Rândul 1

1. Scrieți un subalgoritm pentru a rezolva problema următoare. Specificați complexitatea pentru prima funcție cerută și descrieți în 1-2 propoziții ideea de bază pentru soluție.

Două pisici, numite A și B, stau pe punctele întregi ale axei OX. Pisica A stă pe punctul x și pisica B stă pe punctul y. Ambele pisici aleargă cu aceeași viteză și vor să prindă șoarecele C care este pe punctul z. Având mai multe interogări sub forma (x, y, z) determinați și afișați pentru fiecare interogare, care pisică ajunge prima dată la șoarece (A sau B). Dacă cele 2 pisici ajung în același timp la șoarece, afișați C (cele 2 pisici se ceartă și șoarecele scapă). De ex. dacă avem pozițiile (10, 20, 18), rezultatul este B (A este pe poziția 10, B e pe poziția 20, șoarecele pe poziția 18) și dacă avem (10, 12, 2), rezultatul este A.

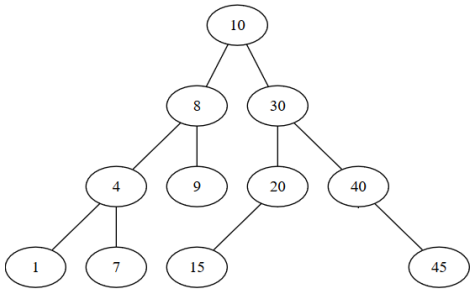
Va trebui să scrieți 2 funcții: una care pentru un triplet de poziții x, y, z (poziția pisicii A, poziția pisicii B, poziția pentru șoarece) determină și returnează cine ajunge prima dată la șoarece (A, B sau C). A 2-a funcție citește pe rând interogările (până se introduc 3 valori egale), și după fiecare interogare citită apelează prima funcție, și afișează rezultatul funcției. Interogările nu vor fi reținute în listă/tablou/vector (tot timpul lucrăm cu interogarea curentă numai).

2. Calculați complexitatea pentru următoarea bucată de cod:

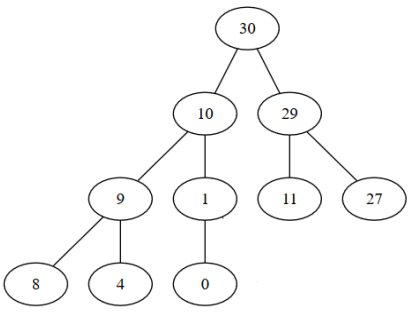
```
Subalgoritm function(n):
        i← n/2
        câttimp i <= n execută
                j ← 1
                câttimp j + n/2 <= n execută
                        k ← 1
                        câttimp k <= n execută
                                k ← k * 2
                        sf_câttimp
                        j ← j + 1
                sf_câttimp
                i ← i + 1
        sf_câttimp
sf_subalgoritm
```

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

a. Cum va arăta arborele binar de căutare dacă adăugăm elementul 17 în el? Cum va arăta rezultatul, dacă după adăugare ștergem elementul 10? Justificați răspunsurile.



b. Avem o TD cu adresare deschisă, verificare pătratică - adică funcție de dispersie d(e, i) = ((e % m) + 1 * i + 3 * i^2) % m – cu 11 locații, numerotate de la 0 la 10. Arătați cum va arăta tabela de dispersie după adăugarea elementelor: 7, 2, 13, 21, 24, 5, 18.



c. Adăugați elementul 33 în ansamblul din dreapta. Arătați pașii intermediari, nu doar răspunsul final.

d. Avem un arbore pentru care cunoaștem parcurgerea în **preordine** și în **inordine**. Parcurgerea în preordine este: X A C B H J K D M Q P iar parcurgerea în inordine este: C A J H K B X M D Q P. Construiți arborele.

4. Alegeți răspunsul corect la următoarele întrebări. Justificați răspunsul ales. La fiecare întrebare, există doar o soluție corectă.

