Максимальная сумма баллов: 100

Задание 1

Задача 1. count1011: Количество вхождений цифры

В задаче рассмотрим две симметричные системы счисления (далее будем писать сокращённо - ССС): девятиричную ССС и двадцатисемиричную ССС. 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 двадцатисемиричной ССС используются цифры N, O, P, Q, R, S, T, U, V, W, X, Y, Z, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D. К записи числа в этой системе приписывают снизу окончание $_{S27}$. $N_{S27}=-13_{10}$, $O_{S27}=-12_{10}$, $P_{S27}=-11_{10}$, $Q_{S27}=-10_{10}$, $R_{S27}=-9_{10}$, $S_{S27}=-8_{10}$, $T_{S27}=-7_{10}$, $U_{S27}=-6_{10}$, $V_{S27}=-5_{10}$, $W_{S27}=-4_{10}$, $X_{S27}=-3_{10}$, $Y_{S27}=-2_{10}$, $Z_{S27}=-1_{10}$, $0_{S27}=0_{10}$, $1_{S27}=1_{10}$, $2_{S27}=2_{10}$, $3_{S27}=3_{10}$, $4_{S27}=4_{10}$, $5_{S27}=5_{10}$, $6_{S27}=6_{10}$, $7_{S27}=7_{10}$, $8_{S27}=8_{10}$, $9_{S27}=9_{10}$, $A_{S27}=10_{10}$, $B_{S27}=11_{10}$, $C_{S27}=12_{10}$, $D_{S27}=13_{10}$. Если имеется запись в двадцатисемиричной ССС $a_n a_{n-1}...a_1 a_{0S27}$, то она обозначает число, равное $a_n*27^n+a_{n-1}*27^{n-1}+...+a_1*27+a_0*1$. Например, $623_{10}=729-108+2=1*27^2+(-4)*27+2*1=1W2_{S27}$. 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 двадцатисемиричной ССС. В этой записи используются десятичные цифры и заглавные латинские буквы {N, ..., Z, 0, 1, ..., 9, A, ..., D}. Длина записи числа K в двадцатисемиричной 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 двадцатисемиричной ССС целого числа K (в этой записи не более чем 100000 символов). В записи числа K используются десятичные цифры и заглавные латинские буквы {N, ..., Z, 0, 1, ..., 9, A, ..., D}.

Формат выходных данных

В первой и единственной строке выводится неотрицательное целое число (от 0 до 150000), равное искомому количеству вхождений цифры J в запись числа K в девятеричной ССС.

Баллы: 20

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. vampir1011: 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 различных положительных чисел F_i, разделённых пробелами. Число N не более чем 800. Числа F_i не более чем 14300. Программа подсчитывает количество различных чисел вампиров$_{S9}$, которые можно получить, используя в качестве их клыков числа F_i, и выводит результат подсчёта.

Формат входных данных

В первой строке содержится целое положительное число N — длина последовательности чисел, ($0 < N < 801$).
Во второй строке содержится N различных целых положительных чисел F_i, разделённых пробелами ($0 < F_i < 14301$).

Формат выходных данных

В первой и единственной строке выводится неотрицательное целое число (не меньше чем 0, не больше чем 32000), равное искомому количеству различных чисел вампиров$_{S9}$, которые можно получить, используя в качестве их клыков числа F_i.

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: 20

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. sphere: Разведка

Корпорация СорКосмос для популяризации своей деятельности заказала анимационный фильм о межпланетной космической программе. В нем автоматический зонд "Великий попил" совершил посадку на поверхности Венеры. На борту зонда находится автоматический квадрокоптер "Полный улет", который будет летать в атмосфере и изучать поверхность планеты.

Квадрокоптер летает по координатной сетке с шагом в одну угловую минуту, так, что его координаты можно представить парой целых чисел (Lat, Lon), где Lat — географическая широта точки, измеряемая в угловых минутах, то есть значение в диапазоне [-5400, 5400], где значению -5400 соответствует южный полюс, а значению 5400 — северный полюс. Lon — географическая долгота точки, измеряемая в угловых минутах, то есть значение в диапазоне (-10800, 10800], где отрицательными числами обозначается западная долгота, а положительными — восточная. Таким образом параллели и меридианы разделяют поверхность планеты на клетки, сторона каждой клетки равна одной угловой минуте.

Для северного и южного полюса географическая долгота может быть произвольным целым числом в диапазоне (-10800, 10800].

Находясь в точке с координатами (Lat, Lon) не на полюсах квадрокоптер изучает поверхность, заключенную в область на сфере, ограниченную дугами координатной сетки между точками (Lat - 1, Lon + 1), (Lat - 1, Lon - 1), (Lat + 1, Lon - 1), (Lat + 1, Lon + 1), то есть покрывает координатный квадрат со стороной в две угловые минуты с центром в точке (Lat, Lon). Мы будем считать, что это — 4 клетки координатной сетки.

Находясь на северном полюсе квадрокоптер изучает область севернее широты 5399, а находясь на южном полюсе квадрокоптер изучает всю область южнее широты -5399.

Находясь на широтах 5399 и -5399 квадрокоптер изучает и все клетки севернее 5399 и южнее -5399 соответственно.

Программа для квадрокоптера представляет собой последовательность команд 'F' — перелететь на следующую точку координатной сетки, 'L' — повернуть налево, 'R' — повернуть направо, 'S' — остановиться. Команда 'S' — единственная и последняя в последовательности команд. Команда 'L' означает, что если квадрокоптер был ориентирован на север, то после выполнения команды он будет ориентирован на запад, потом на юг, потом на восток. Перед командой 'F' может задаваться число повторений этой команды, например, 20 F означает, что квадрокоптер перемещается вперед на 20 угловых минут, исследуя всю область, которую он пролетел. В начальный момент времени квадрокоптер ориентирован на север. Гарантируется, что

в начальный и конечный момент времени квадрокоптер не находится на полюсах, и при пролете полюса квадрокоптер не поворачивает. То есть, при пролете северного полюса квадрокоптер без изменения направления движения меняет северную ориентацию на южную. Пролет северного и южного полюсов в последовательности команд всегда соответствует двум последовательным командам 'F'.

Формат входных данных

На стандартном потоке ввода задаются два целых числа — координаты точки, в которой квадрокоптер начинает свой полет, затем вещественное число — радиус планеты в километрах, затем последовательность команд квадрокоптера, завершающаяся командой S.

Команды и число повторений могут отделяться друг от друга пробельными символами (в том числе переводом строк). Число повторений записывается без пробелов.

Число повторений не превосходит 22000. Общая длина перемещения квадрокоптера не превосходит 1000000 клеток.

Формат выходных данных

На стандартный поток вывода напечатайте количество клеток координатной сетки, которые были исследованы квадрокоптером. Вся область севернее широты 5399 учитывается как одна клетка вся область южнее широты -5399 учитывается как одна клетка. Далее выведите вещественное число — площадь исследованной поверхности планеты в километрах. Пролет над клеткой учитывается только один раз, то есть повторный пролет над нею не увеличивает количество исследованных клеток и исследованную площадь.

Площадь исследованной поверхности планеты выводите не менее чем с 10 десятичными знаками (как показано в примере).

Программа, которая правильно вычисляет только количество клеток, но выводит неправильную площадь (например, всегда выводит площадь 0) получит как минимум 50 баллов из 100.

