Сундуки. 7-9 класс. Заключительный этап

Скупой рыцарь хранит в подвале множество сундуков. Сундуки занумерованы, начиная с единицы. В каждом сундуке могут лежать монеты, либо сундук может быть пуст. О содержимом каждого сундука есть опись. В описи могут использоваться только следующие строчные латинские буквы: *a* означает монету номиналом *1 копейка*; *b – 5 копеек*; *c – 10 копеек*; *d – 50 копеек*; *e – 1 рубль или 100 копеек*; *f – 2 рубля*; *g – 5 рублей*; *h – 10 рублей*; *i – 25 рублей*. Буква входит в опись столько раз, сколько монет соответствующего номинала лежит в сундуке. Буквы в описи могут идти в произвольном порядке, так как скупой рыцарь добавляет монеты в сундуки без какой-либо системы и тут же дописывает нужные буквы в описи. Например, опись *abbahihihi* означает, что в сундуке лежат 105 рублей и 12 копеек (две монеты номиналом *1 копейка*; две монеты номиналом *5 копеек*; три монеты номиналом *10 рублей*; три монеты номиналом *25 рублей*). Опись пустого сундука является пустой строкой.

Скупой рыцарь хочет найти в своём подвале два сундука, таких, что разность денежных сумм, лежащих в них, максимальна. Если в подвале есть несколько искомых пар сундуков, то скупой рыцарь выбирает из них такую пару, что сумма номеров сундуков максимальна.

Составьте программу, принимающую на вход в первой строке десятичное число N – положительное натуральное число ($2 \le N \le 1000$) – количество сундуков, а в последующих N строках – описи сундуков с номерами от 1 до N. Известно, что в каждой описи не более чем 1000 букв, то есть, в каждом сундуке не более чем 1000 монет. Программа находит номера K и L такие, что $K < L$, разность сумм денег в этих сундуках $|S_K - S_L|$ наибольшая, и сумма $K + L$ наибольшая. Программа выводит в первой строке найденное число K, а во второй строке – L. Каждое число записывается в десятичной системе без знака.

Формат ввода: В первой строке содержится десятичное число N — количество сундуков ($2 \le N \le 1000$). В следующих N строках содержатся описи монет из сундуков – последовательности, в которых могут встретиться только строчные латинские буквы *a*, *b*, . . . , *i*. Длины строк находятся в диапазоне от 0 до 1000 включительно.

Формат вывода: В первой строке выводится беззнаковое десятичное натуральное число K. Во второй строке выводится беззнаковое десятичное натуральное число L. Числа K и L таковы, что $K < L$, разность сумм денег в этих сундуках $|S_K - S_L|$ наибольшая, и сумма $K + L$ наибольшая.

Ввод примера №1:	Вывод примера №1:
2 abbahihihi cdffge	1 2
Ввод примера №2:	**Вывод примера №2:**
3 ia ia ai	2 3
Ввод примера №3:	**Вывод примера №3:**
4 hb bfffff eeeeeaeeeee h	2 4

Решение

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

остатка остаётся и количество 1-норублёвых монет прирастает), монет номиналом 5 рублей не более чем 1 (остальные конвертируются в 10-тирублёвые монеты), монет номиналом 10 рублей не более чем 2 (остальные конвертируются в 25-тирублёвые монеты, при этом либо используется ещё одна 5-тирублёвая монета, либо 5 рублей остатка остаётся и количество 5-тирублёвых монет прирастает); 3) поэлементное сравнение двух нормализованных описей в виде массивов; 4) поиск наибольшего и наименьшего элемента в последовательности описей и вывод номеров их последних вхождений. Отдельно стоит рассматривать случай, когда максимум и минимум совпадают, то есть суммы во всех сундуках равны. Тогда выводятся числа $N-1$ и N. Для решения достаточно одного прохода по последовательности, в котором совмещены посимвольный ввод описей и их обработка. Следует хранить текущий рекорд-максимум среди всех описей, которые программа успела считать и текущий рекорд-минимум. Очередная опись после считывания и нормализации сравнивается с рекордами. Если никакой рекорд не побит, то делается переход к обработке следующей описи. Если рекорд побит, то опись становится рекордом, а номер вхождения запоминается. Если опись совпадает с каким-либо рекордом, то номер вхождения этого рекорда обновляется. По окончании обработки рекорды сравниваются между собой. При их совпадении выводятся числа $N-1$ и N. При их несовпадении, выводятся запомненные номера вхождений по возрастанию.

