Adatszerkezetek – záró dolgozat

Név: _____ Hallgatói azonosító: _____ Dátum: _____

Időtartam: 60 perc · **Elérhető pontszám:** 100 pont · **Segédeszköz:** nem használható.
A pszeudokódban a tömbök indexelése **1-től** indul. A listaelemek mezői: `Elem[P].Érték`, `Elem[P].Köv`; a fák csomópontjainak mezői: `Elem[P].Kulcs`, `Elem[P].Bal`, `Elem[P].Jobb`. Az üres mutató jele **Nil**.

1. feladat – Igaz vagy hamis?

14 × 1 = 14 pont

Jelölje X-szel a helyes választ!

#	Állítás	Igaz	Hamis
1.	A tömb elemei a memóriában folytonosan, egymás után helyezkednek el.		
2.	Az egyszerű láncolt lista *i*-edik eleme O(1) lépésben elérhető.		
3.	A bináris keresés rendezetlen tömbön is mindig helyes eredményt ad.		
4.	A dinamikus adatszerkezetek használata erőforrás-megosztást tesz lehetővé.		
5.	Az üres strázsás lista tárigénye nulla, hiszen nem tárolunk benne elemet.		
6.	A verem LIFO (*last in, first out*) elvű adatszerkezet.		
7.	A ciklikus sorban a kivétel (dequeue) lépésszáma O(n), mert az elemeket előre kell tolni.		
8.	A beszúrásos rendezés egy már rendezett tömbön O(n) lépésben végez.		
9.	A gyorsrendezés legrosszabb esetben $O(n^2)$ lépésszámú.		
10.	Az összefésülő rendezés stabil rendezés.		
11.	A bináris keresőfa mindig kiegyensúlyozott bináris fa.		
12.	A bináris keresőfában való keresés lehet ugyanolyan lassú, mint a láncolt listában való keresés.		
13.	A dinamikus adatszerkezetek helytelen használata memóriaszivárgást eredményezhet.		
14.	Java-ban két `String` tartalmának összehasonlítására a `==` operátort kell használni.		

2. feladat – Feleletválasztás

10 × 1 = 10 pont

Karikázza be az egyetlen helyes választ!

1. Melyik tulajdonsággal **nem** rendelkezhet az egyszerű láncolt lista?

a) szétszórt tárolás b) közvetlen elérés c) homogén d) dinamikus

2. A bináris fa, mint adatszerkezet:

a) heterogén b) közvetlen elérésű c) hierarchikus d) egyik sem

3. A strázsás lista elemei elérhetők:

a) inorder módon b) közvetlen módon c) szekvenciálisan d) egyik sem

4. Melyik adatszerkezet **közvetlen** elérésű?

a) kétirányú láncolt lista b) egydimenziós tömb c) láncolt ábrázolású verem d) ciklikus lista

5. A bináris keresőfa kulcsait növekvő sorrendben adja a(z):

a) preorder bejárás b) inorder bejárás c) postorder bejárás d) szintenkénti bejárás

6. Mi a bináris keresés lépésszáma egy *n* elemű rendezett tömbben?

a) O(1) b) O(log n) c) O(n) d) O(n log n)

7. Egy `int A[6][10]` mátrixot sorfolytonosan tárolunk az 1000-es címtől, 4 bájtos elemekkel, **0-tól** indexelve. Mi az `A[2][3]` elem címe?

a) 1092 b) 1132 c) 1128 d) 1052

8. Melyik rendezés végez legfeljebb *n* – 1 cserét?

a) buborékrendezés b) kiválasztásos rendezés c) beszúrásos rendezés d) gyorsrendezés

9. Mennyi a `5 1 2 + 4 * + 3 -` postfix kifejezés értéke?

a) 14 b) 12 c) 18 d) 20

10. Egy 8 elemű (0–7 indexű) ciklikus sorban `eleje = 6` és `db = 3`. Melyik indexre kerül a következő betett elem?

a) 0 b) 1 c) 9 d) 7

3. feladat – Lépésszámok

10 × 1 = 10 pont

Adja meg a műveletek lépésszámát O-jelöléssel (*n* az elemek száma)!

	Művelet	Lépésszám
a)	Egy tömb *i*-edik elemének elérése	
b)	Lineáris keresés egy *n* elemű tömbben (legrosszabb eset)	
c)	Új elem beszúrása egy *n* elemű rendezett tömbbe úgy, hogy rendezett maradjon	
d)	Beszúrás egy egyszerű láncolt lista elejére	
e)	Törlés egy egyszerű láncolt lista végéről	
f)	Push és pop egy tömbbel megvalósított veremben	
g)	A maximum kivétele egy *n* elemű kupacból	
h)	Keresés egy kiegyensúlyozott bináris keresőfában	
i)	Összefésülő rendezés	
j)	Hanoi tornyai *n* koronggal (a lépések száma)	

4. feladat – Kódkövetés: verem

7 pont

Mit ír ki a következő programrészlet? Általánosságban mit csinál egy tetszőleges szöveggel? (A `Stack` egy karaktereket tároló verem.)

```
String t = "DATA STRUCTURE";
Stack s = new Stack();
for (char c : t.toCharArray()) {
    if (c != ' ') s.push(c);
    else {
        while (!s.isEmpty()) System.out.print(s.pop());
        System.out.print(' ');
    }
}
while (!s.isEmpty()) System.out.print(s.pop());
```