Баллы: 20

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. poker1011: 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 положительными целыми числами от 1 до 31918903, а конфигурация 0 является специальной конфигурацией, вызывающей перезагрузку автомата. Конфигурация 0 встречается ровно один раз в конце последовательности игровых конфигураций.
Игровой автомат исполняет последовательность конфигураций циклически, то есть дойдя до конфигурации 0 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 0, 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 конфигураций равна:
1 2 3 1

добавляя к ней 0 и беря все возможные начальные позиции, получим 5 последовательностей:
1 2 3 1 0
2 3 1 0 1
3 1 0 1 2
1 0 1 2 3
0 1 2 3 1

отсортируем эти последовательности лексикографически:
0 1 2 3 1
1 0 1 2 3
1 2 3 1 0
2 3 1 0 1
3 1 0 1 2

Мы получили матрицу из N + 1 строк и N + 1 столбцов. Теперь возьмем в этой матрице последний столбец: 1 3 0 1 2. Это будет закодированная последовательность конфигураций.
Ваша задача — написать программу, которая раскодирует закодированную последовательность конфигураций.

Формат входных данных

На стандартном потоке ввода сначала задается число N (0 < N < 100000), задающее длину последовательности, за которой следует N + 1 целое число — номера конфигураций. Гарантируется, что конфигурация 0 встречается в последовательности ровно один раз.

Формат выходных данных

На стандартный поток вывода напечатайте N чисел — декодированную последовательность конфигураций. Конфигурацию 0 не печатайте.

Баллы: 20

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: 20

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();
        }
}
```

Задание 7:

Задача 7. blackbox: Черный ящик

Вам на исследование дается некоторая программа без исходного кода. Программа принимает на вход последовательность целых и дробных числа со знаком в десятичной записи. Например, 392, -1.28281 и т. п. Ввод завершается признаком конца файла (в Unix - Ctrl-D). Программа выводит результат на стандартный поток вывода. Для некорректных входных данных программа выводит строку invalid.
Напишите программу, которая воспроизводит функциональность данной программы, то есть для корректных входных данных будет давать тот же результат, что и исходная программа. Можете предполагать, что на вход вашей программы не будут подаваться входные данные, для которых исходная программа выводит invalid.
Вы можете использовать jslinux, на который загрузить программу для эмулятора.
После перехода по ссылке в окне браузера загрузится операционная система Linux и вы получите в окне браузера такой скрин:

Кнопка загрузки файла в операционную систему находится в левом нижнем углу.
Загрузите туда этот файл. После загрузки наберите в командной строке команду "ls -l".
Вы должны получить следующее:

Баллы: 20

Решение:

Elf32_Sym struc ; (sizeof=0x10, align=0x4, mappedto_1)

```
                ; XREF: LOAD:000001B0/r
                ; LOAD:000001C0/r ...
st_name dd ?          ; offset (00000280)
st_value dd ?         ; offset (00000000)
st_size dd ?
st_info db ?
st_other db ?
st_shndx dw ?
Elf32_Sym ends


Elf32_Rel struc ; (sizeof=0x8, align=0x4, copyof_2)
                ; XREF: LOAD:00000360/r
                ; LOAD:00000368/r ...
r_offset dd ?
r_info dd ?
Elf32_Rel ends


Elf32_Dyn struc ; (sizeof=0x8, align=0x4, copyof_3)
                ; XREF: LOAD:_DYNAMIC/r
                ; LOAD:00003F08/r ...
d_tag dd ?
d_un Elf32_Dyn::$A263394DDF3EC2D4B1B8448EDD30E249 ?
Elf32_Dyn ends


Elf32_Dyn::$A263394DDF3EC2D4B1B8448EDD30E249 union ; (sizeof=0x4, align=0x4, copyof_4)
                ; XREF: Elf32_Dyn/r
d_val dd ?
d_ptr dd ?
Elf32_Dyn::$A263394DDF3EC2D4B1B8448EDD30E249 ends
```

```
.686p
.mmx
.model flat
.intel_syntax noprefix


; Segment type: Pure data
; Segment permissions: Read
LOAD segment mempage public 'DATA' use32
assume cs:LOAD
dword_0 dd 464C457Fh    ; File format: \x7FELF
db 1                    ; File class: 32-bit
db 1                    ; Data encoding: little-endian
db 1                    ; File version
db 0                    ; OS/ABI: UNIX System V ABI
db 0                    ; ABI Version
db 7 dup(0)             ; Padding
dw 3                    ; File type: Shared object
dw 3                    ; Machine: Intel 386
dd 1                    ; File version
dd offset _start        ; Entry point
dd 34h                  ; PHT file offset
dd 3780h                ; SHT file offset
dd 0                    ; Processor-specific flags
dw 34h                  ; ELF header size
dw 20h                  ; PHT entry size
dw 0Ah                  ; Number of entries in PHT
dw 28h                  ; SHT entry size
word_30 dw 19h          ; Number of entries in SHT
```

```
dw 18h              ; SHT entry index for string table
; ELF32 Program Header
; PHT Entry 0
dword_34 dd 6       ; Type: PHDR
dd 34h              ; File offset
dd offset dword_34  ; Virtual address
dd 34h              ; Physical address
dd 140h             ; Size in file image
dd 140h             ; Size in memory image
dd 4                ; Flags
dd 4                ; Alignment
; PHT Entry 1
dword_54 dd 3       ; Type: INTERP
dd 174h             ; File offset
dd offset aLibLdMuslI386S ; Virtual address
dd 174h             ; Physical address
dd 17h              ; Size in file image
dd 17h              ; Size in memory image
dd 4                ; Flags
dd 1                ; Alignment
; PHT Entry 2
dd 1                ; Type: LOAD
dd 0                ; File offset
dd 0                ; Virtual address
dd 0                ; Physical address
dd 3D0h             ; Size in file image
dd 3D0h             ; Size in memory image
dd 4                ; Flags
dd 1000h            ; Alignment
; PHT Entry 3
dd 1                ; Type: LOAD
dd 1000h            ; File offset
```

```
dd offset _init_proc    ; Virtual address
dd 1000h                ; Physical address
dd 931h                 ; Size in file image
dd 931h                 ; Size in memory image
dd 5                    ; Flags
dd 1000h                ; Alignment
; PHT Entry 4
dd 1                    ; Type: LOAD
dd 2000h                ; File offset
dd offset aInvalid      ; Virtual address
dd 2000h                ; Physical address
dd 16Ch                 ; Size in file image
dd 16Ch                 ; Size in memory image
dd 4                    ; Flags
dd 1000h                ; Alignment
; PHT Entry 5
dd 1                    ; Type: LOAD
dd 2EF0h                ; File offset
dd offset __CTOR_LIST__ ; Virtual address
dd 3EF0h                ; Physical address
dd 114h                 ; Size in file image
dd 134h                 ; Size in memory image
dd 6                    ; Flags
dd 1000h                ; Alignment
; PHT Entry 6
dd 2                    ; Type: DYNAMIC
dd 2F00h                ; File offset
dd offset _DYNAMIC      ; Virtual address
dd 3F00h                ; Physical address
dd 0C0h                 ; Size in file image
dd 0C0h                 ; Size in memory image
dd 6                    ; Flags
```

