Első zárthelyi dolgozat

A feladatok megoldására 120 perc áll rendelkezésre. Az elkészült megoldást tartalmazó mappát tömörítve, az oktató által megjelölt felületen keresztül nyújtsa be. Ügyeljen a megfelelően feltöltött megoldásra!

Fordítási hibát tartalmazó programkód nem értékelhető!

Szupergépidő. Egy kutatóintézet új szuperszámítógépet vásárolt, amivel az élet értelmére keresik a választ. A kutatók a saját munkacsoportjuknak megfelelő témaszámhoz rendelve küldhetik be számítási igényeiket egy olyan felületen keresztül, amely tárolja és ütemezi a beküldött feladatokat. A feladata egy ilyen felület létrehozása és tesztelése.

A rendszer az adatokat egy táblázatnak képzelhető kétdimenziós tömbbe rögzíti. Ennek soraiban egy-egy rögzített feladat adatai találhatók: rendre egy témaszám (karakterlánc), egy gépidő igény (egész) és prioritás (valós) érték kerül.

Valósítsa meg az alábbi metódusokat!

1. `bool HasSubmittedProjectID(string[,] dataArray, string projectID)`
   Eldönti, hogy a `projectID` témaszámra rögzítettek-e már feladatot a `dataArray`-ben. *(4 pont)*

2. `bool SubmitJob(string[,] dataArray, int row, string projectID, int time, double priority)`
   A `dataArray` tömb `row` indexű sorába rögzíti a `projectID` témaszámú, `time` gépidő igényű, `priority` prioritású feladatot. Visszatérési értékként adja meg, hogy ilyen témaszámra a mostani beküldést megelőzően küldtek-e már be feladatot (lásd `HasSubmittedProjectID()`). *(4 pont)*

3. `string DataArrayToString(string[,] dataArray)`
   A paraméterül kapott kétdimenziós tömb alapján egyetlen karakterláncot épít fel a tömb minden adatából úgy, hogy annak sorait
   `témaszám : gépidő (prioritás)`
   formátumnak megfelelően fűzi össze. A megoldáshoz alkalmazza a sorozatszámítás programozási tétel elvét. *(3 pont)*

4. `int CountSubmittedJobs(string[,] dataArray, string projectID)`
   Megadja, hogy a `dataArray` tömbben lévő feladatok közül a `projectID` témaszámra összesen mennyi gépidőt igényeltek. *(4 pont)*

5. `string[] CollectComplexJobsProjectIDs(string[,] dataArray, int threshold)`
   Egydimenziós tömbbe gyűjti a `dataArray`-ben lévő feladatok közül a `threshold`-nál nagyobb gépigényűek `projectID`-it. A visszaadott tömb mérete egyezzen meg annak valódi elemszámával, azaz ne legyenek benne üres elemek. *(5 pont)*

6. `string SelectMaximalComplexity(string[,] dataArray)`
   Megkeresi és visszaadja a legmagasabb igényelt gépidőhöz tartozó témaszámot. *(4 pont)*

7. `void SortDataArray(string[,] dataArray)`
   Az egyes bejegyzések gépidő × prioritás szorzatának nagysága alapján növekvő sorrendbe rendezi a paraméterben átadott kétdimenziós tömb sorait. A megoldáshoz alkalmazza valamely előadáson ismertetett rendező algoritmust. *(6 pont)*

Az alkalmazás teszteléséhez hozzon létre egy adott $N \geq 10$ feladat tárolására alkalmas kétdimenziós tömböt, majd a `SubmitJob()` segítségével töltse fel véletlenszerűen generált adatokkal. A témaszámok véletlen, `ABCD-1-2`[1] alakú szavak legyenek, a gépidő egy véletlen $[1, 1000]$-beli egész, a prioritás pedig egy véletlen 0.0 és 1.0 közötti valós szám legyen. Ha egy feladatnál az ő témaszámára már van beküldött feladat, írja ki a témaszámhoz tartozó összesített gépidőt[2]. *(8 pont)*

A korábban megírt metódusok alkalmazásával gyűjtse le az 500-nál nagyobb gépidő igényű témaszámokat, adja meg a(z egyik) maximális gépidőt igénylő témaszámot, rendezze a kétdimenziós tömböt, majd írja ki a képernyőre a tartalmát. *(2 pont)*

[1] Négy véletlen betű, kötőjel, egy véletlen szám, kötőjel, egy véletlen szám.

[2] Használja ki a `SubmitJob()` és `CountSubmittedJobs()` visszatérési értékét a megoldáshoz.