1. PÉLDA – Anyagmozgatási Rendszer

Fejlesszen alkalmazást, amely szállítópályák óránkénti darabszám teljesítményét egy indexelhető, dinamikusan bővíthető tömb jellegű adatszerkezetben kezeli. Helyezzen el a tárolóban több mérési vagy generált adatot, majd a teljesítményértékeket csökkenő sorrend szerint rendezze. A sorozatban keressen rá egy adott szalag-azonosítóhoz kapcsolódó teljesítményszintre. Zárásként törölje ki azokat a bejegyzéseket, amelyek nem érik el a minimálisan elvárt kapacitásértéket.

Hozzon létre egy függvényobjektumot, amely egy belső hatékonyság-paraméter felhasználásával egy szállítópálya mért teljesítményét átszámítja korrigált teljesítményre, majd hajtsa végre ezt a konverziót a teljes sorozaton.

2. PÉLDA – Pneumatikus Rendszer

Fejlesszen programot, amely egy pneumatikus vezérlőelem állapotkódjainak sorozatát (0 = zárt, 1 = nyitott, 2 = meghibásodás) egy olyan tárolóstruktúrában őrzi, ahol az elemek közötti beszúrási és eltávolítási műveletek gyakoriak, így a *láncolt belső szerkezet* előnyös. Határozza meg, hogy a hibajelző állapot hányszor fordul elő a gyűjteményben. Állítsa növekvő sorrendbe a kódokat, ezt követően fordítsa meg a sorrend irányát, majd egy teljes bejárás során minden kódhoz rendeljen egy érthető szöveges magyarázatot, amit írjon ki.

Konstruáljon egy állapotmegőrző függvényobjektumot, amely a konstruktorban átvesz egy „meghibásodott" állapotkódot, és a meghíváskor bármely kódról eldönti, hogy megegyezik-e ezzel a hibaállapottal; alkalmazza ezt a függvényobjektumot a sorozat átvizsgálása során a hibakódok összeszámolására.

3. PÉLDA – Textilipari Gépsor

Írjon programot, amely szövetsebesség-mérések egymást követő adatpontjait egy olyan tárolóban kezeli, ahol mindkét végponton hatékony a beszúrás és az eltávolítás. Vegye fel a tárolóba több mérési eredményt, majd az első néhány értéket másolja át egy külön adatszerkezetbe további analízis céljából. Azonosítsa a sebesség-sorozat minimális és maximális elemét, végül a sorozat egy kiválasztott szegmensében minden értéket állítson be egy egységes referencia-sebességre (például névleges üzemi értékre), ezzel szimulálva egy kalibrációs műveletet.

Definiáljon egy predikátum függvényt, amely bool visszatérési értékkel jelzi, hogy egy mért sebességérték kilép-e egy meghatározott toleranciasávból, és használja ezt a predikátumot a sorozat végigfuttatásakor a specifikáción kívüli minták kiszűrésére.

4. PÉLDA – Élelmiszeripar Csomagolósor

Valósítson meg programot, amely rendezett, duplikátumokat kizáró halmaz típusú tárolókban nyilvántartja különféle csomagoló berendezések azonosító kódjait: az első gyűjteményben a napi ütemtervben szereplő gépek listáját, a második gyűjteményben pedig a ténylegesen működésben lévő gépek felsorolását. Határozza meg a két halmaz unióját (az összes érintett berendezés), a metszetet (amelyek mind tervezve voltak, mind ténylegesen üzemelnek), valamint a különbségi halmazt (amelyek az ütemtervben szerepeltek, viszont valójában nem működtek).

Készítsen egy lambda kifejezést, amely egy gép-azonosítóból és egy szöveges státusz jelzésből összeállít egy formázott riport sort (például „Gép X: üzemben/nem üzemben"), és alkalmazza ezt a riportgenerálás folyamatában.

5. PÉLDA – Hajóépítési Hegesztőüzem

Készítsen programot, amely egy rendezetlen, duplikátumokat kizáró halmaz struktúrában őrzi az engedélyezett varrat típuskódokat. Két független hajótest-szekció hegesztési sorozatát reprezentálja két különálló szekvenciális tárolóban, majd ellenőrizze, hogy ez a két szekvencia azonos típuskészletet tartalmaz-e (a sorrend figyelmen kívül hagyásával). Az egyik sorozatban keresse meg az első olyan elemet, amely előfordul egy külön definiált problémás kódok jegyzékében, továbbá határozza meg azt a pozíciót, ahol a két szekció hegesztési sorozata először eltérést mutat egymáshoz képest.