```
dd 4                ; Alignment
; PHT Entry 7
dd 6474E550h        ; Type: EH_FRAME
dd 2010h            ; File offset
dd offset __GNU_EH_FRAME_HDR ; Virtual address
dd 2010h            ; Physical address
dd 44h              ; Size in file image
dd 44h              ; Size in memory image
dd 4                ; Flags
dd 4                ; Alignment
; PHT Entry 8
dd 6474E551h        ; Type: STACK
dd 0                ; File offset
dd 0                ; Virtual address
dd 0                ; Physical address
dd 0                ; Size in file image
dd 0                ; Size in memory image
dd 6                ; Flags
dd 10h              ; Alignment
; PHT Entry 9
dd 6474E552h        ; Type: RO-AFTER
dd 2EF0h            ; File offset
dd offset __CTOR_LIST__ ; Virtual address
dd 3EF0h            ; Physical address
dd 110h             ; Size in file image
dd 110h             ; Size in memory image
dd 4                ; Flags
dd 1                ; Alignment
aLibLdMuslI386S db '/lib/ld-musl-i386.so.1',0
align 4
; ELF GNU Hash Table
elf_gnu_hash_nbuckets dd 2
```

```
elf_gnu_hash_symbias dd 0Bh
elf_gnu_hash_bitmask_nwords dd 1
elf_gnu_hash_shift dd 5
elf_gnu_hash_indexes dd 81002400h
elf_gnu_hash_bucket dd 0Bh, 0
elf_gnu_hash_chain dd 0EF18DB8h, 0EEFD3EBh
; ELF Symbol Table
Elf32_Sym <0>
Elf32_Sym <offset aPrintf - offset unk_280,\ ; "printf"
        offset dword_0, 0, 12h, 0, 0>
Elf32_Sym <offset aPuts - offset unk_280,\ ; "puts"
        offset dword_0, 0, 12h, 0, 0>
Elf32_Sym <offset aCxaFinalize - offset unk_280,\ ; "__cxa_finalize"
        offset dword_0, 0, 22h, 0, 0>
Elf32_Sym <offset aStackChkFail - offset unk_280,\ ; "__stack_chk_fail"
        offset dword_0, 0, 12h, 0, 0>
Elf32_Sym <offset aRegisterFrameI - offset unk_280,\ ; "__register_frame_info_bases"
        offset dword_0, 0, 20h, 0, 0>
Elf32_Sym <offset aItmRegistertmc - offset unk_280,\ ; "_ITM_registerTMCloneTable"
        offset dword_0, 0, 20h, 0, 0>
Elf32_Sym <offset aDeregisterFram - offset unk_280,\ ; "__deregister_frame_info_bases"
        offset dword_0, 0, 20h, 0, 0>
Elf32_Sym <offset aItmDeregistert - offset unk_280,\ ; "_ITM_deregisterTMCloneTable"
        offset dword_0, 0, 20h, 0, 0>
Elf32_Sym <offset aGetcharUnlocke - offset unk_280,\ ; "getchar_unlocked"
        offset dword_0, 0, 12h, 0, 0>
Elf32_Sym <offset aLibcStartMain - offset unk_280,\ ; "__libc_start_main"
        offset dword_0, 0, 12h, 0, 0>
Elf32_Sym <offset aInit - offset unk_280,\ ; "_init"
        offset _init_proc, 1, 12h, 0,\
        7>
Elf32_Sym <offset aFini - offset unk_280,\ ; "_fini"
```

```
        offset _term_proc, 1, 12h, 0,\
        0Bh>
; ELF String Table
unk_280 db    0
aLibcMuslX86So1 db 'libc.musl-x86.so.1',0
aPuts db 'puts',0
aStackChkFail db '__stack_chk_fail',0
aInit db '_init',0
aPrintf db 'printf',0
aFini db '_fini',0
aCxaFinalize db '__cxa_finalize',0
aGetcharUnlocke db 'getchar_unlocked',0
aLibcStartMain db '__libc_start_main',0
aRegisterFrameI db '__register_frame_info_bases',0
aItmRegistertmc db '_ITM_registerTMCloneTable',0
aDeregisterFram db '__deregister_frame_info_bases',0
aItmDeregistert db '_ITM_deregisterTMCloneTable',0
align 10h
; ELF REL Relocation Table
Elf32_Rel <3FE4h, 8>    ; R_386_RELATIVE
Elf32_Rel <3FF8h, 8>    ; R_386_RELATIVE
Elf32_Rel <3FFCh, 8>    ; R_386_RELATIVE
Elf32_Rel <4000h, 8>    ; R_386_RELATIVE
Elf32_Rel <3FE0h, 306h> ; R_386_GLOB_DAT __cxa_finalize
Elf32_Rel <3FE8h, 506h> ; R_386_GLOB_DAT __register_frame_info_bases
Elf32_Rel <3FECh, 606h> ; R_386_GLOB_DAT _ITM_registerTMCloneTable
Elf32_Rel <3FF0h, 706h> ; R_386_GLOB_DAT __deregister_frame_info_bases
Elf32_Rel <3FF4h, 806h> ; R_386_GLOB_DAT _ITM_deregisterTMCloneTable
; ELF JMPREL Relocation Table
Elf32_Rel <3FCCh, 107h> ; R_386_JMP_SLOT printf
Elf32_Rel <3FD0h, 207h> ; R_386_JMP_SLOT puts
Elf32_Rel <3FD4h, 407h> ; R_386_JMP_SLOT __stack_chk_fail
```

```
Elf32_Rel <3FD8h, 907h> ; R_386_JMP_SLOT getchar_unlocked
Elf32_Rel <3FDCh, 0A07h> ; R_386_JMP_SLOT __libc_start_main
LOAD ends


; Segment type: Pure code
; Segment permissions: Read/Execute
_init segment byte public 'CODE' use32
assume cs:_init
;org 1000h
assume es:nothing, ss:nothing, ds:_data, fs:nothing, gs:nothing



public _init_proc
_init_proc proc near
sub     esp, 0Ch        ; _init
call    frame_dummy
call    __do_global_ctors_aux
add     esp, 0Ch
retn
_init_proc endp

_init ends


; Segment type: Pure code
; Segment permissions: Read/Execute
LOAD segment mempage public 'CODE' use32
assume cs:LOAD
;org 1011h
assume es:nothing, ss:nothing, ds:_data, fs:nothing, gs:nothing
```

```
align 10h
LOAD ends



; Segment type: Pure code
; Segment permissions: Read/Execute
_plt segment para public 'CODE' use32
assume cs:_plt
;org 1020h
assume es:nothing, ss:nothing, ds:_data, fs:nothing, gs:nothing



sub_1020 proc near
push    dword ptr [ebx+4]
jmp     dword ptr [ebx+8]
sub_1020 endp

align 10h
; [00000006 BYTES: COLLAPSED FUNCTION _printf. PRESS CTRL-NUMPAD+ TO EXPAND]
push    0
jmp     sub_1020
; [00000006 BYTES: COLLAPSED FUNCTION _puts. PRESS CTRL-NUMPAD+ TO EXPAND]
push    8
jmp     sub_1020
; [00000006 BYTES: COLLAPSED FUNCTION ___stack_chk_fail. PRESS CTRL-NUMPAD+ TO EXPAND]
push    10h
jmp     sub_1020
; [00000006 BYTES: COLLAPSED FUNCTION _getchar_unlocked. PRESS CTRL-NUMPAD+ TO EXPAND]
push    18h
jmp     sub_1020
; [00000006 BYTES: COLLAPSED FUNCTION ___libc_start_main. PRESS CTRL-NUMPAD+ TO EXPAND]
```

