

Формат входных данных

В первой строке вводится N и K - число узлов в сети, включая центральный и число частей обновления ($1 \le N \le 10^5$. $1 \le K \le 10^6$). В следующих строках находятся N чисел - времена включения узлов, начиная с нулевого ($0 \le t_i \le 10^6$, $t_0 = 0$)

Формат выходных данных

Выведите одно число - минимальный момент времени, когда на всех узлах сети будут все части обновления

Баллы: 33.33333

Решение:

```
import java.io.FileInputStream;
import java.util.Arrays;
import java.util.Scanner;


public class update_sol {
        static int n, k;
        static int time_max; // final moment
   static int last_batch; // number of peers that come online at the final moment
        static int solution;


        private void read() {
                Scanner s = new Scanner(System.in);
                n = s.nextInt() - 1; // number of peers
                k = s.nextInt(); // number of pieces
                time_max = -1;
```

```
            int tmp = s.nextInt();
                for (int i=0; i<n; i++) {
                   int time = s.nextInt();
                   if (time > time_max) {
                      time_max = time;
                      last_batch = 1;
                   } else if (time == time_max) {
                      last_batch++;
                   }
                }
      }

      private void solve() {
         int prev_swarm = n-last_batch;
         int first_step = prev_swarm+1;
         int next_steps = n;
solution = time_max + 1;
         if (first_step < k) {
            int remainder = (k-first_step);
            solution += remainder / next_steps;
            if (remainder % next_steps > 0) {
               solution++;
            }
         }
      }

      private void write() {
         System.out.println("" + solution);
      }

      private void run() {
   read();
```

```
        solve();
        write();
        }

        public static void main(String[] args) {
            (new update_sol()).run();
        }
}
```