Implementáljon egy állapottároló függvényobjektumot, amely a konstruktorban megkapja a „problémás" típuskódok listáját, és a híváskor dönt arról, hogy egy adott varrat típuskód tagja-e ennek a listának; használja fel a sorozat bejárása közben a problematikus varratok számának megállapítására.

6. PÉLDA – Félvezető Gyártósor

Készítsen programot, amely kulcs szerint rendezett asszociatív tárolóban rögzíti a félvezető lapkák (waferek) azonosítóihoz tartozó hibastatisztikai adatokat, mégpedig három különböző gyártási beállítás mellett. Vesse össze az első két leképezést annak eldöntésére, hogy pontosan megegyező kulcs–érték párosításokat tartalmaznak-e. Vizsgálja meg, hogy a harmadik konfiguráció hibanyilvántartása teljes mértékben részhalmazát képezi-e a másik konfigurációénak, végül határozza meg, hogy a kulcsok és értékeik rendezett szekvenciája szerint melyik beállítás számít „kisebbnek", amennyiben az első eltérés balról jobbra haladva döntő.

Konstruáljon egy állapotmegőrző függvényobjektumot, amely a konstruktorban átvesz egy maximálisan megengedhető hibastatisztikai értéket, és a meghíváskor eldönti egy wafer hibaértékéről, hogy az e küszöb alatt vagy felette helyezkedik el; alkalmazza a leképezés feldolgozása közben a selejtezendő tételek azonosítására.

7.Példa – Okos Otthon Energiafelügyelet

Valósítson meg programot, amely egy kulcs–érték párosítást tartalmazó, kulcs szerint nem rendezett asszociatív struktúrában tárolja helyiségek neveire vonatkozó pillanatnyi energiafogyasztási adatokat. Tegye lehetővé, hogy egy konkrét helyiség névhez kapcsolódó összes bejegyzést (például több időpontos mérési rekordot) egyszerre le lehessen olvasni. Vigye át az első néhány rekordot egy külön napló adatszerkezetbe, majd építsen fel egy új perspektívát úgy, hogy ebből a másolatból bizonyos, hibásként megjelölt helyiségek bejegyzései kikerüljenek.

Hozzon létre egy predikátum műveletet, amely egy helyiség energiafogyasztási értékéről eldönti, hogy meghalad-e egy definiált fogyasztási határt, és használja fel ennek megállapítására, hogy mennyi helyiség fogyaszt a megengedett mértéket meghaladó szinten.

8. PÉLDA

Készítsen programot, amely egy online szerverfarm terhelési csúcspontjait tárolja egy dinamikusan bővíthető, index alapján könnyen bejárható tömbszerű tárolóban. Generáljon legalább 20 szimulált CPU-terhelési százalékot, majd rendezze a tárolót úgy, hogy a legnagyobb terhelések kerüljenek az elejére. A sorozatban keressen egy megadott küszöbértéknél nagyobb terhelési értéket, végül törölje a tárolóból az összes olyan mérési pontot, amely egy megengedett maximális terhelést túllép.

Hozzon létre egy predikátumot (bool értéket visszaadó hívható objektumot), amely egy terhelési értékről eldönti, hogy kritikus tartományba esik-e, és használja arra, hogy megállapítsa, van-e a sorozatban ilyen veszélyes csúcs.

9. PÉLDA

Készítsen programot, amely egy láncolt jellegű sorozattárolóban kezeli különböző fémöntvény-minták szakítószilárdsági értékeit. Számolja meg, hány próbatest marad egy megadott minimális szilárdsági küszöb alatt. Rendelje sorba a tároló elemeit növekvő sorrendbe, majd fordítsa meg a sorrendet, és egy bejárás során írja ki minden próbatest minősítését (pl. „hibás”, „közepes”, „kiváló”) a mért szilárdság alapján.

Írjon egy lambda kifejezést, amely egy szilárdsági értékről visszaad egy rövid szöveges kategóriát (pl. „alacsony”, „normál”, „magas”) a megadott küszöbök alapján, és használja a próbatestek kiírásakor.