```
push    20h ; ' '
jmp     sub_1020
_plt ends


; Segment type: Pure code
; Segment permissions: Read/Execute
; Segment alignment 'qword' can not be represented in assembly
_plt_got segment para public 'CODE' use32
assume cs:_plt_got
;org 1080h
assume es:nothing, ss:nothing, ds:_data, fs:nothing, gs:nothing
; [00000006 BYTES: COLLAPSED FUNCTION ___cxa_finalize. PRESS CTRL-NUMPAD+ TO EXPAND]
align 4



___register_frame_info_bases proc near
jmp     ds:(_term_proc_ptr+44h)[ebx] ; PIC mode
___register_frame_info_bases endp

align 10h



___deregister_frame_info_bases proc near
jmp     ds:(_term_proc_ptr+4Ch)[ebx] ; PIC mode
___deregister_frame_info_bases endp

align 4
_plt_got ends
```

```
; Segment type: Pure code
; Segment permissions: Read/Execute
LOAD segment mempage public 'CODE' use32
assume cs:LOAD
;org 1098h
assume es:nothing, ss:nothing, ds:_data, fs:nothing, gs:nothing
align 10h
LOAD ends


; Segment type: Pure code
; Segment permissions: Read/Execute
_text segment para public 'CODE' use32
assume cs:_text
;org 10A0h
assume es:nothing, ss:nothing, ds:_data, fs:nothing, gs:nothing


; Attributes: bp-based frame fuzzy-sp

; int __cdecl main(int argc, const char **argv, const char **envp)
public main
main proc near

var_78= dword ptr -78h
var_74= dword ptr -74h
var_70= dword ptr -70h
var_6C= dword ptr -6Ch
var_68= dword ptr -68h
var_64= dword ptr -64h
var_5C= dword ptr -5Ch
```

```
var_58= dword ptr -58h
var_54= dword ptr -54h
var_50= dword ptr -50h
var_4C= dword ptr -4Ch
var_3D= byte ptr -3Dh
var_3C= dword ptr -3Ch
var_38= dword ptr -38h
var_34= dword ptr -34h
var_30= dword ptr -30h
var_2C= byte ptr -2Ch
var_2B= byte ptr -2Bh
var_1C= dword ptr -1Ch
argc= dword ptr  8
argv= dword ptr  0Ch
envp= dword ptr  10h

lea     ecx, [esp+4]
and     esp, 0FFFFFFF0h
push    dword ptr [ecx-4]
push    ebp
mov     ebp, esp
push    edi
call    __x86_get_pc_thunk_di
add     edi, 2F0Dh
push    esi
push    ebx
push    ecx
sub     esp, 68h
mov     eax, large gs:14h
mov     [ebp+var_1C], eax
xor     eax, eax
mov     ebx, edi
```

```
call    _getchar_unlocked
mov     esi, eax
cmp     eax, 0FFFFFFFFh
jz      loc_122E
lea     eax, [esi-9]
cmp     eax, 4
jbe     loc_120F
lea     esi, [esi+0]
nop

loc_10F0:
cmp     esi, 20h ; ' '
jz      loc_120F

loc_10F9:
cmp     esi, 2Bh ; '+'
jz      loc_13B0
cmp     esi, 2Dh ; '-'
jz      loc_1310
lea     edx, [ebp+var_3C]
mov     [ebp+var_3C], 30303030h
mov     [ebp+var_5C], edx
mov     [ebp+var_38], 30303030h
mov     [ebp+var_34], 30303030h
mov     [ebp+var_30], 30303030h
mov     [ebp+var_2C], 30h ; '0'
mov     [ebp+var_2B], 0
mov     [ebp+var_58], 0
mov     [ebp+var_54], 0
cmp     esi, 30h ; '0'
jnz     short loc_116A
lea     esi, [esi+0]
```

```
nop

loc_1150:
mov     ebx, edi
call    _getchar_unlocked
cmp     eax, 30h ; '0'
jz      short loc_1150
mov     esi, eax
cmp     eax, 0FFFFFFFFh
jz      loc_13D8

loc_1167:
lea     eax, [esi-9]

loc_116A:
cmp     eax, 4
jbe     loc_1398
cmp     esi, 20h ; ' '
jz      loc_1398
cmp     esi, 2Eh ; '.'
jz      loc_1448
lea     ecx, [ebp+var_3D]
lea     eax, [esi-30h]
xor     edx, edx
mov     [ebp+var_4C], ecx
cmp     eax, 9
ja      short loc_11F0
mov     eax, esi
mov     esi, edx
jmp     short loc_11A9
align 10h
```

```
loc_11A0:
cmp     esi, 11h
jz      loc_1250

loc_11A9:
mov     edx, [ebp+var_4C]
mov     [ebp+var_50], esi
add     esi, 1
mov     ebx, edi
mov     [edx+esi], al
call    _getchar_unlocked
lea     ecx, [eax-30h]
cmp     ecx, 9
jbe     short loc_11A0
mov     edx, esi
mov     esi, eax
cmp     eax, 2Eh ; '.'
jz      loc_13F3

loc_11D1:
cmp     esi, 0FFFFFFFFh
jz      loc_1280
lea     eax, [esi-9]
cmp     eax, 4
jbe     loc_1280
cmp     esi, 20h ; ' '
jz      loc_1280
nop

loc_11F0:
sub     esp, 0Ch
push    esi
```

```
call    skip_invalid
add     esp, 10h
mov     esi, eax

loc_11FE:
cmp     esi, 0FFFFFFFFh
jz      short loc_122E
lea     eax, [esi-9]
cmp     eax, 4
ja      loc_10F0

loc_120F:
mov     ebx, edi
call    _getchar_unlocked
mov     esi, eax
lea     eax, [eax-9]
cmp     eax, 4
jbe     short loc_120F
cmp     esi, 20h ; ' '
jz      short loc_120F
cmp     esi, 0FFFFFFFFh
jnz     loc_10F9

loc_122E:
mov     eax, [ebp+var_1C]
xor     eax, large gs:14h
jnz     loc_16BA
lea     esp, [ebp-10h]
xor     eax, eax
pop     ecx
pop     ebx
pop     esi
```

```
pop     edi
pop     ebp
lea     esp, [ecx-4]
retn
align 10h

loc_1250:
mov     ebx, edi
mov     [ebp+var_4C], esi
add     esi, 1
call    _getchar_unlocked
lea     ecx, [eax-30h]
cmp     ecx, 9
ja      loc_15A4
cmp     esi, 2710h
jnz     short loc_1250
mov     esi, eax
jmp     loc_11F0
align 10h

loc_1280:
mov     ecx, [ebp+var_50]
movsx   ebx, byte ptr [ebp+var_3C]
add     ecx, 1Fh
shl     ecx, 19h
mov     [ebp+var_50], ecx
test    bl, bl
jz      loc_164B
lea     ecx, [ebp+var_3C+1]
mov     [ebp+var_5C], esi
xor     eax, eax
xor     edx, edx
```

