Rândul 1

1. **Scrieți soluția (în pseudocod sau cod Java) pentru problema următoare. Specificați complexitatea pentru prima funcție cerută și descrieți pe scurt (2-3 propoziții) ideea de bază pentru soluție.**

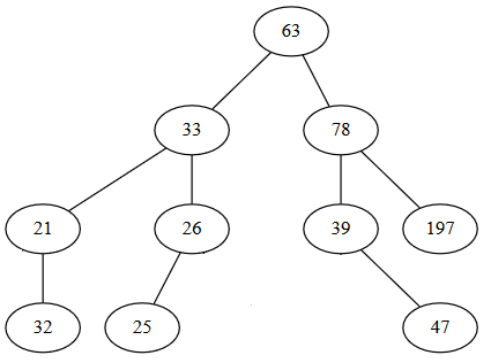
Elena locuiește pe o altă planetă pe care un an este alcătuit din N zile. Ea trebuie să ajungă la o clădire care este la M km distanță față de locația ei curentă. Ea pornește în prima zi a anului și ar vrea să ajungă cât mai repede la clădirea respectivă, dar în fiecare zi a anului poate să parcurgă maximum o anumită distanță (posibil diferită pentru zile diferite din an). Având valori pentru M, N, și cele N distanțe care pot fi parcurse în zilele anului (sub forma unui tablou de N elemente), determinați în a câta zi a anului va ajunge Elena la destinația ei.
De exemplu: dacă N este 5 (sunt 5 zile într-un an), M este 23 și distanțele care pot fi parcurse sunt [4, 9, 0, 1, 3], Elena va parcurge în prima zi 4 km, după a 2-a zi 13 km, după a 3-a zi tot 13 km, după a 4-a zi 14 km, după a 5-a zi 17 km (aici se termină anul și începem an nou) după prima zi (a anului nou) 21 km și în a 2-a zi ajunge la destinație. Deci răspunsul este că Elena ajunge la destinație în a 2-a zi a anului (nu contează că a trecut un an de când a plecat).

Va trebui să implementați 2 funcții: prima funcție primește ca parametru numerele N, M și un tablou cu N elemente reprezentând distanțele care pot fi parcurse în fiecare zi. Funcția trebuie să returneze în a câta zi a anului ajunge Elena la destinație. A 2-a funcție citește valoarea N și un tablou cu N elemente reprezentând distanțele care pot fi parcurse în fiecare zi. După aceea funcția citește pe rând valori pentru M (până când valoarea 0 este introdusă) și afișează în ce zi ajunge Elena la o destinație la M km distanță. Presupunem că pentru fiecare valoare M, ea pornește la drum în prima zi a anului.

2. **Calculați complexitatea pentru subalgoritmul următor:**

```
subalgorithm magic(n) is:
        i = 1
        while i < n execute:
                j = n
                while j > 0 execute:
                        print("*")
                        j = j / 2
                endwhile
                for (k = 0; k < n; k++) execute:
                        print ("**");
                endfor
                i = i * 2
        endwhile
endsubalgorithm
```

