Сундуки. 10-11 класс. Заключительный этап

Скупой рыцарь хранит в подвале множество сундуков. Сундуки занумерованы, начиная с единицы. В каждом сундуке могут лежать монеты и/или банкноты, либо сундук может быть пуст. О содержимом каждого сундука есть опись. В описи строчными латинскими буквами обозначаются монеты: *a* – монета 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*. В описи заглавными латинскими буквами обозначаются банкноты: *A* – банкнота номиналом *5 рублей*; *B* – банкнота номиналом *10 рублей*; *C* – *50 рублей*; *D* – *100 рублей*; *E* – *200 рублей*; *F* – *500 рублей*; *G* – *1000 рублей*; *H* – *2000 рублей*; *I* – *5000 рублей*. Другие символы в описях не используются. Буква входит в опись столько раз, сколько монет или банкнот соответствующего номинала лежит в сундуке. Буквы в описи могут идти в произвольном порядке, так как скупой рыцарь добавляет монеты и банкноты в сундуки без какой-либо системы и тут же дописывает нужные буквы в описи. Например, опись *abbaHihihihI* означает, что в сундуке лежат 7105 рублей и 12 копеек (две монеты номиналом *1 копейка*; две монеты номиналом *5 копеек*; три монеты номиналом *10 рублей*; три монеты номиналом *25 рублей*; одна банкнота номиналом *2000 рублей*; одна банкнота номиналом *5000 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 разность 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 и/или банкнот. Программа находит номера K и L такие, что $K < L$, разность сумм денег в этих сундуках $|S_K - S_L|$ наибольшая и разность $L - K$ 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 латинские буквы *a*, *b*, ..., *i*, *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 разность $L - K$ наибольшая.

Ввод примера №1:	Вывод примера №1:
2 abbahihihi CDFFGE	1 2
Ввод примера №2:	**Вывод примера №2:**
3 i i ai	1 3
Ввод примера №3:	**Вывод примера №3:**
4 Bb bAA eeeeeaeeeee h	1 4

Решение

В решении можно запрограммировать следующие подзадачи: 1) считывание очередной описи сундука и представление её в виде массива из 16 элементов, в каждом из которых хранится количество монет/банкнот 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/банкнот номиналом 5 рублей не более чем 1 (остальные конвертируются в 10-тирублёвые монеты/банкноты), монет/банкнот номиналом 10 рублей не более чем 2 (остальные конвертируются в 25-тирублёвые монеты, при этом либо используется ещё одна 5-тирублёвая монета/банкнота, либо 5 рублей остатка остаётся и количество 5-тирублёвых монет/банкнот прирастает); монет номиналом 25 рублей не более чем 1 (остальные конвертируются в 50-тирублёвые банкноты); банкнот номиналом 50 рублей не более чем 1 (остальные конвертируются в 100-рублёвые банкноты); банкнот номиналом 100 рублей не более чем 1 (остальные конвертируются в 200-рублёвые банкноты); банкнот номиналом 200 рублей не более чем 2 (остальные конвертируются в 500-рублёвые банкноты, при этом либо используется ещё одна 100-рублёвая банкнота, либо 100 рублей остаётся и количество 100-рублёвых банкнот прирастает); банкнот номиналом 500 рублей не более чем 1 (остальные конвертируются в 1000-рублёвые банкноты); банкнот номиналом 1000 рублей не более чем 1 (остальные конвертируются в 2000-рублёвые банкноты); банкнот номиналом 2000 рублей не более чем 2 (остальные конвертируются в 5000-рублёвые банкноты, при этом либо используется ещё одна 1000-рублёвая банкнота, либо 1000 рублей остаётся и количество 1000-рублёвых банкнот прирастает); 3) поэлементное сравнение двух нормализованных описей в виде массивов; 4) поиск наибольшего и наименьшего элемента в последовательности описей. И для минимума и для максимума следует найти номера их первого и последнего вхождения в последовательность. Затем нужно найти, какие вхождения отвечают наибольшей разности номеров и вывести номера. Для решения достаточно одного прохода по последовательности, в котором совмещены посимвольный ввод описей и их обработка. Следует хранить текущий рекорд-максимум (максимум среди всех описей, которые программа успела считать) и текущий рекорд-минимум (минимум среди всех описей, которые программа успела считать). Очередная опись после считывания и нормализации сравнивается с обоими рекордами. Если опись хуже каждого рекорда, то делается переход к обработке следующей описи. Если опись совпадает с каким-либо рекордом, то у этого рекорда корректируется его последнее вхождение. Если очередная опись лучше рекорда, то она становится рекордом, и запоминается ее вхождение (как первое и как последнее). По окончании обработки определяется, какие номера вхождений рекордов дают максимальную разность. Найденные номера выводятся по возрастанию.