```
mov     esi, ecx
lea     esi, [esi+0]

loc_12A8:
imul    ecx, edx, 0Ah
mov     [ebp+var_4C], ecx
mov     ecx, 0Ah
mul     ecx
mov     ecx, ebx
add     edx, [ebp+var_4C]
sar     ebx, 1Fh
add     eax, ecx
adc     edx, ebx
add     eax, 0FFFFFFD0h
adc     edx, 0FFFFFFFFh
movsx   ebx, byte ptr [esi]
add     esi, 1
test    bl, bl
jnz     short loc_12A8
mov     esi, [ebp+var_5C]

loc_12D4:
sub     esp, 4
mov     ecx, [ebp+var_50]
or      ecx, [ebp+var_54]
add     eax, 903F0000h
adc     edx, 0FFDC790Dh
mov     [ebp+var_68], eax
or      ecx, edx
mov     [ebp+var_64], ecx
push    [ebp+var_64]
push    [ebp+var_68]
```

```
loc_12F6:
lea     eax, (aLlx - 3FC0h)[edi] ; "%llx\n"
mov     ebx, edi
push    eax
call    _printf
add     esp, 10h
jmp     loc_11FE
align 10h

loc_1310:
mov     ebx, edi
call    _getchar_unlocked
mov     [ebp+var_58], 0
mov     esi, eax
lea     eax, [eax-30h]
mov     [ebp+var_54], 80000000h
cmp     eax, 9
ja      loc_11F0

loc_1333:
lea     eax, [ebp+var_3C]
mov     [ebp+var_3C], 30303030h
mov     [ebp+var_5C], eax
mov     [ebp+var_38], 30303030h
mov     [ebp+var_34], 30303030h
mov     [ebp+var_30], 30303030h
mov     [ebp+var_2C], 30h ; '0'
mov     [ebp+var_2B], 0
cmp     esi, 30h ; '0'
jz      loc_1150
jmp     loc_1167
```

```
loc_136B:
mov     ebx, edi
call    _getchar_unlocked

loc_1372:
lea     edx, [eax-30h]
cmp     edx, 9
jbe     short loc_136B
mov     esi, eax
cmp     eax, 0FFFFFFFFh
jz      short loc_1398
lea     eax, [eax-9]
cmp     eax, 4
jbe     short loc_1398
cmp     esi, 20h ; ' '
jnz     loc_11F0
lea     esi, [esi+0]

loc_1398:
sub     esp, 4
push    [ebp+var_54]
push    [ebp+var_58]
jmp     loc_12F6
align 10h

loc_13B0:
mov     ebx, edi
call    _getchar_unlocked
mov     [ebp+var_58], 0
mov     esi, eax
lea     eax, [eax-30h]
```

```
mov     [ebp+var_54], 0
cmp     eax, 9
jbe     loc_1333
jmp     loc_11F0

loc_13D8:
lea     eax, (aLlx - 3FC0h)[edi] ; "%llx\n"
push    esi
push    [ebp+var_54]
push    [ebp+var_58]
push    eax
call    _printf
add     esp, 10h
jmp     loc_122E

loc_13F3:
mov     ebx, edi
mov     [ebp+var_5C], edx
call    _getchar_unlocked
mov     esi, eax
lea     eax, [eax-30h]
cmp     eax, 9
ja      loc_11F0
mov     edx, [ebp+var_5C]
mov     eax, esi
mov     esi, edx
cmp     edx, 11h
jnz     short loc_1429
jmp     loc_157D
align 10h

loc_1420:
```

```
cmp     esi, 11h
jz      loc_1572

loc_1429:
mov     edx, [ebp+var_4C]
add     esi, 1
mov     ebx, edi
mov     [edx+esi], al
call    _getchar_unlocked
lea     ecx, [eax-30h]
cmp     ecx, 9
jbe     short loc_1420

loc_1441:
mov     esi, eax
jmp     loc_11D1

loc_1448:
mov     ebx, edi
call    _getchar_unlocked
mov     esi, eax
cmp     eax, 30h ; '0'
jnz     loc_1678
xor     esi, esi
jmp     short loc_1468
align 10h

loc_1460:
cmp     esi, 270Fh
jg      short loc_1477

loc_1468:
```

```
mov     ebx, edi
add     esi, 1
call    _getchar_unlocked
cmp     eax, 30h ; '0'
jz      short loc_1460

loc_1477:
mov     edx, esi
mov     [ebp+var_50], esi
mov     esi, eax
cmp     edx, 2710h
jz      loc_1372

loc_148A:
cmp     esi, 0FFFFFFFFh
jz      loc_1398
lea     eax, [esi-9]
cmp     eax, 4
jbe     loc_1398
cmp     esi, 20h ; ' '
jz      loc_1398
lea     eax, [esi-30h]
cmp     eax, 9
ja      loc_11F0
lea     eax, [ebp+var_3D]
xor     edx, edx
mov     [ebp+var_4C], eax
mov     eax, esi
mov     esi, edx
jmp     short loc_14CB

loc_14C2:
```

```
cmp     esi, 11h
jz      loc_165C

loc_14CB:
mov     edx, [ebp+var_4C]
add     esi, 1
mov     ebx, edi
mov     [edx+esi], al
call    _getchar_unlocked
lea     ecx, [eax-30h]
cmp     ecx, 9
jbe     short loc_14C2
mov     esi, eax

loc_14E5:
cmp     esi, 0FFFFFFFFh
jz      short loc_14FB
lea     eax, [esi-9]
cmp     eax, 4
jbe     short loc_14FB
cmp     esi, 20h ; ' '
jnz     loc_11F0

loc_14FB:
mov     eax, 1Eh
sub     eax, [ebp+var_50]
test    eax, eax
jle     loc_1398
mov     ecx, [ebp+var_5C]
shl     eax, 19h
mov     [ebp+var_5C], esi
xor     edx, edx
```

```
mov     [ebp+var_50], eax
xor     eax, eax
mov     esi, ecx
jmp     short loc_153E

loc_151F:
imul    ebx, edx, 0Ah
mov     [ebp+var_4C], ebx
mov     ebx, 0Ah
mul     ebx
mov     ebx, ecx
add     edx, [ebp+var_4C]
sar     ebx, 1Fh
add     ecx, 0FFFFFFD0h
adc     ebx, 0FFFFFFFFh
add     eax, ecx
adc     edx, ebx

loc_153E:
movsx   ecx, byte ptr [esi]
add     esi, 1
test    cl, cl
jnz     short loc_151F
add     eax, 903F0000h
mov     ecx, [ebp+var_50]
mov     esi, [ebp+var_5C]
push    ebx
adc     edx, 0FFDC790Dh
mov     [ebp+var_70], eax
or      ecx, [ebp+var_54]
mov     eax, edx
or      eax, ecx
```