10. PÉLDA

Készítsen programot, amely egy két végén könnyen bővíthető sorozattárolóban tartja nyilván egy 3D-nyomtató folyamatos hőmérsékletméréseit. Töltse fel a sorozatot adatokkal, majd másolja át az utolsó néhány mérési pontot egy külön tárolóba részletes elemzés céljából. Keresse meg a teljes sorozat legkisebb és legnagyobb hőmérsékletét, végül a sorozat eleje egy meghatározott részét állítsa be egy konstans „bemelegedési” hőmérsékletre.

Írjon egy lambda kifejezést, amely egy hőmérsékletméréshez hozzáad egy rögzített kalibrációs eltérést, és használja arra, hogy a tárolt mérési sorozatból egy kalibrált sorozatot állítson elő.

11. PÉLDA

Készítsen programot, amely rendezett halmaz jellegű tárolókban tárolja egy raktári logisztikai rendszer különböző áruszállítási útvonalainak szakaszazonosítóit (például főútvonal, alternatív útvonal). Állítsa elő azon szakaszok halmazát, amelyek bármelyik vizsgált útvonalon szerepelnek, azon szakaszok halmazát, amelyek mindkét útvonal közös részei, valamint azt a halmazt, amely csak az egyik útvonalhoz tartozó, de a másikban nem szereplő szakaszokat tartalmazza.

Definiáljon egy állapotos függvényobjektumot, amely belső számlálóval követi, hányszor hívták meg, és minden híváskor kiír vagy visszaad egy útvonalszakasz-azonosítót sorszámmal együtt; használja a különböző útvonalak szakaszainak felsorolásához.

12. PÉLDA

Készítsen programot, amely egy hash-alapú, sorrendfüggetlen halmaz jellegű tárolóban tartja nyilván egy ipari vezérlőrendszer engedélyezett biztonsági beállításainak kódjait. Két külön konfigurációs beállítási sorozatot (például eltérő üzemmódokban lefutó folyamatokat) hasonlítson össze: vizsgálja meg, hogy ugyanazok a kódok jelentek-e meg a két sorozatban, keressen meg egy adott sorozatban elsőként olyan kódot, amely egy kritikus kódlistában is szerepel, és határozza meg, hányadik lépésnél kezdődnek az eltérések a két beállítási sorozat között.

Készítsen egy predikátumot, amely egy kódról eldönti, hogy kritikusnak minősül-e (pl. egy külön megadott veszélyes kódlistához tartozik), és használja a konfigurációs sorozatok vizsgálatakor.

13. PÉLDA

Készítsen programot, amely két kulcs szerint rendezett asszociatív tárolóban tartja nyilván különböző energiafogyasztási kategóriák referenciaértékeit két eltérő mérési protokollhoz. Állapítsa meg, hogy a két protokoll pontosan ugyanazokat a kategória–érték párokat tartalmazza-e, továbbá vizsgálja meg, hogy az egyik protokoll minden bejegyzése szerepel-e a másikban is. Végül hasonlítsa össze a két protokollt úgy, hogy a kategórianevek és a hozzájuk tartozó értékek ábécérend szerinti, elemről elemre történő összevetése alapján dönti el, melyik protokoll „előrébb való”.

Készítsen egy predikátumot, amely egy kategória energiafogyasztási értékéről bool formában jelzi, hogy egy minimális elvárt szint alatt marad-e, és használja annak meghatározására, hány ilyen gyenge fogyasztási szintű tétel szerepel a protokollban.

14. PÉLDA

Készítsen programot, amely egy kulcs–érték párokat tartalmazó, gyors kulcsalapú elérésre optimalizált asszociatív tárolóban kezeli különböző szenzorazonosítókhoz tartozó jelkésleltetési mérési eredményeket. Legyen lehetőség egy adott szenzorhoz kapcsolódó összes mérés együttes lekérdezésére (például többszöri mérés esetén). Másolja át az első néhány bejegyzést egy külön jelentés-tárolóba, majd állítson elő egy másik nézetet úgy, hogy a másolatból a megadott kritikus késleltetési szint fölötti mérések kimaradjanak.

Írjon egy lambda kifejezést, amely egy szenzor jelkésleltetési értékéből (pl. milliszekundumban) visszaadja, hogy „kritikus”-nak minősül-e, és használja a problémás szenzorok figyelmeztető listájának összeállításához.