3. **Răspundeți la următoarele întrebări cu un desen și justificări scurte.**

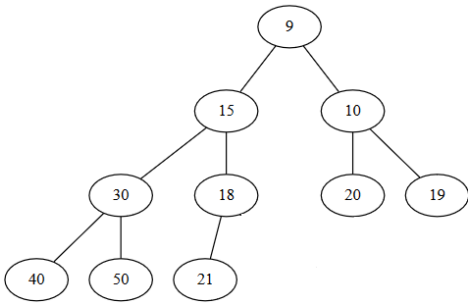


1) Pe desen aveți un arbore binar de căutare în care criteriul de ordonare al numerelor este produsul cifrelor. Arătați cum va arăta arborele după adăugarea, în această ordine, a numerelor 11, 18 și 292 (un singur desen cu rezultatele după 3 adăugări este suficient). Cum va arăta arborele dacă ștergem numărul 33 (după cele 3 adăugări)?
2) Avem un arbore binar pentru care cunoaștem parcurgerea în postordine și inordine. Parcurgerea în **postordine** este **C B H F A D X N Y Z Q M** iar parcurgerea în **inordine** este **B C D H A F M X Q Z Y N**. Construiți arborele.
3) Avem o tabelă de dispersie cu rezolvarea coliziunilor prin liste independente cu m= 11, în care vom adăuga numere alcătuite din cel mult 3 cifre, c_0, c_1 și c_2 (de exemplu pentru numărul 721 c_2=7, c_1=2, c_0=1). Vom considera 3 funcții de dispersie:

I. $d1(x)=(c_0+c_1+c_2)\ \%\ m$ II. $d2(x)=(c_1+c_2)\ \%\ m$ III. $d3(x)=(c_2)\ \%\ m$

Considerați următoarele elemente: 160, 610, 345, 254, 532, 449. Arătați cum pot fi adăugate aceste elemente în tabela de dispersie pentru fiecare funcție (3 desene diferite). Ordonați funcțiile de dispersie pe baza coliziunilor pentru acest exemplu (considerăm coliziune numărul elementelor care nu sunt singure într-o listă).

4) Având ansamblul din dreapta, ștergeți 2 elemente din el. Desenați ansamblul rezultat după fiecare ștergere. După ștergeri adăugați numărul 17 în ansamblu.



4. Alegeți răspunsul corect la următoarele întrebări și justificați alegerea făcută. La fiecare întrebare există un singur răspuns corect.

1) Având un iterator pe o Mulțime reprezentată pe o tabelă de dispersie, rezolvare coliziuni prin adresare deschisă, care dintre operațiile iteratorului NU are complexitate Θ(1)?
   a. element
   b. următor
   c. valid
   d. toate au complexitate Θ(1)

2) Dacă folosim o listă simplu înlănțuită pentru a reprezenta o coadă, unde ar trebui să punem începutul (front) pentru complexitate optimă la toate operațiile cozii?
   a. nu putem implementa o coadă pe o listă simplu înlănțuită
   b. La începutul listei
   c. la sfârșitul listei
   d. nu contează dacă la început sau la sfârșit

3) Cât este valoarea expresiei următoare (în forma postfixată): 6 3 2 4 7 + - * - ?
   a. O valoare între -100 și -15
   b. O valoare între -15 și -5
   c. O valoare între -5 și 5
   d. O valoare între 5 și 15
   e. O valoare între 15 și 100

4) Care dintre următoarele containere nu poate fi implementat pe o tabelă de dispersie?
   a. Mulțime
   b. Colecție
   c. Stivă
   d. Dicționar

5) Care este cea mai bună reprezentare pentru o Coadă cu Priorități dacă vrem ca operațiile *adaugă* și *șterge* să aibă complexitate în caz defavorabil $\Theta(\log_2 n)$ iar operația element să aibă complexitate Θ(1)?
   a. Vector Dinamic
   b. Ansamblu
   c. Arbore Binar de Căutare
   d. Listă Dublu înlănțuită

6) Să considerăm H un ansamblu cu 16 noduri. Care afirmație este adevărată pentru cel mai Scurt Drum (SD) și cel mai Lung Drum (LD) de la rădăcină la o frunză?
   a. SD este 2 și LD este 4
   b. SD este 2 și LD este 3
   c. SD este 3 și LD este 5
   d. SD este 3 și LD este 4
   e. SD este 3 și LD este 3

5. **Avem containerul MulțimeOrdonată, reprezentată pe o listă simplu înlănțuită. Dați reprezentarea MulțimiiOrdonate (ce structuri și ce câmpuri sunt folosite). Specificați și implementați operația de adăugare. Cât este complexitatea operației? Arătați reprezentarea Iteratorului și implementați la alegere 2 operații de la iterator.**

