Problema hrs

Soluție 1 – O(n) – 100p

Înainte de toate trebuie să stabilim în ce structură de date vom păstra elementele șirului ***hrs***. Observăm că fiecare element din șirul ***r*** se calculează pe baza precedentului, astfel am putea trage concluzia că este suficient să păstrăm în memorie aceste două elemente. Dar ne creează probleme șirul ***s*** ale cărui elemente iau valori – nu doar în funcție de valorile din propriul șir ***s*** – ci și în funcție de valorile tuturor elementelor șirului ***r*** generate până la momentul curent. Da, dar nu avem nevoie de toate elementele șirului ***r***, deoarece elementul curent din șirul ***s*** primește cea mai mică valoare care nu apare în șirul ***r***. Astfel, deoarece elementele șirului ***s*** „cresc" mai încet decât cele din ***r***, vom depista (datorită valorii maxime a lui ***n***) că este suficient ca din șirul ***r*** să păstrăm cel mult 500 de elemente (de fapt exact 345). Vom fi atenți ca elementele acestuia, precum și restul variabilelor/parametrilor să fie declarate în C/C++ de tipul `long`, iar în Pascal – de tipul `longint`.
În concluzie, vom avea un șir ***r*** (de lungime maximă 500) și o variabilă ***s*** în care generăm valoarea elementului curent din șirul ***s*** (este suficient pentru determinarea următorului element din șirul ***r***).
Observăm că valorile șirului ***s*** *cresc cu* 1*, sărind peste acea valoare care apare deja în șirul* ***r***. În variabila ***pr*** vom păstra indicele elementului șirului ***r*** care este mai mare decât valoarea curentă a variabilei ***s***.
Pornim cu primii termeni ai șirurilor ***r*** și ***s*** și determinăm cel de al ***n***-lea element al șirului ***r***, respectiv al șirului ***s***, utilizând formula $r_n = r_{n-1} + s$. Al ***n***-lea element al șirului ***s*** va fi $s_n = s_{n-1} + 1$, dacă $s_{n-1} + 1 \neq r_{pr}$, și $s_n = s_{n-1} + 2$, dacă $s_{n-1} + 1 = r_{pr}$. În cazul $s_n = s_{n-1} + 2$ valoarea ***pr*** crește cu 1, astfel indicând următorul element din șirul ***r***.
Pentru a putea determina și acei termeni din șirul ***r*** care nu sunt memorați în șirul ***r***, folosim două variabile: ***r1*** și ***r2***, unde ***r1*** este, de exemplu, al ***i*** – 1-lea element, iar ***r2*** al ***i***-lea. La fiecare pas nou, ***r1*** primește valoarea lui ***r2***, urmând să determinăm un nou ***r2***.

```
Subalgoritm hrs(n, rn, s, n):                { rn va fi al n-lea element din șirul r, s al n-lea din șirul s }
   r1 ← 1                                                          { elementul actual din șirul r }
   r[1] ← 1                                                        { memorăm primul element }
   s ← 2                                                           { primul element din șirul s }
   pr ← 2                          { poziția în șirul r, unde avem o valoare mai mare decât elementul curent în șirul s }
   k ← 1                                                           { indice în șirul r }
   i ← 1                                                           { contor până la n }
   CâtTimp i < n execută:                    { dorim să determinăm al n-lea element din șirul r și din șirul s }
      i ← i + 1
      r2 ← r1 + s                  { calculăm următorul termen în șirul r pe baza precedentului și a lui s }
      s ← s + 1                                  { s crește cu 1 față de valoarea elementului precedent }
      Dacă k < 500 atunci                        { păstrăm doar primele 500 elemente în șirul r }
         k ← k + 1
         r[k] ← r2
      SfDacă
      Dacă s = r[pr] atunci                                  { dacă valoarea s propusă se află în șirul r }
         s ← s + 1                                { s trebuie să fie cel mai mic, dar mai mare decât r[pr] }
         pr ← pr + 1                              { data viitoare comparăm cu următorul element din șirul r }
```

```
      SfDacă
      r1 ← r2                    { actualul element din șirul r va fi cel folosit la pasul următor pentru a calcula următorul element }
   SfCâtTimp
   rn ← r2                       { rn va fi rezultatul (al n-lea termen din șirul r), rn și s sunt parametri de ieșire }
SfSubalgoritm
```

Observații:

- O rezolvare în Pascal în care se păstrează întreg șirul ***r***, eventual și ***s***, nu va reuși să obțină puncte, decât pentru testele 1-7.
- Dacă în Pascal nu se lucrează cu tipul `longint`, de asemenea, nu se vor obține puncte pentru testele 4-10.