Код возможного решения

```
program TREASURECHESTS1011 (input, output);
type    coins = array ['a'..'i'] of word;
        banknotes = array ['C'..'I'] of word;
        chests = record coin : coins; banknote : banknotes end;
        answer = record num1st, numlast : word; chest : chests end;
var     CURCHEST : chests; N, I, K1, K2, L1, L2 : word;
        CHECK : integer; CURMAX, CURMIN : answer;
procedure readchest(var CHEST : chests);
var     CH, J : char; I : word;
begin with CHEST do begin
   for J := 'a' to 'i' do coin[J] := 0;
   for J := 'C' to 'I' do banknote[J] := 0;
   if (not EOLn) then begin
        read(CH);
        I:=999;
        while (I > 0) and (not EOLn) do begin
          case CH of
             'a'..'i': coin[CH] := coin[CH] + 1;
             'C'..'I': banknote[CH] := banknote[CH] + 1;
             'A': coin['g'] := coin['g'] + 1;
             'B': coin['h'] := coin['h'] + 1;
```

```
  end;
  read(CH);
  I := I - 1;
end;
if (EOLn) then
  case CH of
     'a'..'i': coin[CH] := coin[CH] + 1;
     'C'..'I': banknote[CH] := banknote[CH] + 1;
     'A': coin['g'] := coin['g'] + 1;
     'B': coin['h'] := coin['h'] + 1;
  end;
for J := 'a' to 'i' do
  case J of
    'a', 'c': if (coin[J] > 4) then begin
              coin[succ(J)] := coin[succ(J)] + coin[J] div 5;
              coin[J] := coin[J] mod 5
          end;
    'b', 'd', 'e', 'g': if (coin[J] > 1) then begin
              coin[succ(J)] := coin[succ(J)] + coin[J] div 2;
              coin[J] := coin[J] mod 2
          end;
    'f', 'h': if (coin[J] > 2) then begin
              coin[succ(J)] := coin[succ(J)] + (coin[J] * 2) div 5;
              I := (coin[J] * 2) mod 5;
              if odd(I) then
                 if (coin[pred(J)] > 0) then begin
                    coin[pred(J)] := coin[pred(J)] - 1;
                    I := I + 1 end
                 else begin
                    coin[pred(J)] := coin[pred(J)] + 1;
                    I := I - 1 end;
                    coin[J] := I div 2;
          end;
    'i': if (coin[J] > 1) then begin
              banknote['C'] := banknote['C'] + coin[J] div 2;
              coin[J] := coin[J] mod 2
          end;
  end;
for J := 'C' to 'H' do
  case J of
    'E', 'H': if (banknote[J] > 2) then begin
              banknote[succ(J)] := banknote[succ(J)] + (banknote[J] * 2) div 5;
              I := (banknote[J] * 2) mod 5;
              if odd(I) then
                 if (banknote[pred(J)] > 0) then begin
                    banknote[pred(J)] := banknote[pred(J)] - 1;
                    I := I + 1 end
                 else begin
                    banknote[pred(J)] := banknote[pred(J)] + 1;
                    I := I - 1 end;
                    banknote[J] := I div 2;
          end;
    'C', 'D', 'F', 'G': if (banknote[J] > 1) then begin
              banknote[succ(J)] := banknote[succ(J)] + banknote[J] div 2;
              banknote[J] := banknote[J] mod 2
          end;
```

```
          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 >= 'C') do begin
            if (CHEST1.banknote[J] > CHEST2.banknote[J]) then
                RESULT := 1
            else if (CHEST1.banknote[J] < CHEST2.banknote[J]) then
                RESULT := -1;
            J := pred(J)
        end;
        J := 'i';
        while (RESULT = 0) and (J >= 'a') do begin
            if (CHEST1.coin[J] > CHEST2.coin[J]) then
                RESULT := 1
            else if (CHEST1.coin[J] < CHEST2.coin[J]) then
                RESULT := -1;
            J := pred(J)
        end;
        CompareChest := RESULT
end;
begin   readln(N);
        with CURMAX do begin
            num1st := 1; numlast := 1;
            readchest(chest);
            readln;
        end;
        CURMIN := CURMAX;
        readchest(CURCHEST);
        CHECK := CompareChest(CURMAX.chest, CURCHEST);
        if (CHECK <= 0) then begin
            with CURMAX do begin
               numlast := 2;
               if (CHECK < 0) then num1st := 2 else CURMIN.numlast := 2;
               chest := CURCHEST
            end end
        else
            with CURMIN do begin
               num1st := 2; numlast := 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
                   num1st := I; numlast := I;
                   chest := CURCHEST
            end
            else if (CHECK = 0) then CURMAX.numlast := I
            else begin
```

```
            CHECK := CompareChest(CURCHEST, CURMIN.chest);
            if (CHECK < 0) then
                with CURMIN do begin
                   num1st := I; numlast := I;
                   chest := CURCHEST
            end
            else if (CHECK = 0) then CURMIN.numlast := I
        end;
    end;
    if (CURMAX.numlast > CURMIN.num1st) then begin
        K1 := CURMIN.num1st;
        L1 := CURMAX.numlast end
    else begin
        K1 := CURMAX.numlast;
        L1 := CURMIN.num1st
    end;
    if (CURMAX.num1st > CURMIN.numlast) then begin
        K2 := CURMIN.numlast;
        L2 := CURMAX.num1st end
    else begin
        K2 := CURMAX.num1st;
        L2 := CURMIN.numlast
    end;
    if (L1 - K1) >= (L2 - K2) then begin
        writeln(K1);
        writeln(L1) end
    else begin
        writeln(K2);
        writeln(L2) end
end.
```