Kimenet	
Általánosan	

5. feladat – Kódkövetés: sor

7 pont

Mit ír ki a következő programrészlet? Általánosságban milyen ismert feladatot old meg, ha a sorba az 1, 2, …, *n* számokat tesszük?

```
Queue q = new Queue();
for (int i = 1; i <= 6; i++) q.enqueue(i);
while (q.size() > 1) {
    q.enqueue(q.dequeue());
    System.out.print(q.dequeue() + " ");
}
System.out.print(q.dequeue());
```

Kimenet	
Általánosan	

6. feladat – Algoritmusok végrehajtása papíron

4 + 4 = 8 pont

a) Rendezze a `[6, 2, 9, 1, 5]` tömböt **beszúrásos rendezéssel**! Írja fel a tömb állapotát a külső ciklus minden lépése után!

b) Keresse meg a `45`-öt **bináris kereséssel** az alábbi tömbben (1-től indexelve)! Írja fel minden lépésben a `bal`, `jobb` és `k` értékét! `A = [4, 9, 13, 21, 30, 38, 45, 52, 60, 71]`

7. feladat – Bináris keresőfa

3 + 3 + 3 + 3 = 12 pont

a) Szúrja be egy üres bináris keresőfába sorban a következő kulcsokat, és rajzolja le a fát!
`45, 20, 70, 10, 30, 60, 80, 25, 35`

b) Adja meg a fa preorder, inorder és postorder bejárását!

	Bejárás
Pre	
In	
Post	

c) Törölje a `20`-as kulcsot az a) pontban kapott fából! Rajzolja le az eredményt, és írja le, melyik törlési esetről van szó!

d) Mennyi az a) pontban kapott fa magassága? Hány kulcs-összehasonlítás kell a `35` megtalálásához?

A 8–10. feladatban egészítse ki a pszeudokódot! A hiányzó részek egy-egy kifejezést, mutatót vagy **relációjelet** (=, ≠, <, >, ≤, ≥) jelentenek. A válaszokat a kód melletti táblázatba írja!

8. feladat – Keresés bináris keresőfában

5 × 2 = 10 pont

A függvény az `x` kulcsú csomópont mutatóját adja vissza, vagy Nil-t, ha nincs ilyen kulcs. A fa gyökerére a **Gyökér** mutató mutat.

```
Függvény Keres(Gyökér, x)
  P ← __(1)__
  Ciklus amíg P __(2)__ Nil és
              Elem[P].Kulcs __(3)__ x
    Ha x __(4)__ Elem[P].Kulcs akkor
      P ← Elem[P].Bal
    különben
      P ← __(5)__
    Elágazás vége
  Ciklus vége
  Vissza: P
Függvény vége
```

	Hiányzó rész
(1)	
(2)	
(3)	
(4)	
(5)	

9. feladat – Strázsás keresés

5 × 2 = 10 pont

A strázsás lista a **Fej** és a **Vég** mutatóval adott; a Vég a lista végén álló strázsa elemre mutat. A függvény az `x` értéket tartalmazó első elem mutatóját adja vissza, vagy Nil-t, ha `x` nincs a listában.

```
Függvény StrázsásKeres(Fej, Vég, x)
  Elem[Vég].Érték ← __(1)__
  P ← __(2)__
  Ciklus amíg Elem[P].Érték __(3)__ x
    P ← __(4)__
  Ciklus vége
  Ha P __(5)__ Vég akkor
    Vissza: Nil
  különben
    Vissza: P
  Elágazás vége
Függvény vége
```

	Hiányzó rész
(1)	
(2)	
(3)	
(4)	
(5)	

10. feladat – Beszúrás egy egyszerű láncolt lista végére

6 × 2 = 12 pont

A lista a **Fej** mutatóval adott. A `Lefoglal(P)` függvény új elemet foglal: siker esetén P a lefoglalt elemre mutat, és a függvény értéke igaz.

```
Függvény VégéreBeszúr(Fej, Adat)
  Ha nem Lefoglal(P) akkor Vissza: hamis
  Elem[P].Érték ← Adat
  Elem[P].Köv ← __(1)__
  Ha Fej __(2)__ Nil akkor
    Fej ← __(3)__
  különben
    Q ← Fej
    Ciklus amíg Elem[Q].Köv __(4)__ Nil
      Q ← __(5)__
    Ciklus vége
    __(6)__ ← P
  Elágazás vége
  Vissza: igaz
Függvény vége
```

	Hiányzó rész
(1)	
(2)	
(3)	
(4)	
(5)	
(6)	

Pontozás

Feladat	1.	2.	3.	4.	5.	6.	7.	8.	9.	10.	Összesen
Elérhető	14	10	10	7	7	8	12	10	10	12	**100**
Elért											

Pontszám	0–49	50–61	62–74	75–86	87–100
Érdemjegy	elégtelen (1)	elégséges (2)	közepes (3)	jó (4)	jeles (5)

Javító: ..... **Érdemjegy:** .....