6. **Avem containerul Dicționar, reprezentat pe o Tabelă de Dispersie, rezolvare coliziuni prin adresare deschisă (puteți folosi orice funcție de dispersie cu verificare pătratică). Dați reprezentarea Dicționarului (ce structuri și ce câmpuri sunt folosite). Specificați și implementați operația de căutare. Cât este complexitatea operației?**

Punctaj: 1 – 1p; 2 – 1p; 3 – 3p (4*0.75); 4 – 2p (6*0.33); 5 – 1.5p; 6 – 1.5p, Of – 1p;

Rândul 2

1. **Scrieți soluția (în pseudocod sau în cod Java) pentru problema următoare. Specificați complexitatea pentru prima funcție cerută și descrieți pe scurt (2-3 propoziții) ideea de bază pentru soluție.**

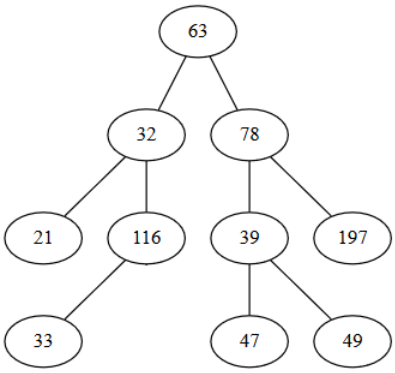
Oana se plimbă prin pădure și găsește un arbore magic. Pe fiecare frunză a acestui arbore este un string care reprezintă o expresie aritmetică, formată din cifre (0-9) și operatorii + și – (nu există paranteze, nu există înmulțire sau împărțire). Oana vrea să ia acasă frunza pe care este scrisă expresia care are cea mai mare valoare. Ajutați-o pe Oana să găsească această frunză. Dacă sunt mai multe frunze care au expresie de valoare maximă, Oana alege ultima expresie găsită.
De exemplu: dacă o frunză conține expresia 4+2-5+3+2-4 valoarea frunzei este 2, dacă frunza conține expresia 1+1+1+1 valoarea frunzei este 4. Dacă după frunza cu 1+1+1+1 Oana găsește frunza 2+7+1-6 (tot de valoare 4) ea va alege frunza 2+7+1-6.

Va trebui să implementați 2 funcții: prima funcție primește ca parametru o expresie (sub forma unui string) și returnează valoarea expresiei. A 2-a funcție citește pe rând expresiile (până se introduce un String vid), calculează valoarea lor și la final returnează expresia de pe frunza luată acasă de Oana.
Pentru parcurgerea expresiei puteti folosi funcția charAt(i) din Java care vă dă caracterul de pe o poziție dintr-un string, iar pentru transformarea unui caracter in cifra corespunzătoare puteți folosi (int cifra = caracter – '0').

2. **Calculați complexitatea pentru subalgoritmul următor:**

```
subalgorithm magic(n) is:
      for (i = 0; i < 5; i++) execute:
            for (j = 0; j < n; j++) execute:
                  for (k = 0; k < n; k++) execute:
                        print ("*")
                  endfor
            endfor
            for (m = 0; m < n; m = m + 4) execute:
                  print("**")
            endfor
      endfor
endsubalgorithm
```

3. **Răspundeți la următoarele întrebări cu un desen și justificări scurte.**
   1) Pe desen aveți un arbore binar de căutare în care criteriul de ordonare al numerelor este suma cifrelor. Arătați cum va arăta arborele după adăugarea, în această ordine, a numerelor 11, 61, 40 (un singur desen cu rezultatele după 3 adăugări e suficient). Cum va arăta arborele dacă ștergem numărul 32 (după cele 3 adăugări)?



