Zadania przygotowawcze do II kolokwium (drzewa) i do egzaminu z Metod programowania w semestrze letnim 2004/2005

Dany jest typ

```
drzewo=↑ węzeł; węzeł = record w :  integer; lsyn, psyn :  drzewo;
(* opcjonalnie link:drzewo; gdy mamy dodatkowo fastrygę w drzewie *) end;
```

1. Umieść w każdym węźle drzewa binarnego wartość największego liścia pod nim podwieszonego.

2. Znajdź w drzewie binarnym liść o najmniejszej głębokości.

3. Zbuduj drzewo binarne *T1 : drzewo;* takie, że pole w z wierzchołka drzewa T o i-tym numerze w porządku infiksowym znajdzie się w i-tym wierzchołku w $T1$ względem numeracji prefiksowej. Warto znaleźć rozwiązania w których drzewo $T1$ będzie niezbyt wysokie, w szczególności nie wyższe niż T

4. Dane jest drzewo T które jest drzewem BST z kluczami będącymi liczbami całkowitymi. Napisać funkcję *function K-ty (k : integer) : węzeł* która wyliczy wskaźnik do k-tego co wielkości klucza elementu drzewa T.

5. n-ty kopiec Fibonacciego F_n jest drzewem, które definiuje się rekurencyjnie. F_0 zawiera jeden wierzchołek z pewnym kluczem. F_1 zawiera dwa wierzchołki, z których jeden zawierający mniejszy klucz jest rodzicem drugiego węzła który zawiera większy klucz. F_{n+2} definiuje się jako drzewo zawierające F_{n+1} zaczepione w korzeniu oraz F_n którego korzeń jest dzieckiem korzenia kopca F_{n+1}. Dodatkowo każdy element kopca F_n jest większy niż korzeń kopca F_{n+1}.

   Sprawdź czy dane drzewo jest kopcem Fibonacciego.

6. *Czerwono-czarnym* nazwiemy drzewo binarne o kluczach przyjmujących wartości typu `(czerwony,czarny)` spełniające następujące warunki:

   - Każdy klucz jest czerwony lub czarny
   - Każdy liść jest czarny
   - Jeżeli węzeł jest czerwony i ma ojca, to ojciec jest czarny
   - Każda ścieżka prosta prowadząca od korzenia do dowolnego liścia zawiera tę samą liczbę czarnych węzłów.

   Napisz funkcję `czcz(d:drzewo):boolean`, która sprawdzi, czy podane drzewo `d` jest czerwono-czarne.

7. Dokonano dwóch obiegów drzewa o różnych wartościach węzłów w porządku infiksowym i postfiksowym, a kolejno napotykane wartości umieszczono w dwóch tablicach, które są dane. Odtwórz to drzewo.

8. W plikach *prefix* i *infix* zapisane są dwa wyrażenia arytmetyczne złożone ze stałych a, b, c oraz operatorów wyrażeń $*, +$. Sprawdź, czy odczytane w sposób prefiksowy wyrażenie z pliku *prefix* jest identyczne z odczytanym w sposób infiksowy wyrażeniem z pliku *infix*.

9. Znajdź średnicę drzewa binarnego, czyli długość najdłuższej ścieżki łączącej pewne dwa węzły tego drzewa.

10. Sprawdź, czy w drzewie binarnym istnieje taki węzeł, że w jego lewym niepustym poddrzewie jest tyle samo węzłów, co w prawym.

11. Dany jest graf o n węzłach za pomocą tablicy n list sąsiedztwa (w k-tej liście znajdują się numery węzłów sąsiadujących z węzłem o numerze k). Sprawdź, czy graf ten jest drzewem. Rozważ dwa przypadki: gdy graf jest skierowany i nieskierowany.

12. Sprawdź, czy drzewo binarne jest zrównoważone.

13. Sprawdź, czy drzewo binarne jest BST.

14. Sprawdź, czy drzewo binarne jest kopcem (kopcem zupełnym).

15. Polu `link` każdego węzła `v` w drzewie fastrygowanym nadaj wartość wskaźnika do ojca tego węzła

16. Polu `link` każdego węzła `v` w drzewie fastrygowanym nadaj wartość wskaźnika do największego elementu w drzewie pod nim podwieszonym.

17. Polu `link` każdego węzła `v` w drzewie fastrygowanym nadaj wartość wskaźnika do dowolnego zera w drzewie pod nim podwieszonym lub `nil`, gdy zera pod nim nie ma.

18. Polu `link` każdego węzła `v` w drzewie fastrygowanym nadaj wartość wskaźnika do węzła następującego bezpośrednio po `v` w porządku a) prefiksowym, b) infiksowym, c) postfiksowym.
    Ostatni węzeł w danym porządku powinien pokazywać na `nil`.

