Gyakorlati jegy – értékelési útmutató

Előkészítés

- A hallgatók gépére másolandó: a `gyakorlati/java/` és a `gyakorlati/go/` mappa (kiinduló kód), valamint a kurzus diái és a `kod/` példaprogramok.
- **Nem** kerülhet a hallgatók gépére: a `gyakorlati/megoldas/` mappa (mintamegoldás Java és Go nyelven).
- Ellenőrizendő a gépeken: `java -version` (17+) és `go version` (1.23+). Az internet-hozzáférést lehetőleg tiltsuk le.
- A kiinduló kód a mintamegoldásból készül: `python3 eszkozok/kiindulo_kod.py <megoldás> <kimenet> java|go` – így az ellenőrző rész garantáltan azonos.

Értékelés menete (hallgatónként kb. 5–7 perc)

1. **Futtatás:** `java StudentRegistry.java` vagy `go run .` – a program kiírja az automatikusan elért szintet.
2. **Gyors kódátnézés:**
   - nem használ-e tiltott gyűjteményt (`ArrayList` , `LinkedList` , `HashMap` , `container/list` , `map` …)?
   - nem módosította-e az ellenőrző részt (a `main` és a `check` függvény)?
   - nem „csal”-e (pl. a `toString` fix szöveget ad vissza, vagy a `find` nem a saját listában keres)?
3. **Szóbeli kérdések:** 2–3 kérdés az alábbi listából, a hallgató szintjének megfelelően.
4. **Jegy:** az automatikus szint, amely **egy jeggyel csökkenthető**, ha a hallgató nem tudja megmagyarázni a saját kódját; súlyos esetben (nyilvánvalóan nem saját munka) elégtelen.

Szóbeli kérdések

1–2. szint

- Mi a lépésszáma az `add` műveletnek? Hogyan lehetne O(1)? *(Vég mutató / tail)*
- Mi történik a `remove` -ban, ha az első elemet töröljük? Miért kell az előző elemet is megjegyezni?
- Mit ad vissza a `find` , ha a lista üres? Miért nem száll el a program?
- Miért O(n) a `find` láncolt listában, és miért nem lehet bináris keresést használni?

3. szint

- Hogyan éri el, hogy az `addSortedByScore` stabil legyen? *(`>=` a feltételben – az egyenlők mögé lép)*
- Mutassa meg a `reverse` működését egy háromelemű listán! Miért kell a `next` mutatót elmenteni?
- Mi a lépésszáma a rendezett beszúrásnak? *(O(n))*

4. szint

- Mit mutat a mérés? Miért gyorsabb a keresőfa? *(O(log n) átlagosan vs O(n))*
- Mekkora lenne a fa magassága, ha az azonosítókat **növekvő** sorrendben szúrnánk be? Mi következne ebből?
- Miért adja az inorder bejárás a rendezett sorrendet?
- Hogyan törölne egy két gyerekkel rendelkező csomópontot? *(inorder rákövetkező)*

Gyakori hibák, amelyekre figyeljünk

Hiba	Tünet az ellenőrzésben
az `add` nem kezeli az üres listát	`NullPointerException` / nil pointer pánik az 1. szinten
a `remove` az első elemnél nem állítja a fejet	„az első elem törlése” HIBA
`best()` a `>=` miatt a későbbi egyenlőt adja	„Best() -> Wang Fang” HIBA
`addSortedByScore` `>` feltétellel – nem stabil	„stabil sorrend” HIBA
a `reverse` új csomópontokat hoz létre	az ellenőrzés átmegy – **szóban** kérdezzük meg!
a BST nem kezeli az ismétlődést	„ismétlődő azonosítót nem szúr be” HIBA

Alternatíva: beadandóként

A feladat **otthoni beadandóként** is kiadható (mint a régi kurzusban): a hallgató egy héttel a 16. hét előtt kapja meg, és a 16. héten **szóban védi** meg. Ilyenkor a szóbeli rész súlya nagyobb (10 perc, legalább 4 kérdés), és a megoldás minden sorát el kell tudni magyarázni.