Код возможного решения

```
program TREASURECHESTS79 (input, output);
type    chests = array ['a'..'i'] of word;
        answer = record number : word; chest : chests end;
var  CURCHEST : chests; N, I : word; CHECK : integer; CURMAX, CURMIN : answer;
procedure readchest(var CHEST : chests);
var     CH, J : char; I : word;
begin
   for J := 'a' to 'i' do CHEST[J] := 0;
   if (not EOLn) then begin
        read(CH);
        I:=999;
        while (I > 0) and (not EOLn) do begin
          CHEST[CH] := CHEST[CH] + 1;
          read(CH);
          I := I - 1;
        end;
        if (EOLn) then CHEST[CH] := CHEST[CH] + 1;
        for J := 'a' to 'h' do
          case J of
            'a', 'c': if (CHEST[J] > 4) then begin
                      CHEST[succ(J)] := CHEST[succ(J)] + CHEST[J] div 5;
                      CHEST[J] := CHEST[J] mod 5
                 end;
            'b', 'd', 'e', 'g': if (CHEST[J] > 1) then begin
                      CHEST[succ(J)] := CHEST[succ(J)] + CHEST[J] div 2;
                      CHEST[J] := CHEST[J] mod 2
                 end;
            'f', 'h': if (CHEST[J] > 2) then begin
                      CHEST[succ(J)] := CHEST[succ(J)] + (CHEST[J] * 2) div 5;
                      I := (CHEST[J] * 2) mod 5;
                      if odd(I) then
                         if (CHEST[pred(J)] > 0) then begin
                            CHEST[pred(J)] := CHEST[pred(J)] - 1;
                            I := I + 1 end
                         else begin
                            CHEST[pred(J)] := CHEST[pred(J)] + 1;
                            I := I - 1 end;
                      CHEST[J] := I div 2;
```

```
                end;
          end;
   end
end;
function CompareChest(CHEST1, CHEST2: chests) : integer;
var J : char; RESULT : integer;
begin   RESULT := 0;
        J := 'i';
        while (RESULT = 0) and (J >= 'a') do begin
            if (CHEST1[J] > CHEST2[J]) then
                RESULT := 1
            else if (CHEST1[J] < CHEST2[J]) then
                RESULT := -1;
            J := pred(J)
        end;
        CompareChest := RESULT
end;
begin   readln(N);
        with CURMAX do begin
            number := 1;
            readchest(chest);
            readln;
        end;
        CURMIN := CURMAX;
        readchest(CURCHEST);
        CHECK := CompareChest(CURMAX.chest, CURCHEST);
        if (CHECK <= 0) then
            with CURMAX do begin
               number := 2;
               chest := CURCHEST
            end;
        CHECK := CompareChest(CURCHEST, CURMIN.chest);
        if (CHECK <= 0) then
            with CURMIN do begin
               number := 2;
               chest := CURCHEST
            end;
        for I := 3 to N do begin
            readln;
            readchest(CURCHEST);
            CHECK := CompareChest(CURMAX.chest, CURCHEST);
            if (CHECK <= 0) then
                with CURMAX do begin
                   number := I;
                   chest := CURCHEST
                end;
            CHECK := CompareChest(CURCHEST, CURMIN.chest);
            if (CHECK <= 0) then
                with CURMIN do begin
                   number := I;
                   chest := CURCHEST
                end;
        end;
        if (CURMAX.number > CURMIN.number) then begin
            writeln(CURMIN.number);
            writeln(CURMAX.number) end
```

```
        else if (CURMAX.number < CURMIN.number) then begin
            writeln(CURMAX.number);
            writeln(CURMIN.number) end
        else begin
            writeln(CURMAX.number - 1);
            writeln(CURMAX.number) end
end.
```