19. W zadaniu tym przyjmujemy, że w strukturze `d:drzewo` wskaźniki na lewego i prawego syna każdego węzła `v` mogą wskazywać na dowolne węzły znajdujące się na ścieżce od `d` do węzła `v` (w szczególności na siebie). "Drzewo" `d` jest 3-krótkie, jeśli w tak zdefiniowanym grafie dla każdej ścieżki $v_0, \ldots, v_n, n \geq 3$, istnieje i takie że $(0 \leq i < n)$ $v_n = v_i$. Napisz funkcję `krotkie3(d:drzewo):Boolean` sprawdzającą, czy "drzewo" `d` jest 3-krótkie.

20. Napisz funkcję `maxpost(d:drzewo):Integer` (odpowiednio `maxpref,maxinf`), która wyznaczy maksymalną liczbę gałęzi dzielącą kolejne węzły w postfiksowym (prefiksowym, infiksowym) obiegu drzewa `d`. Jeśli drzewo ma mniej niż 2 węzły, to funkcja ta powinna przyjąć wartość -1.

21. Napisz funkcję `podizo(x,y:drzewo):Boolean`, która przyjmie wartość `true` wtedy i tylko wtedy gdy w drzewie `y` istnieje węzeł będący korzeniem poddrzewa izomorficznego z drzewem `x`.

22. Spójrzmy na drzewo binarne d wysokości h, jak na poddrzewo pełnego drzewa binarnego t o wysokości h. Pewne krawędzie i węzły drzewa t po prostu nie zostały „wybrane" do budowy drzewa d. Sąsiednim kuzynem węzła v drzewa d nazwiemy kolejny na tym samym poziomie węzeł drzewa d, a przez dystans między nimi będziemy rozumieli liczbę brakujących węzłów drzewa t, które je dzielą. Zatem dystans między braćmi wynosi 0, a na poziomie g dystans nigdy nie przekracza $2^g - 2$. Jeżeli na jakimś poziomie znajduje się tylko jeden węzeł, to dla tego poziomu maksymalny dystans między sąsiednimi kuzynami wynosi -1. Napisz funkcję `dalecykuzyni(d:drzewo):Integer`, która obliczy ile wynosi maksymalny dystans między sąsiednimi kuzynami w drzewie d.

23. Napisz funkcję even(d:drzewo):Integer, która obliczy ile liści drzewa `d` znajduje się na parzystych poziomach. Przyjmij, że korzeń jest na poziomie 0.

24. Napisz funkcję, `Ogolne2Binarne(d:DrzewoOgolne):drzewo` która drzewo ogólne `d` zamieni na binarne i przekaże je jako wynik, przy założeniu że drzewo `d` jest rzędu nieprzekraczającego 2. Gdyby było inaczej, to wynikiem ma być **nil**.

25. Napisz funkcję `AVL(d:drzewo):Boolean`, która stwierdzi, czy binarne drzewo `d` jest zrównoważonym drzewem binarnych wyszukiwań.

26. Napisz procedurę `WiekszyPrzodek(d:drzewof)`, która w każdym węźle `v` drzewa `d` pole `link` skieruje na najbliższego przodka na drodze do korzenia, którego wartość `w` jest większa, niż w węźle `v`. Jeśli takiego przodka nie ma, to pole `link` powinno wskazywać na `nil`.

27. Napisz funkcję `PelneBST(var s:stos):drzewo`, która dla posortowanego niemalejąco stosu `s` liczb całkowitych zbuduje pełne drzewo binarnych wyszukiwań zawierające wartości występujące na stosie.

28. Drzewo binarne jest stożkowe, jeśli kolejne poziomy zawierają coraz więcej węzłów. Napisz funkcję `stozkowe(d:drzewo):Boolean`, która przyjmie wartość `true` wtedy i tylko wtedy, gdy drzewo `d` jest stożkowe.

29. *Trudne i wykraczające poza program jest rozwiązanie liniowe z pamięcią O(max(h1,h2))*
Napisz funkcję `jest(d1,d2:drzewo):Boolean`, która dla dwóch drzew BST `d1` i `d2` stwierdzi, czy istnieje choć jedna wspólna wartość znajdująca się w tych drzewach.

30. Napisz funkcję `postdrzewo(l:lista):drzewo`, która utworzy zrównoważone drzewo binarne dające w obiegu postfiksowym od lewej do prawej kolejne elementy listy `l`.

31. Ścieżką zygzakowatą w drzewie binanym nazwiemy scieżkę zaczynającą się w dowolnym węźle drzewa i na przemian schodzącą w dół na lewo lub na prawo. Napisz funkcję `NajdluzszyZygzak(d:drzewo):Integer`, która obliczy długość najdłuższej ścieżki zygzakowatej w drzewie `d`. Długość ścieżki, to liczba węzłów znajdujących się na niej.

