Domino

În jocul Domino se utilizează 28 de piese distincte plate, dreptunghiulare. Fiecare piesă se identifică prin două numere, simbolizate prin puncte (*figura 1*). Însemnarea acestor numere pe fiecare placă se efectuează prin divizarea feţei respective în două părţi egale, fiecare număr fiind din mulţimea {0, 1, 2, 3, 4, 5, 6}.

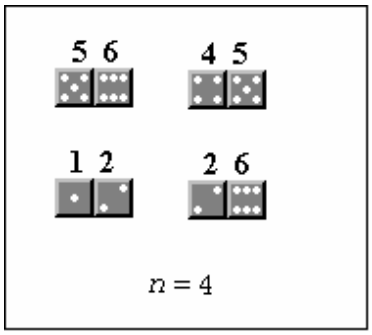

Fig. 1

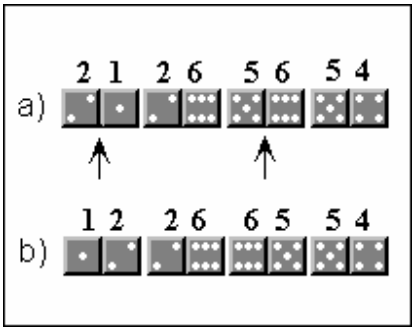

Fig. 2

Utilizînd *n* piese distincte de domino în calitate de vagoane imaginare, pe masă de joc se construieşte un "tren" (*fig. 2a*). Trenul construit se numeşte **tren perfect** dacă cele două numere de pe părţile vecine ale oricăror două piese adiacente sînt egale între ele (*fig. 2b*). În unele cazuri, un tren obişnuit poate fi transformat în unul perfect prin rotaţia cu 180° a anumitor piese. De exemplu, trenul perfect din *fig. 2b* poate fi obţinut din trenul obişnuit din *fig. 2a* prin rotaţia celor două piese indicate cu săgeată.

Elaboraţi un program care transformă, daca-i posibil, un tren obişnuit în unul perfect.

Date de intrare.

Fişierul text `DOMINO.IN` conţine pe prima linie numărul de piese *n*. Următoarele *n* linii ale fişierului de intrare conţin cîte două numere întregi separate prin spaţiu, fiecare linie reprezentînd o piesă de domino. Piesele sînt listate în ordina apariţiei lor în tren.

Date de ieşire.

Fişierul text `DOMINO.OUT` va conţine pe prima linie cuvîntul `DA` dacă trenul din fişierul de intrare poate fi transformat în unul perfect şi `NU` în caz contrar. În cazul răspunsului `DA` următoarele *n* linii ale fişierului de ieşire vor conţine cîte două numere întregi separate prin spaţiu, fiecare linie reprezentînd o piesă de domino. Piesele vor fi listate în ordinea apariţiei lor în fişierul de intrare, astfel încît cele două numere de pe părţile vecine ale oricăror două piese adiacente să fie egale între ele.

Exemplu.

`DOMINO.IN`

```
4
2 1
2 6
5 6
5 4
```

`DOMINO.OUT`

```
DA
1 2
2 6
6 5
5 4
```

Restricţii. $1 \le n \le 28$. Timpul de execuţie nu va depăşi 1 secundă. Fişierul sursă va avea denumirea `DOMINO.PAS`, `DOMINO.C` sau `DOMINO.CPP`.

Rezolvare

Vom identifica piesele trenului prin perechile ordonate (a_i, b_i), $i = 1, 2, ..., n$. Evident, operaţia de rotire a piesei i poate fi redată prin notaţia $(a_i, b_i) \rightarrow (b_i, a_i)$.

Presupunem că trenul iniţial poate fi transformat în unul perfect. În aceste condiţii sînt posibile următoarele cazuri:

a) piesa $i = 1$ şi-a păstrat poziţia iniţială (a_i, b_i);

b) piesa $i = 1$ a fost rotită, avînd în trenul perfect poziţia (b_i, a_i).

În continuare, vom încerca să construim trenul perfect pentru cazurile *a*) şi *b*) separat, rotind, în caz de necesitate, piesele 2, 3, ..., *n*.

Admitem, că în procesul examinării consecutive a pieselor din componenţa trenului, piesele $i = 1, 2, ..., k-1$ şi-au ocupat deja poziţia lor corectă şi se examinează piesa *k*. Evident, sînt posibile următoarele cazuri:

1) $b_{k-1} = a_k$. Evident, piesa *k* ocupă poziţia corectă. Se trece la piesa $k + 1$.

2) $b_{k-1} = b_k$. În acest caz piesa *k* trebuie rotită: $(a_k, b_k) \rightarrow (b_k, a_k)$. În continuare se trece la piesa $k + 1$.

3) $b_{k-1} \neq a_k$, $b_{k-1} \neq b_k$. Evident, trenul nu poate fi transformat în unul perfect. Stop.

Accentuăm, că piesa $k - 1$, care se află înaintea piesei curente *k*, nu mai poate fi rotită, întrucît poziţia ei este determinată de poziţia piesei $k - 2$, poziţia căreia, la rîndul ei, este determinată de poziţia piesei $k - 3$ ş.a.m.d., pînă la piesa 1.

În programul ce urmează, fiecare piesă de domino este reprezentată cu ajutorul tipului de date

```
Piesa = record
          a, b : 0..6;
        end;
```

iar trenul T – cu ajutorul variabilei

```
T : array[1..28] of Piesa;
```

Evident, algoritmul care verifică condiţiile *1*) – *3*) şi roteşte, în caz de necesitate piesele respective, poate fi realizat printr-o singură parcurgere a tabloului `T`.

```
Program Domino;
 { Clasele 7-9 }
type Piesa = record
               a, b : 0..6;
             end;
var T : array[1..28] of Piesa; { trenul }
    n : 1..28; { numarul de vagoane in tren }
    Perfect : boolean;
    c : integer; { variabila de lucru }

procedure Citeste;
 { Citirea datelor din fisierul de intrare }
var i : integer;
    Intrare : text;
begin
```

```
   assign(Intrare, 'DOMINO.IN');
   reset(Intrare);
   readln(Intrare, n);
   for i:=1 to n do
      readln(Intrare, T[i].a, T[i].b);
   close(Intrare);
end; { Citeste }

procedure Scrie;
 { Scrierea datelor in fisierul de iesire }
var i : integer;
    Iesire : text;
begin
   assign(Iesire, 'DOMINO.OUT');
   rewrite(Iesire);
   if not Perfect then writeln(Iesire, 'NU')
      else begin
             writeln(Iesire, 'DA');
             for i:=1 to n do
               writeln(Iesire, T[i].a, ' ', T[i].b);
           end; { else }
   close(Iesire);
end; { Scrie }

procedure Transforma;
var j, k : integer;
begin
  Perfect:=true;
  k:=2;
  while ((k<=n) and Perfect) do
    begin
      if ((T[k-1].b<>T[k].a) and (T[k-1].b<>T[k].b))
        then Perfect:=false;
      if (T[k-1].b=T[k].b) then
        begin
          { rotim piesa T[k] }
          j:=T[k].a; T[k].a:=T[k].b; T[k].b:=j;
        end; { then }
      k:=k+1;
    end; { while }
end; { Transforma }

begin
  Citeste;
  Transforma;
  if not Perfect then
     begin
       { rotim prima piesa }
       c:=T[1].a; T[1].a:=T[1].b; T[1].b:=c;
       Transforma;
     end; { then }
  Scrie;
end.
```