ASD - egzamin, zadania
30.01.2002

- Zadanie 1 – 18 pkt
  Rozważamy implementację AVL-drzew, w której każdy węzeł ma 4 atrybuty: *kl* – klucz, *ws* – wskaźnik zrównoważenia (-1 – przechył w lewo, 0 – równowaga, +1 – przechył w prawo), *lewy* – wskaźnik do lewego syna, *prawy* – wskaźnik do prawego syna. Drzewo jest dostępne przez wskaźnik do korzenia.
  (a - 7 pkt): Zaproponuj efektywny algorytm obliczania wysokości danego AVL-drzewa.
  (b - 11 pkt): Zaproponuj efektywny algorytm scalania dwóch drzew AVL T_1 i T_2 takich, że każdy klucz w T_1 jest mniejszy od każdego klucza w T_2.

- Zadanie 2 – 10 pkt
  Na planszy będącej ciągiem n pól dwóch graczy umieszcza na przemian po jednym kamieniu. Jeśli kamień jest umieszczany na już zajętym polu, to jeden z kamieni jest zdejmowany z planszy (jest zbijany), a drugi jest umieszczany na następnym polu. Procedura ta jest kontynuowana, aż kamień zostanie umieszczony na pustym polu. Jeśli gracz zbije piona znajdującego się na n-tym polu - przegrywa.
  Zaproponuj strukturę danych, która umożliwia efektywne przeprowadzenie rozgrywki i wykonanie następujących operacji:
  - zmianę struktury odpowiadającej ruchowi jednego gracza.
  - znalezienie dla k, $1 \leq k \leq n-1$, najbliższego pola o numerze większym niż k, na którym jest kamień (o ile takie pole istnieje),
  - znalezienie długości najdłuższego ciągu kolejnych pustych pól.

- Zadanie 3 - 12 pkt
  Dany jest niezorientowany, spójny graf $G = (V, E)$ oraz jego drzewo rozpinające $T = (V, F)$ z wierzchołkami ponumerowanymi w porządku preorder. Przyjmujemy, że $V = \{1, 2, ..., n\}$ oraz że G jest zadany przez listy sąsiedztwa. Zaproponuj efektywny algorytm, który sprawdza, czy można tak uporządkować listy sąsiedztwa grafu G, żeby T było jego DFS-drzewem rozpinającym o numeracji zgodnej z zadaną numeracją preorder.

- Zadanie 4 - 12 pkt (można wybrać zamiast innego zadania)
  Udowodnij, że dowolny algorytm stwierdzający, czy dla danego ciągu $x_1, \ldots, x_n$ liczb rzeczywistych istnieją takie i, j, że $x_i = x_j$, musi w modelu decyzyjnym wykonać $\Omega(n \log n)$ porównań.
  Wskazówka: Udowodnij, że dla dla każdej permutacji liczb $\{1, \ldots, n\}$ na wejściu, algorytm dochodzi do innego liścia z odpowiedzią "NIE".