Zadanie zaliczeniowe nr 10 (2 trymestr), 19 II 2003

Każda permutacja składa się z jednego lub więcej rozłącznych cykli, np. permutacja liczb od **1** do **6** postaci **4 2 6 1 3 5** ma trzy cykle: **<1,4> <2> <3,6,5>**.

Permutacje ciągu liczb całkowitych możemy reprezentować korzystając z deklaracji:

```
type lista=^wezel;
     wezel=record
             wart:integer;
             nast:lista
           end;
     permutacja=^wperm;
     wperm=record
             cykl:lista;
             nast:permutacja;
           end
```

Permutacja będzie przechowywana jako lista cykli a cykle jako listy cykliczne liczb całkowitych.

Napisz program, który wczyta ze standardowego wejścia permutację liczb **1..n** (stałej **n** nie znamy) i wypisze na standardowe wyjście permutację odwrotną do niej, tzn. mającą odwróconą kolejność elementów w cyklach.

Program główny powinien mieć postać:

```
var memStart:longint;
    p1,p2:permutacja;
begin
  memStart:=memAvail;
  p1:=wczytaj;
  if p1=nil then
    writeln('Bledne dane')
  else begin
    p2:=odwrotna(p1);
    writeln('Permutacja dana:');
    wypisz(p1);
    usun(p1);
    writeln('Permutacja odwrotna:');
    wypisz(p2);
    usun(p2)
  end;
  if memStart<>memAvail then
    writeln('Program niepoprawny')
end.
```

Zakładamy, że permutacja dana i wynikowa będą zapisane jako ciąg cykli, z których każdy będzie opisany ciągiem liczb znajdujących się w cyklu kończącym się liczbą, od której rozpoczęliśmy. Np. permutacja wymieniona na początku treści zadania mogłaby być zapisana tak: **1 4 1 2 2 3 6 5 3** a permutacja do niej odwrotna: **1 4 1 2 2 3 5 6 3**.

Uwaga: program powinien sprawdzać poprawność danych wejściowych.