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 függvény 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 1

- A #define PART<X> direktívák az alapkódban is szereplő **megoldott_feladatok.h** fájlban szerepeljenek! Ez a fájl is beadandó, az **ratedproduct.h/.cpp** fájlok mellett.
- A feladatok között lehet függőség, egymásra épülés.

Adott a **Product** osztály, amely egy eladásra kínált terméket reprezentál, tárolja a termék nevét (**string**) és egységárát (**double**), pár alapvető függvénnyel. A feladat a termékek mellé az **értékeléseket** tárolni. A **Review** osztály reprezentál egy-egy értékelést, amiből a feladat szempontjából csak a csillagok száma (*getStars*) lesz érdekes.

A **Product** és **Review** osztályok kódja nem módosítható!

1. Legyen egy **RatedProduct** osztály a **Product** osztályból származtatva. Konstruktorban három paramétert vár: a termék nevén (**string**) és egységárán (**double**) kívül az értékelések darabszámát (**int**). Ennyi darab **Review** objektumot kell tárolni egy dinamikus tömbben, ügyelve később a felszabadításra. **(2 pont)**
   A feladat kötelező, enélkül a későbbi tesztkódok nem fognak működni.
2. Legyen egy *setReview* metódus a **RatedProduct** osztályban, ami paraméterben egy indexet és egy **Review**-t kap. Tárolja el az adott **Review**-t a tömb megadott indexű helyén. Az index érvényességét nem kell ellenőrizni. **(1 pont)**
3. Legyen a *print* metódus a **RatedProduct** osztályban felülírva. A termék nevén és egységárán kívül írja ki soronként az értékeléseket a megadott formátumban. Egy-egy **Review** kiírásához használható a **Review** osztály saját *print* metódusa. **(2 pont)**
4. Legyen egy *getRating* metódus a **RatedProduct** osztályban, ami visszaadja a tárolt **Review**-k alapján a termék átlagos értékelését (**double**), ami a kapott csillagok számának az átlaga. A csillagok száma egy-egy **Review** objektumból a *getStars* metódussal kérdezhető le. Feltehetjük, hogy van legalább egy **Review**. **(2 pont)**
5. Legyen egy statikus **double** adattag a **RatedProduct** osztályban, amivel egy „elfogadható minimális értékelési szintet” lehet beállítani. Kezdeti értéke legyen 4.2, és a *getAcceptableLevel* és *setAcceptableLevel* statikus metódusokkal lehessen lekérdezni és beállítani. **(1 pont)**
6. Legyen egy *isAcceptable* metódus a **RatedProduct** osztályban, ami visszaadja, hogy a termék értékelése elfogadható-e (**bool**). Ez akkor teljesül, ha az adott termék értékeléseinek az átlaga legalább akkora, mint az előző feladatban említett elfogadható minimális szint. **(1 pont)**
7. Legyen a **RatedProduct** osztály egy **ToolsForRating** nevű névtérben. **(1 pont)**