   2) Care dintre tablourile următoare reprezintă un ansamblu (doar o soluție este corectă)?
      a. [1, 12, 23, 10, 15, 38, 45, 15, 18, 20, 21]
      b. [1, 8, 27, 10, 45, 83, 91, 31, 12, 52, 51]
      c. [1, 13, 20, 21, 65, 54, 67, 41, 30, 83, 52]
      În varianta selectată adăugați numărul 23 și arătați ansamblul (în forma de arbore) după adăugare.
   3) Avem o tabelă de dispersie cu rezolvare coliziuni prin liste independente, de dimensiune m = 7. Arătați cum arată tabela de dispersie după adăugarea elementelor: 14, 32, 19, 102, 59, 66, 91, 73.
   4) Arătați cum poate fi implementată o Stivă folosind 2 Cozi. Presupunem că avem TAD Coadă implementată (fiecare operație are complexitate Θ(1)), și vrem să implementăm Stiva folosind 2 cozi. Arătați cum putem face acest lucru, astfel încât Stiva să aibă o singură operație cu complexitate diferită de Θ(1). Explicați cum se folosește coadă și explicați cum ar trebui implementată fiecare operație de la Stivă.

4. Alegeți răspunsul corect la următoarele întrebări și justificați alegerea făcută. La fiecare întrebare există un singur răspuns corect.

1) Într-o implementare optimă a unei cozi pe o listă simplu înlănțuită, care operație are complexitatea Θ(n) în caz defavorabil?
   a. Adaugă (push)
   b. Sterge (pop)
   c. Element (top/peek)
   d. Vidă (isEmpty)
   e. Niciuna dintre operații

2) Considerați tabela de dispersie de mai jos, construită cu funcția de dispersie d(c,i) = (c % m + i^2) % m. În ce ordine puteau fi adăugate elementele?

	25	30		11	61	

   a. 25, 61, 11, 30
   b. 11, 25, 61, 30
   c. 11, 30, 61, 25
   d. 61, 30, 25, 11

3) Cât este valoarea expresiei următoare (în forma postfixată): 9 1 6 5 3 4 - - + * + ?
   a. O valoare între -100 și -15
   b. O valoare între -15 și -5
   c. O valoare între -5 și 5
   d. O valoare între 5 și 15
   e. O valoare între 15 și 100

4) Ce structură de date este cea mai potrivită pentru implementarea unei Liste dacă trebuie să răspundem des la întrebarea următoare: "care element este pe poziția *i*"?
   a. doar vectorul dinamic
   b. doar lista simplu înlănțuită
   c. doar lista dublu înlănțuită
   d. doar tabela de dispersie

5) Avem o stivă inițial vidă în care adăugăm, în această ordine, elementele A, B, C, D, E. După adăugări ștergem 4 elemente și după fiecare ștergere adăugăm elementul șters într-o coadă inițial vidă. Dacă ștergem 2 elemente din coadă, ce element rămâne la începutul (front) cozii?
   a. A
   b. B
   c. C
   d. D
   e. E

6) Din care combinație de parcurgeri nu se poate reconstrui un arbore binar?
   a. Preordine și Inordine
   b. Postordine și Inordine
   c. Preordine și Postordine
   d. Din orice combinație de 2 parcurgeri se poate reconstrui arborele

5. Avem containerul Colecție, reprezentat pe o listă dublu înlănțuită în care fiecare nod reține un element (unic) și frecvența lui. Dați reprezentarea Colecției (ce structuri și ce câmpuri sunt folosite). Specificați și implementați operația de ștergere. Cât este complexitatea operației? Arătați reprezentarea Iteratorului și implementați la alegere 2 operații de la iterator.

6. Avem containerul MulțimeOrdonată, reprezentat pe un Arbore Binar de Căutare. Dați reprezentarea MulțimiiOrdonate (ce structuri și ce câmpuri sunt folosite). Specificați și implementați operația de adăugare. Cât este complexitatea operației?

Punctaj: 1 – 1p; 2 – 1p; 3 – 3p (4*0.75); 4 – 2p (6*0.33); 5 – 1.5p; 6 – 1.5p, Of – 1p;