```
mov     [ebp+var_6C], eax
push    [ebp+var_6C]
push    [ebp+var_70]
jmp     loc_12F6

loc_1572:
mov     esi, eax
cmp     ecx, 9
ja      loc_11D1

loc_157D:
mov     ebx, edi
call    _getchar_unlocked
lea     edx, [eax-30h]
cmp     edx, 9
ja      loc_1441
mov     ebx, edi
call    _getchar_unlocked
lea     edx, [eax-30h]
cmp     edx, 9
jbe     short loc_157D
jmp     loc_1441

loc_15A4:
mov     edx, esi
mov     esi, eax
cmp     edx, 2710h
jz      loc_11F0
cmp     eax, 2Eh ; '.'
jz      loc_1684

loc_15BD:
```

```
cmp     esi, 0FFFFFFFFh
jz      loc_1654
lea     eax, [esi-9]
cmp     eax, 4
jbe     short loc_15D7
cmp     esi, 20h ; ' '
jnz     loc_11F0

loc_15D7:
mov     ecx, [ebp+var_4C]
add     ecx, 1Fh
cmp     ecx, 3Eh ; '>'
jg      loc_11F0
shl     ecx, 19h
xor     eax, eax
xor     edx, edx
mov     [ebp+var_50], ecx
mov     ecx, [ebp+var_5C]
mov     [ebp+var_5C], esi
mov     esi, ecx
jmp     short loc_1619

loc_15FA:
imul    ebx, edx, 0Ah
mov     [ebp+var_4C], ebx
mov     ebx, 0Ah
mul     ebx
mov     ebx, ecx
add     edx, [ebp+var_4C]
sar     ebx, 1Fh
add     ecx, 0FFFFFFD0h
adc     ebx, 0FFFFFFFFh
```

```
add     eax, ecx
adc     edx, ebx

loc_1619:
movsx   ecx, byte ptr [esi]
add     esi, 1
test    cl, cl
jnz     short loc_15FA
push    ecx
mov     ecx, [ebp+var_50]
or      ecx, [ebp+var_54]
add     eax, 903F0000h
mov     esi, [ebp+var_5C]
adc     edx, 0FFDC790Dh
mov     [ebp+var_78], eax
or      ecx, edx
mov     [ebp+var_74], ecx
push    [ebp+var_74]
push    [ebp+var_78]
jmp     loc_12F6

loc_164B:
xor     eax, eax
xor     edx, edx
jmp     loc_12D4

loc_1654:
or      esi, 0FFFFFFFFh
jmp     loc_15D7

loc_165C:
mov     esi, eax
```

```
mov     eax, ecx
jmp     short loc_166E

loc_1662:
mov     ebx, edi
call    _getchar_unlocked
mov     esi, eax
lea     eax, [eax-30h]

loc_166E:
cmp     eax, 9
jbe     short loc_1662
jmp     loc_14E5

loc_1678:
mov     [ebp+var_50], 0
jmp     loc_148A

loc_1684:
call    _getchar_unlocked
lea     edx, [eax-30h]
cmp     edx, 9
ja      short loc_16A7

loc_1691:
mov     ebx, edi
call    _getchar_unlocked
mov     esi, eax
lea     eax, [eax-30h]
cmp     eax, 9
jbe     short loc_1691
jmp     loc_15BD
```

```
loc_16A7:
sub     esp, 0Ch
push    eax
call    skip_invalid
add     esp, 10h
mov     esi, eax
jmp     loc_11FE

loc_16BA:
call    __stack_chk_fail_local
main endp ; sp-analysis failed



; Attributes: fuzzy-sp

public _start
_start proc near

var_C= dword ptr -0Ch

xor     ebp, ebp
mov     eax, esp
and     esp, 0FFFFFFF0h
push    eax
push    eax
call    $+5
add     [esp+0Ch+var_C], 2833h
push    eax
call    $+5
_start endp ; sp-analysis failed
```

```
public _start_c
_start_c proc near

arg_0= dword ptr  4

push    ebx
call    __x86_get_pc_thunk_bx
add     ebx, 28E0h
sub     esp, 10h
mov     eax, [esp+14h+arg_0]
push    0               ; rtld_fini
lea     edx, [eax+4]
push    ds:(_term_proc_ptr - 3FC0h)[ebx] ; fini
push    ds:(_init_proc_ptr - 3FC0h)[ebx] ; init
push    edx             ; ubp_av
push    dword ptr [eax] ; argc
push    ds:(main_ptr - 3FC0h)[ebx] ; main
call    ___libc_start_main
add     esp, 28h
pop     ebx
retn
_start_c endp



public __x86_get_pc_thunk_bx
__x86_get_pc_thunk_bx proc near
```

```
mov     ebx, [esp+0]
retn
__x86_get_pc_thunk_bx endp



deregister_tm_clones proc near
call    __x86_get_pc_thunk_ax
add     eax, 28A6h
lea     ecx, (_edata - 3FC0h)[eax]
lea     edx, (_edata - 3FC0h)[eax]
cmp     edx, ecx
jz      short locret_1747
mov     eax, ds:(_ITM_deregisterTMCloneTable_ptr - 3FC0h)[eax]
test    eax, eax
jz      short locret_1747
push    ebp
mov     ebp, esp
sub     esp, 14h
push    ecx
call    eax
add     esp, 10h
leave
retn

locret_1747:
retn
deregister_tm_clones endp
```

```
register_tm_clones proc near
call    __x86_get_pc_thunk_cx
add     ecx, 2873h
push    ebp
mov     ebp, esp
push    esi
mov     esi, 2
push    ebx
lea     ebx, (_edata - 3FC0h)[ecx]
lea     eax, (_edata - 3FC0h)[ecx]
sub     eax, ebx
sar     eax, 2
cdq
idiv    esi
test    eax, eax
jz      short loc_1788
mov     edx, ds:(_ITM_registerTMCloneTable_ptr - 3FC0h)[ecx]
test    edx, edx
jz      short loc_1788
push    ecx
push    ecx
push    eax
push    ebx
call    edx
add     esp, 10h

loc_1788:
lea     esp, [ebp-8]
pop     ebx
pop     esi
pop     ebp
```

```
retn
register_tm_clones endp



; Attributes: bp-based frame

__do_global_dtors_aux proc near
push    ebp
mov     ebp, esp
push    edi
push    esi
push    ebx
call    __x86_get_pc_thunk_bx
add     ebx, 2826h
sub     esp, 0Ch
cmp     ds:(_edata - 3FC0h)[ebx], 0
jnz     short loc_1815
cmp     ds:(__cxa_finalize_ptr - 3FC0h)[ebx], 0
jz      short loc_17C6
sub     esp, 0Ch
push    (__dso_handle - 3FC0h)[ebx]
call    ___cxa_finalize
add     esp, 10h

loc_17C6:
lea     esi, (__DTOR_LIST__ - 3FC0h)[ebx]
lea     edi, (__DTOR_END__ - 3FC0h)[ebx]
sub     edi, esi
sar     edi, 2
dec     edi
```

```
loc_17D8:
mov     eax, ds:(dtor_idx_5809 - 3FC0h)[ebx]
cmp     eax, edi
jnb     short loc_17EE
inc     eax
mov     ds:(dtor_idx_5809 - 3FC0h)[ebx], eax
call    dword ptr [esi+eax*4]
jmp     short loc_17D8

loc_17EE:
call    deregister_tm_clones
cmp     ds:(__deregister_frame_info_bases_ptr - 3FC0h)[ebx], 0
jz      short loc_180E
sub     esp, 0Ch
lea     eax, (__EH_FRAME_BEGIN__ - 3FC0h)[ebx]
push    eax
call    ___deregister_frame_info_bases
add     esp, 10h

loc_180E:
mov     ds:(_edata - 3FC0h)[ebx], 1

loc_1815:
lea     esp, [ebp-0Ch]
pop     ebx
pop     esi
pop     edi
pop     ebp
retn
__do_global_dtors_aux endp
```

