Problema ord

Soluție 1 – O(n) – 100p

Prelucrăm numerele aflate pe bucățile de benzi în paralel cu citirea: construim un vector (***predecesor***) care păstrează pentru fiecare număr, elementul aflat în fața numărului curent (primul de pe bucata respectivă va avea un element fictiv egal cu 0) și un alt vector (***succesor***) care păstrează elementul aflat după elementul curent. Astfel, în vectorul predecesorilor rămâne un singur număr care are predecesorul egal cu 0 (și un singur număr care are ca succesor 0). Rămâne deci, să depistăm primul număr și să afișăm elementele din succesor în succesor.

```
Subalgoritm citire(predecesor, succesor):
   Citește: n
   Pentru i = 1, n execută:                                  { stabilim pentru toate numerele }
      predecesor[i] ← 0                                      { numărul aflat în fața }
      succesor[i] ← 0                                        { și cel aflat după numărul curent }
   SfPentru
   Pentru i = 1, 2 execută:                                  { prelucrăm cele două benzi }
      Citește: buc                                           { numărul bucăților rezultate din tăierea benzii }
      Pentru k = 1, buc execută:                             { prelucrăm cele buc bucăți ale benzii actuale }
         nrAct ← 0                                           { elementul actual (deocamdată fictiv) }
         Citește: bucAct                                     { numărul numerelor pe bucata curentă }
         Pentru j = 1, bucAct execută:                       { prelucrăm numerele aflate pe bucata curentă }
            Citește: nr;                                     { numărul curent }
            Dacă nrAct ≠ 0 atunci                            { cu excepția elementului fictiv }
               predecesor[nr] ← nrAct                        { predecesorul numărului nr este nrAct pe bucata curentă }
               succesor[nrAct] ← nr                          { numărul nr urmează după nrAct }
            SfDacă
            nrAct ← nr                                       { nr devine element curent }
         SfPentru
      SfPentru
   SfPentru
SfSubalgoritm
```

```
Subalgoritm afișare(predecesor, succesor):
   start ← 1                          { căutăm elementul care îl are ca predecesor pe elementul fictiv, deci este primul }
   CâtTimp predecesor[start] ≠ 0 execută:
      start ← start + 1
   SfCâtTimp
   CâtTimp succesor[start] ≠ 0 execută:                      { mergem din element în element conform succesorilor }
      Scrie: start, " "
      start ← succesor[start]
   SfCâtTimp
   Scrie: start                                              { ultimul element din șirul stabilit inițial }
SfSubalgoritm
```

Evident, există și alte abordări, dar acest algoritm parcurge o singură dată numerele aflate pe bucățile celor două benzi, deci este optim.