1. Având o stivă inițial vidă, adăugăm pe rând elementele 7, 3, 10, 8 în stivă. Ștergem 3 elemente. Care element rămâne în stivă?

a. 7 b. 3 c. 10 d. 8

2. Cât este complexitatea de timp (caz defavorabil) pentru a adăuga un element într-un vector/tablou ordonat?

a. $\Theta(n^2)$ b. $\Theta(n)$ c. $\Theta(\log_2 n)$ d. $\Theta(1)$

3. Alegeți afirmația corectă:

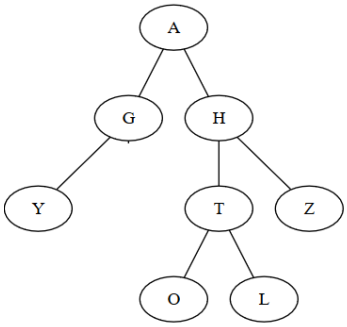
a. Orice arbore binar este un ansamblu
b. Orice arbore binar de căutare este un ansamblu
c. Orice ansamblu este un arbore binar
d. Orice ansamblu este un arbore binar de căutare

4. Avem o tabelă de dispersie cu 1024 de poziții. Dacă folosim pentru rezolvarea coliziunilor adresare deschisă, câte elemente putem insera maximum în tabelă?

a. 1024 b. 512 c. Oricâte d. 1023

5. Care este parcurgerea în preordine pentru arborele următor?

a. A G H Y T Z O L
b. Y G O L T Z H A
c. A G Y H T O L Z
d. Y G A O T L H Z



6. Având un iterator pe o listă simplu înlănțuită, care dintre operațiile iteratorului are complexitate $\Theta(n)$ – unde n este numărul de elemente din listă?

a. valid b. următor c. element
d. toate operațiile e. nicio operație

5. Avem containerul Dicționar reprezentat pe un VectorDinamic. Dați reprezentarea Dicționarului (ce structuri și ce câmpuri sunt folosite). Specificați și implementați operația de ștergere. Cât este complexitatea operației? Arătați reprezentarea pentru Iterator pe Dicționar și implementați la alegere 2 operații de la iterator.

6. Avem containerul Mulțime reprezentat pe o Tabelă de Dispersie cu rezolvarea coliziunilor prin liste independente. Dați reprezentarea Mulțimii (ce structuri și ce câmpuri sunt folosite). Specificați și implementați operația de adăugare. Cât este complexitatea operației?

Punctaj: 1 – 1p; 2 – 0.5p; 3 – 3p (4*0.75); 4 – 3p (6 * 0.5); 5 – 1.5p; 6 – 1p; Of: 1p

Rândul 2

1. **Scrieți un subalgoritm pentru a rezolva problema următoare. Specificați complexitatea pentru prima funcție cerută, și descrieți în 1-2 propoziții ideea de bază pentru soluție.**

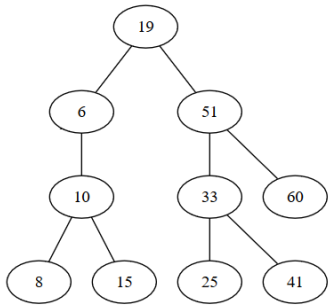
Dan participă într-o competiție de alergare cu obstacole (garduri), unde el va trebui să sară peste n obstacole care au înălțimile h_0, h_1, h_2,…,h_{n-1}. La începutul cursei el poate să sară la o înălțime de k unități. Dan are un stoc nelimitat de poțiune magică, care poate să-l ajute să sară mai sus. Fiecare porție de poțiune băută crește înălțimea la care poate să sară cu 1. Însă, după fiecare săritură el devine mai obosit, și înălțimea la care poate să sară scade cu 1 (indifferent de faptul că a băut poțiune sau nu). Calculați numărul minim de poțiuni de care are nevoie Dan, pentru a termina cursa. De ex. dacă sunt 5 obstacole de înălțime [1, 6, 3, 5, 2] și înălțimea inițială este 4, Dan are nevoie de minim 4 poțiuni (peste obstacolul 1 sare fără probleme, dar înălțimea scade la 3, pentru a sări peste 6 are nevoie de 3 poțiuni. După săritura peste obstacolul cu 6 va avea înălțime 5, deci sare peste obstacolul cu 3 fără probleme, dar înălțimea scade la 4. Pentru a sări peste obstacolul cu 5 are nevoie de o poțiune, după săritură având înălțime 4, deci poate să sară peste obstacolul 2).

Pentru rezolvarea problemei va trebui să scrieți 2 funcții: una, care primește ca parametru numărul de obstacole, un tablou/o listă cu înălțimile obstacolelor și valoarea k (înălțimea inițială). Funcția calculează și returnează numărul minim de poțiuni necesare. A 2-a funcție citește numărul de obstacole, citește înălțimea obstacolelor și valoarea k. După citirea datelor funcția apelează prima funcție și afișează rezultatul ei.