```
; Attributes: bp-based frame

frame_dummy proc near

var_4= dword ptr -4

push    ebp
call    $+5
pop     edx
add     edx, 279Dh
mov     ebp, esp
push    ebx
call    __x86_get_pc_thunk_bx
add     ebx, 278Eh
push    eax
cmp     dword ptr [ebx+28h], 0
jz      short loc_185B
lea     eax, [ebx+4Ch]
push    edx
push    0
push    eax
lea     eax, [ebx-1F1Ch]
push    eax
call    ___register_frame_info_bases
add     esp, 10h

loc_185B:
mov     ebx, [ebp+var_4]
leave
jmp     register_tm_clones
frame_dummy endp
```

```
public __x86_get_pc_thunk_ax
__x86_get_pc_thunk_ax proc near
mov     eax, [esp+0]
retn
__x86_get_pc_thunk_ax endp



public __x86_get_pc_thunk_cx
__x86_get_pc_thunk_cx proc near
mov     ecx, [esp+0]
retn
__x86_get_pc_thunk_cx endp

align 10h



public skip_invalid
skip_invalid proc near

arg_0= dword ptr  4

push    esi
push    ebx
call    __x86_get_pc_thunk_bx
add     ebx, 2749h
```

```
sub     esp, 10h
mov     esi, [esp+18h+arg_0]
lea     eax, (aInvalid - 3FC0h)[ebx] ; "invalid"
push    eax
call    _puts
add     esp, 10h
cmp     esi, 0FFFFFFFFh
jz      short loc_18C7
lea     eax, [esi-9]
cmp     eax, 4
jbe     short loc_18D2
cmp     esi, 20h ; ' '
jnz     short loc_18BD
jmp     short loc_18D2
align 10h

loc_18B0:
lea     edx, [eax-9]
cmp     edx, 4
jbe     short loc_18CC
cmp     eax, 20h ; ' '
jz      short loc_18CC

loc_18BD:
call    _getchar_unlocked
cmp     eax, 0FFFFFFFFh
jnz     short loc_18B0

loc_18C7:
mov     eax, 0FFFFFFFFh

loc_18CC:
```

```
add     esp, 4
pop     ebx
pop     esi
retn

loc_18D2:
mov     eax, esi
jmp     short loc_18CC
skip_invalid endp

align 10h



public decode
decode proc near
retn
decode endp



public __x86_get_pc_thunk_di
__x86_get_pc_thunk_di proc near
mov     edi, [esp+0]
retn
__x86_get_pc_thunk_di endp



public __stack_chk_fail_local
```

```
__stack_chk_fail_local proc near
push    ebx
call    __x86_get_pc_thunk_bx
add     ebx, 26D5h
sub     esp, 8
call    ___stack_chk_fail
add     esp, 8
pop     ebx
retn
__stack_chk_fail_local endp



__do_global_ctors_aux proc near
call    __x86_get_pc_thunk_ax
add     eax, 26BDh
push    ebp
mov     ebp, esp
push    ebx
push    edx
lea     ebx, (__CTOR_LIST__ - 3FC0h)[eax]

loc_1913:
mov     eax, [ebx]
cmp     eax, 0FFFFFFFFh
jz      short loc_1921
call    eax
sub     ebx, 4
jmp     short loc_1913

loc_1921:
```

```
pop     eax
pop     ebx
pop     ebp
retn
__do_global_ctors_aux endp

_text ends


; Segment type: Pure code
; Segment permissions: Read/Execute
_fini segment byte public 'CODE' use32
assume cs:_fini
;org 1925h
assume es:nothing, ss:nothing, ds:_data, fs:nothing, gs:nothing



public _term_proc
_term_proc proc near
sub     esp, 0Ch        ; _fini
call    __do_global_dtors_aux
add     esp, 0Ch
retn
_term_proc endp

_fini ends


; Segment type: Pure data
; Segment permissions: Read
_rodata segment byte public 'CONST' use32
```

```
assume cs:_rodata
;org 2000h
aInvalid db 'invalid',0
aLlx db '%llx',0Ah,0
_rodata ends


; Segment type: Pure data
; Segment permissions: Read
LOAD segment mempage public 'DATA' use32
assume cs:LOAD
;org 200Eh
align 10h
LOAD ends


; Segment type: Pure data
; Segment permissions: Read
_eh_frame_hdr segment dword public 'CONST' use32
assume cs:_eh_frame_hdr
;org 2010h
__GNU_EH_FRAME_HDR db    1
db  1Bh
db    3
db  3Bh ; ;
db  40h ; @
db    0
db    0
db    0
db    7
db    0
db    0
```

```
db    0
db  10h
db 0F0h
db 0FFh
db 0FFh
db  5Ch ; \
db    0
db    0
db    0
db  70h ; p
db 0F0h
db 0FFh
db 0FFh
db  80h
db    0
db    0
db    0
db  90h
db 0F0h
db 0FFh
db 0FFh
db 0DCh
db    0
db    0
db    0
db  60h ; `
db 0F8h
db 0FFh
db 0FFh
db  94h
db    0
db    0
```

```
db    0
db 0D0h
db 0F8h
db 0FFh
db 0FFh
db 0C8h
db    0
db    0
db    0
db 0D1h
db 0F8h
db 0FFh
db 0FFh
db  24h ; $
db    1
db    0
db    0
db 0D5h
db 0F8h
db 0FFh
db 0FFh
db  38h ; 8
db    1
db    0
db    0
_eh_frame_hdr ends


; Segment type: Pure data
; Segment permissions: Read
_eh_frame segment dword public 'CONST' use32
assume cs:_eh_frame
```

```
;org 2054h
db  14h
db   0
db   0
db   0
db   0
db   0
db   0
db   0
db   1
db  7Ah ; z
db  52h ; R
db   0
db   1
db  7Ch ; |
db   8
db   1
db  1Bh
db  0Ch
db   4
db   4
db  88h
db   1
db   0
db   0
db  20h
db   0
db   0
db   0
db  1Ch
db   0
db   0
```

```
db    0
db 0ACh
db 0EFh
db 0FFh
db 0FFh
db  60h ; `
db    0
db    0
db    0
db    0
db  0Eh
db    8
db  46h ; F
db  0Eh
db  0Ch
db  4Ah ; J
db  0Fh
db  0Bh
db  74h ; t
db    4
db  78h ; x
db    0
db  3Fh ; ?
db  1Ah
db  3Bh ; ;
db  2Ah ; *
db  32h ; 2
db  24h ; $
db  22h ; "
db  10h
db    0
db    0
```

```
db    0
db  40h ; @
db    0
db    0
db    0
db 0E8h
db 0EFh
db 0FFh
db 0FFh
db  18h
db    0
db    0
db    0
db    0
db    0
db    0
db    0
__EH_FRAME_BEGIN__ dd offset word_30
dd offset dword_54
db 0C4h
db 0F7h
db 0FFh
db 0FFh
db  66h ; f
db    0
db    0
db    0
db    0
db  41h ; A
db  0Eh
db    8
db  86h
```

```
db   2
db  41h ; A
db  0Eh
db  0Ch
db  83h
db   3
db  4Eh ; N
db  0Eh
db  1Ch
db  4Bh ; K
db  0Eh
db  20h
db  48h ; H
db  0Eh
db  10h
db  7Ch ; |
db  0Ah
db  0Eh
db  0Ch
db  41h ; A
db 0C3h
db  0Eh
db   8
db  41h ; A
db 0C6h
db  0Eh
db   4
db  41h ; A
db  0Bh
db   0
db   0
db  10h
```

