FONTOS INFÓK

- A feladat során alkalmazd a megtanult objektum-orientáltsági elveket, figyelj a konstansok és referenciák megfelelő használatára! A javító képes ezeket is ellenőrizni, ami eredményezhet nem forduló kódot. Például, ha egy metódus nem const, pedig annak kellene lennie, lehet, hogy nem fordul majd le a kód.
- A megadott példakódon ne módosíts, hacsak a feladat nem kéri! A megoldásnak ugyanezen fájlokkal kell működnie, hiszen az ellenőrzés során a moodle biztosítja őket.
- Figyelj a kiírás megfelelő formátumára, szóközökre, sortörésekre! A javító csak a tényleges kimenetet látja, nem tudja, mit akartál.
- Ügyelj arra, hogy minden lefoglalt memória kerüljön megfelelően felszabadításra!
- Minden pont értékeléséhez szükséges, hogy az adott ponthoz tartozó define szerepeljen a **megoldott_feladatok.h** header fájlban (#define PART1 az 1. feladathoz, #define PART4a a 4. feladat (a) részéhez stb).
- A header fájlokban csak a header guard-on belülre dolgozz!
- Csak olyan kódot tölts fel, ami nálad fordul. Ami nálad nem fordul, a moodle-ben sem fog.
- A fájlokat nem tömörítve kell feltölteni, hanem önmagukban (drag-and-drop-pal egyszerre be lehet húzni az összeset).
- Figyelj rá, hogy a fájlnevek pontosan azok legyenek, amiket a feladat kér!
- Példakimenet a **main.cpp**-ben kommentben, illetve egy külön **minta-stdout.txt** fájlban található.

Feladat – Minta KisZH 3

- A #define PART<X> direktívák az alapkódban is szereplő **megoldott_feladatok.h** fájlban szerepeljenek! Ez a fájl is beadandó, az **itemcounter.h** és **extra.h** fájlok mellett.

A feladat egy osztály sablont készíteni, amely tetszőleges, **T** típusú elemek előfordulását tudja számolni. **Fontos**: ez egy sablon, ezért minden az **itemcounter.h** fájlba kerül, **itemcounter.cpp** nincs és ne is legyen. Az projekthez adott az Nlohmann JSON könyvtár a **json.hpp** fájl formájában.

Az első feladat kötelező, a 2-4-es feladatok egymástól függetlenül megoldhatók.

1. Legyen egy **ItemCounter** osztály sablon, aminek egyetlen sablonparamétere egy **T** típus, és az alábbi alap funkcionalitással rendelkezik.
   - Tárol egy kezdetben üres **vector<T>** tárolóban **T** típusú elemeket.
   - Tárol egy **vector<int>** tárolóban a **T** típusú elemekhez egy-egy számlálót. Az első tárolóban az i-edik elemhez a második tárolóban az i-edik számláló tartozik, a két tároló mindig ugyanakkora.
   - Konstruktor nem kell.
   - Legyen egy *getIndexOfItem* metódus, ami paraméterben egy **T** értéket kap, és visszaadja az indexét (**int**), amely alatt az adott elem el van tárolva (0-tól elemszám-1-ig). Ha nincs eltárolva az elem, akkor -1-et kell visszaadni.
   - Legyen egy *addItem* metódus, ami paraméterben egy **T** értéket kap, és 1-gyel növeli a hozzá tartozó számlálót. Ha nincs eltárolva az elem, akkor be kell szúrni a tároló végére, és be kell szúrni a számlálót is, ami 1-ről induljon.
   - Legyen egy *getCounter* metódus, ami paraméterben egy **T** értéket kap, és visszaadja a hozzá tartozó számlálót (**int**). Ha nincs eltárolva az elem, akkor 0-t kell visszaadni. **(3 pont)**
2. Legyen az **ItemCounter** osztályban **[]** operátor, ami paraméterben egy **T** értéket kap, és visszaad a hozzá tartozó számlálóra egy módosítható referenciát (**int&**). Ha nincs eltárolva az elem, akkor először be kell szúrni a számlálójával együtt, ami 0-ról induljon.[1] **(2 pont)**
3. Legyen az **ItemCounter** osztályhoz **<<** operátor, amivel tetszőleges kimeneti folyamra kiírható a tartalma. A kiírás formátuma egy JSON tömb legyen, melynek minden elemében legyen **"counter"** és **"item"** kulcs, amihez az adott elem számlálója, illetve maga az elem van rendelve. A JSON legyen 2 szóközönként indentálva. Lásd a példa kimenetet. **(2 pont)**
   Fontos: bár az elvárt operátor osztályon kívüli bináris operátor, de az osztály maga sablon. Ezért ajánlott az osztályon belül, **friend** delkarációval együtt kifejteni azt, lásd a kódrészletet – csak a függvény törzsét kell megírni.
4. Legyen az **extra.h** fájlban egy *getCharFrequency* függvény, ami paraméterben egy fájlnevet kap, és egy **ItemCounter<char>** tárolóban visszaadja a fájlon belül a karakterek előfordulását. A visszaadandó tárolóban az elemek az előforduló karakterek (tetszőleges sorrendben), a hozzájuk tartozó számláló pedig mutatja, hogy a karakter hányszor szerepelt a fájlban. A fájlt szavanként érdemes beolvasni, a whitespace karakterek figyelmen kívül hagyhatók. **(3 pont)**

[1] Ez a feladat megfelel annak, ahogyan az **std::map** asszociatív tároló **[]** operátora is működik.