2. **Calculați complexitatea pentru următoarea bucată de cod:**

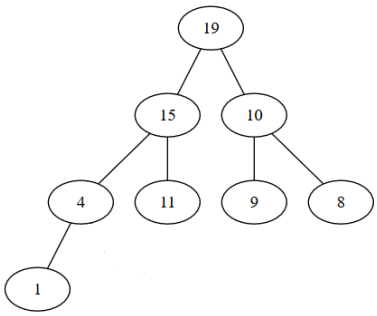
```
Subalgoritm function(n):
        i← n/2
        câttimp i <= n execută
                j ← 1
                câttimp j <= n execută
                        k ← 1
                        câttimp k <= n execută
                                k ← k * 2
                        sf_câttimp
                        j ← j * 2
                sf_câttimp
                i ← i + 1
        sf_câttimp
sf_subalgoritm
```

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



a. Cum va arăta arborele binar de căutare dacă adăugăm elementul 13 în el? Cum va arăta rezultatul, dacă după adăugare ștergem elementul 19? Justificați răspunsurile.

b. Avem o TD cu adresare deschisă, verificare pătratică - adică funcție de dispersie d(e, i) = ((e % m) + 3 * i + 1 * i^2) % m – cu 13 locații, numerotate de la 0 la 12. Arătați cum va arăta tabela de dispersie după adăugarea elementelor: 9, 21, 130, 5, 31, 41, 19.



c. Adăugați elementul 33 în ansamblul din dreapta. Arătați pașii intermediari, nu doar răspunsul final.

d. Presupunem că avem o coadă reprezentată prin 2 stive (adică, avem TAD Stivă implementată, dar nu avem TAD Coadă implementată. Încercăm să simulăm comportamentul unei cozi, folosind 2 variabile de tip stivă). La un moment dat adăugăm în *coadă* elementele 10, 5, 32, 8. Explicați cum arată stiva/stivele și ce se întâmplă dacă vrem să adăugăm elemente în *coadă,* dacă vrem să ștergem sau să accesăm elementul de la începutul cozii. Scrieți complexitatea acestor operații.

4. Alegeți răspunsul corect la următoarele întrebări. Justificați răspunsul ales. La fiecare întrebare, există doar o soluție corectă.

1. Având o coadă inițial vidă, adăugăm pe rând elementele 7, 3, 10, 8 în coadă. Ștergem 3 elemente. Care element rămâne în coadă?

a. 7 b. 3 c. 10 d. 8

2. Avem o tabelă de dispersie cu 1024 poziții. Pentru rezolvarea coliziunilor folosim liste independente. Care este numărul maxim de elemente care pot fi adăugate în tabela de dispersie?

a. 1024 b. 512 c. 1023 d. oricâte

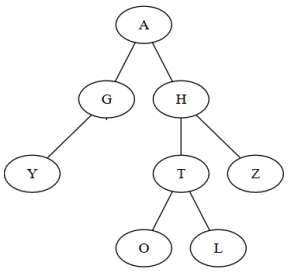
3. Având o stivă reprezentată pe o listă simplu înlănțuită, unde adăugăm un element nou, pentru complexitate cât mai bună?

a. înainte de primul nod b. după primul nod c. înainte de ultimul nod d. după ultimul nod

4. Care dintre următoarele operații are complexitate mai bună pentru un vector dinamic decât pentru o listă dublu înlănțuită?

a. adăugare la început b. adăugare la final c. accesare element de pe o poziție d. căutare un element

5. Care este parcurgerea în postordine pentru arborele următor:



a. A G H Y T Z O L b. Y G O L T Z H A
c. A G Y H T O L Z d. Y G A O T L H Z

6. Putem implementa un ansamblu pe o listă simplu înlănțuită?
a. nu b. da, dar vom avea complexitate mai mare la operații
c. da, și vom avea complexitate identică cu cea pentru un ansamblu pe vector dinamic.

5. Avem containerul Colecție reprezentat pe o Listă Dublu Înlănțuită. Dați reprezentarea Colecției (ce structuri și ce câmpuri sunt folosite). Specificați și implementați operația de ștergere. Cât este complexitatea operației? Arătați reprezentarea pentru Iterator pe Colecție și implementați la alegere 2 operații de la iterator.

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

Punctaj: 1 – 1p; 2 – 0.5p; 3 – 3p (4*0.75); 4 – 3p (6 * 0.5); 5 – 1.5p; 6 – 1p; Of: 1p