32. Fastrygowane drzewo binarne `d` bylo tworzone „od dołu" i w rezultacie w polach link każdego węzła zostały umieszczone dowiązania do ojca. Pole `link` korzenia drzewa wskazuje na `nil`. Pola `lsyn` i `psyn` drzewa `d` są niezainicjalizowane.
Napisz funkcję `ustawsynow(l:lista):drzewo`, która mając dowiązania do liści drzewa `d` umieszczone w liście `l` ustawi dowiązania `lsyn` i `psyn` w drzewie `d` i przekaże jako wynik dowiązanie do korzenia tego drzewa. O drzewie `d` wiadomo, że każdy węzeł mający prawego syna ma również lewego syna, a dodatkowo, dla każdego węzła, wszystkie liście poddrzewa lewego syna w liście `l` poprzedzają wszystkie liście poddrzewa prawego syna.

33. Napisz funkcję `razem(d1,d2:drzewof):drzewof`, która dla dwóch fastrygowanych drzew BST `d1` i `d2` z niezainicjalizowanymi polami `link` ustawi je tak, żeby zaczynajć od pewnego węzła tt v któregoś z drzew, zawierającego najmniejszą wartość i posuwając się po dowiązaniach `link`, przejść przez wszyskie węzły obu drzew o niemalejących wartościach. Wynikiem funkcji `razem` powinien być węzeł `v`.

34. Dla zadanych dwóch ciągów (w tablicach) znaleźć najdłuższy ciąg będący podciągiem obu.

35. W tablicy [1..n,1..n] wypełnionej zerami i jedynkami znajdź element, który znajduje się na skrzyżowaniu dwóch prostopadłych jedynkowych korytarzy o maksymalnej łacznej długości (czyli liczba jedynek do napotkania zera lub brzegu tablicy we wszystkich czterech kierunkach ma być maksymalna).

36. W tablicy [1..n,1..m] wypełnionej zerami i jedynkami znajdź złożony z jedynek kwadrat o maksymalnym polu.

37. W tablicy [1..n,1..m] wypełnionej zerami i jedynkami znajdź złożony z jedynek prostokąt o maksymalnym polu.

38. Dane są prawdobodobieństwa występowania elementów skończonego zbioru liniowo uporządkowanego. Znajdź drzewo BST o minimalnym oczekiwanym koszcie dostępu, w którym reprezentowany będzie ten zbiór (czyli żeby suma głębokości węzłów ważonych ich prawdopodobieństwami była możliwie mała).

39. Napisz procedurę obliczającą, ile jest różnych marszrut cyklicznych skoczka szachowego dla szachownicy $m \times n$.

40. Napisz procedurę obliczającą, ile jest różnych rozstawień n nieszachujących się hetmanów dla szachownicy $m \times n, n \leq m$.

41. Dane są dwie tablice liczb całkowitych `P[1..n] of Integer` posortowana rosnąco i `O[1..m] of Integer` posortowana niemalejąco. W tablicy `P` podane są współrzędne punktów na prostej, w tablicy `O` długości domkniętych odcinków ze zbioru $\mathcal{O}$, którymi dysponujemy. Napisz funkcję `pokrycie: Integer`, która wyliczy ile co najmniej odcinków ze zbioru $\mathcal{O}$ trzeba użyć, żeby pokryć nimi punkty zapisane w tablicy `P`.

42. Tablica A typu `tab2` jest zgodna z porządkami po współrzędnych, jeśli z tego że $A[x, y] = 1$ wynika że $A[u, w] = 1$ dla wszystkich $x \leq u \leq m, y \leq w \leq n$. Napisz funkcję `zgodna(A:tab2):Boolean` sprawdzającą, czy tablica `A` jest zgodna z porządkami po współrzędnych.

43. Dany jest typ `slowo01 = array[1..n] of 0..1`. Napisz funkcję

    `nadciag(u,v:slowo01):Integer;`

    obliczającą długość najkrótszego słowa zerojedynkowego, ktorego zarówno `u`, jak i `v` są podciągami.

44. Lista cykliczna `l` zawiera wartości dodatnie. Postępujemy następująco: pobieramy wartości wskazywane aktualnie przez pierwszy element listy i usuwamy tyle elementów z listy, ile wynosi odczytana wartość poczynając od właśnie wskazywanego jako pierwszy. Robimy to tak długo, aż albo lista będzie pusta, albo będzie zawierała jeden element (pokazujący na siebie). Jeżeli w pewnym momencie liczba elementów do usunięcia przekroczy liczbę elemntów pozostałych na liście, to w efekcie dostaniemy listę pustą.

    Napisz funkcję `cozostanie(l:lista):Integer`, która przekaże wartość elementu, który pozostanie, jako ostatni w wyniku zastosowania opisanych czynności na liście `l`, lub `-1`, jeśli lista wynikowa jest pusta.