```
db    0
db    0
db    0
db  88h
db    0
db    0
db    0
db    0
db 0F8h
db 0FFh
db 0FFh
db    1
db    0
db    0
db    0
db    0
db    0
db    0
db    0
db  44h ; D
db    0
db    0
db    0
db  9Ch
db    0
db    0
db    0
db 0ACh
db 0EFh
db 0FFh
db 0FFh
db  1Fh
```

```
db   6
db   0
db   0
db   0
db  44h ; D
db  0Ch
db   1
db   0
db  47h ; G
db  10h
db   5
db   2
db  75h ; u
db   0
db  43h ; C
db  10h
db   7
db   2
db  75h ; u
db  7Ch ; |
db  4Eh ; N
db  0Fh
db   3
db  75h ; u
db  70h ; p
db   6
db  10h
db   6
db   2
db  75h ; u
db  78h ; x
db  10h
```

```
db   3
db   2
db  75h ; u
db  74h ; t
db   3
db  88h
db   1
db  0Ah
db 0C1h
db  0Ch
db   1
db   0
db  41h ; A
db 0C3h
db  41h ; A
db 0C6h
db  41h ; A
db 0C7h
db  41h ; A
db 0C5h
db  43h ; C
db  0Ch
db   4
db   4
db  45h ; E
db  0Bh
db   0
db  10h
db   0
db   0
db   0
db 0E4h
```

```
db    0
db    0
db    0
db 0A5h
db 0F7h
db 0FFh
db 0FFh
db    4
db    0
db    0
db    0
db    0
db    0
db    0
db    0
db  1Ch
db    0
db    0
db    0
db 0F8h
db    0
db    0
db    0
db  95h
db 0F7h
db 0FFh
db 0FFh
db  19h
db    0
db    0
db    0
db    0
```

```
db  41h ; A
db  0Eh
db   8
db  83h
db   2
db  4Eh ; N
db  0Eh
db  10h
db  48h ; H
db  0Eh
db   8
db  41h ; A
db 0C3h
db  0Eh
db   4
__FRAME_END__ db   0
db   0
db   0
db   0
_eh_frame ends


; Segment type: Pure data
; Segment permissions: Read/Write
_ctors segment dword public 'DATA' use32
assume cs:_ctors
;org 3EF0h
__CTOR_LIST__ dd 0FFFFFFFFh
__CTOR_END__ dd 0
_ctors ends
```

```
; Segment type: Pure data
; Segment permissions: Read/Write
_dtors segment dword public 'DATA' use32
assume cs:_dtors
;org 3EF8h
__DTOR_LIST__ dd 0FFFFFFFFh
public __DTOR_END__
__DTOR_END__ dd 0
_dtors ends

; ELF Dynamic Information

; Segment type: Pure data
; Segment permissions: Read/Write
LOAD segment mempage public 'DATA' use32
assume cs:LOAD
;org 3F00h
_DYNAMIC Elf32_Dyn <1, <1>> ; DT_NEEDED libc.musl-x86.so.1
Elf32_Dyn <0Ch, <1000h>> ; DT_INIT
Elf32_Dyn <0Dh, <1925h>> ; DT_FINI
Elf32_Dyn <6FFFFEF5h, <18Ch>> ; DT_GNU_HASH
Elf32_Dyn <5, <280h>>   ; DT_STRTAB
Elf32_Dyn <6, <1B0h>>   ; DT_SYMTAB
Elf32_Dyn <0Ah, <0DFh>> ; DT_STRSZ
Elf32_Dyn <0Bh, <10h>>  ; DT_SYMENT
Elf32_Dyn <15h, <0>>    ; DT_DEBUG
Elf32_Dyn <3, <3FC0h>>  ; DT_PLTGOT
Elf32_Dyn <2, <28h>>    ; DT_PLTRELSZ
Elf32_Dyn <14h, <11h>>  ; DT_PLTREL
Elf32_Dyn <17h, <3A8h>> ; DT_JMPREL
Elf32_Dyn <11h, <360h>> ; DT_REL
Elf32_Dyn <12h, <48h>>  ; DT_RELSZ
```

```
Elf32_Dyn <13h, <8>>    ; DT_RELENT
Elf32_Dyn <18h, <0>>    ; DT_BIND_NOW
Elf32_Dyn <6FFFFFFBh, <8000001h>> ; DT_FLAGS_1
Elf32_Dyn <6FFFFFFAh, <4>> ; DT_RELCOUNT
Elf32_Dyn <0>           ; DT_NULL
align 40h
LOAD ends


; Segment type: Pure data
; Segment permissions: Read/Write
_got segment dword public 'DATA' use32
assume cs:_got
;org 3FC0h
_GLOBAL_OFFSET_TABLE_ dd offset _DYNAMIC
dd 0
dd 0
printf_ptr dd offset printf
puts_ptr dd offset puts
__stack_chk_fail_ptr dd offset __stack_chk_fail
getchar_unlocked_ptr dd offset getchar_unlocked
__libc_start_main_ptr dd offset __libc_start_main
__cxa_finalize_ptr dd offset __cxa_finalize
_init_proc_ptr dd offset _init_proc
__register_frame_info_bases_ptr dd offset __register_frame_info_bases
_ITM_registerTMCloneTable_ptr dd offset _ITM_registerTMCloneTable
__deregister_frame_info_bases_ptr dd offset __deregister_frame_info_bases
_ITM_deregisterTMCloneTable_ptr dd offset _ITM_deregisterTMCloneTable
main_ptr dd offset main
_term_proc_ptr dd offset _term_proc
_got ends
```

```
; Segment type: Pure data
; Segment permissions: Read/Write
_data segment dword public 'DATA' use32
assume cs:_data
;org 4000h
public __dso_handle
__dso_handle dd offset __dso_handle
_data ends



; Segment type: Uninitialized
; Segment permissions: Read/Write
_bss segment dword public 'BSS' use32
assume cs:_bss
;org 4004h
assume es:nothing, ss:nothing, ds:_data, fs:nothing, gs:nothing
public _edata
_edata db ?           ; Alternative name is '__TMC_END__'
                      ; completed.5807
                      ; __bss_start
align 4
dtor_idx_5809 dd ?
object_5819 db    ? ;
db    ? ;
db    ? ;
db    ? ;
db    ? ;
db    ? ;
db    ? ;
db    ? ;
db    ? ;
```

```
db    ? ;
db    ? ;
db    ? ;
db    ? ;
db    ? ;
db    ? ;
db    ? ;
db    ? ;
db    ? ;
db    ? ;
db    ? ;
db    ? ;
db    ? ;
db    ? ;
db    ? ;
_bss ends



; Segment type: Zero-length
_prgend segment byte public '' use32
_end label byte
_prgend ends



; Segment type: Externs
; extern
extrn printf:near
extrn puts:near
extrn __cxa_finalize:near ; weak
extrn __stack_chk_fail:near
extrn getchar_unlocked:near
; int _libc_start_main(int (*main)(int, char **, char **), int argc, char **ubp_av, void (*init)(void), void
(*fini)(void), void (*rtld_fini)(void), void *stack_end)
```

```
extrn __libc_start_main:near
extrn __register_frame_info_bases ; weak
extrn _ITM_registerTMCloneTable ; weak
extrn __deregister_frame_info_bases ; weak
extrn _ITM_deregisterTMCloneTable ; weak



end _start
```