1. Feladat: Telefonszám Adatbázis és Statisztikai Elemzés

Készíts egy programot, amely telefonszám adatbázist kezel és különböző elemzéseket végez rajta.

Készíts egy `TelefonAdatbazis` osztályt név-telefonszám párokat tároló vektorral. Legyen konstruktor, amely **CSV fájl**ból tölt, **metódus** új telefonszám hozzáadására, és függvény az alapsztatisztikák kiírására.

Implementálj fájl feldolgozást (regex alapú) **telefonszám validációval**. Fogadj el +36 vagy 06 kezdetű, 20/30/70 prefixű számokat. Érvényes számokat **ments külön fájlba**, érvényteleneket **logolj**. Alkalmazz **kivételkezelés**t.

Készíts **template** vagy sima segédfüggvényeket: telefonszám hosszának **minimum és maximum** keresése referenciával, két telefonszám lista összefűzése, névhosszúságok mediánjának számítása. Teszteld telefonszám és név adatokkal.

Implementálj szöveganalízis funkciókat: **nevek formátum ellenőrzése** (nagy kezdőbetű), telefonszámok prefix szerinti **csoportosítás**a, névhosszúságok megszámlálása, bizonyos betűre végződő nevek keresése.

Szimulálj véletlenszerű telefonszám generálást prefix választással (20, 30, 70).
Készíts egyszerű hisztogramot csillagokkal a prefix eloszlásról.

2. Feladat: Film Adatbázis és Értékelés Rendszer

Készíts egy filmes adatbázis-kezelő programot értékelésekkel és statisztikákkal.

Hozz létre egy `Film` osztályt (cím, év, műfaj) és egy `FilmAdatbazis` osztályt, amely filmeket és hozzájuk tartozó értékeléseket 1-10 skálán ***(akár többet)*** tárol. Készíts függvényt új film hozzáadásához és értékelés rögzítéshez.

Olvasd be filmek adatait CSV fájlból és validáld (regex segítségével) az évszámokat (pl. 1900–2024 határok között), ezen felül készíts **kettő tetszőleges validációt** pl. 1-et címekre, 1-et műfajra (***pl. egy meghatározott listából lehetnek műfajok csak***). Hibás adatokat loggolj, érvényeseket ments feldolgozott formátumban. Alkalmazz try-catch blokkokat.

Készíts segédfüggvényeket, legalább egy templatet: értékelések átlagának, minimum és maximum értékének megkeresésére (referenciával), **két filmlista összevonására**, értékelések mediánjának számítására.

Implementálj szövegfeldolgozást: filmcímek elemzése (hány szó, nagybetűvel kezdődnek-e), műfaj szerinti csoportosítás, adott szóra végződő címek keresése, címszosszúság szerinti rendezés.

Szimulálj véletlenszerű értékelés-generálást különböző eloszlásokkal műfajonként. Rajzold ki az értékelések eloszlását egyszerű grafikonon.

3. Feladat: Könyvtári Rendszer és Kölcsönzés Kezelő

Építs fel egy könyvtári nyilvántartó rendszert kölcsönzési funkcionalitással.

Készíts `Konyv` osztályt (szerző, cím, ISBN) és `Konyvtar` osztályt, amely könyveket és kölcsönzési adatokat kezel vektorokban. Legyen metódus könyv hozzáadására, kölcsönzésre és visszavételre.

Dolgozz fel könyv adatokat tartalmazó **.txt fájlt** és validáld az ISBN formátumot (regex-szel) (*10 vagy 13 jegyű számjegyek*). Sikeres beolvasások eredményét ments külön fájlba, hibásokat loggolj részletesen. Kezeld a fájlműveletek kivételeit.

Implementálj segédfüggvényeket, ezekből egy template függvényt it: kölcsönzési idők minimum és maximum értékének keresésére, két könyvlista **vagy könyvtár egyesítésére**, szerzők nevének hosszúságai mediánjának számítására.

Készíts szövegfeldolgozó funkciókat: szerzők nevének validálása (megfelelő formátum), könyvcímek elemzése (szavak száma, nagybetű használat), műfaj alapú csoportosítás, adott betűvel kezdődő címek listázása.

Generálj véletlenszerű kölcsönzési szimulációt (random könyv választás, kölcsönzési időtartam). Készíts statisztikát a legnépszerűbb könyvekről és szerzőkről.

4. Feladat: Online Rendelés és Raktárkezelés

Fejlessz egy egyszerű webshop rendelés és raktárkezelő rendszert.

Hozz létre `Termek` osztályt (név, ár, raktárkészlet) és `Rendeles` osztályt, amely termékeket és mennyiségeket tárol. Készíts `Raktar` osztályt a készletkezeléshez és rendelések feldolgozásához.

Olvass be termék adatokat fájlból és ellenőrizd az árakat (regex-szel) (érvényes pénzösszeg formátum). Sikeres rendeléseket ments külön fájlba, hibásokat loggolj okkal együtt. Alkalmazz kivételkezelést a készlethiány esetére.

Implementálj segédfüggvényeket, legalább egy templatet: árak minimum és maximum keresésére, terméklisták összevonására, rendelési mennyiségek mediánjának számítására.

Készíts szöveganalízis funkciókat: terméknevek validálása (megfelelő formátum), kategória szerinti csoportosítás, adott szóval kezdődő terméknevek keresése, névhosszúságok elemzése.

Szimulálj véletlenszerű vásárlási aktivitást különböző termék kategóriákban. Készíts egyszerű kimutatást a legkeresettebb termékekről grafikusan.

5. Feladat: Egyetemi Kurzus és Hallgató Nyilvántartás

Alkoss egy egyetemi adminisztrációs rendszert kurzusokkal és hallgatókkal.

Készíts `Hallgato` osztályt (név, neptun kód, jegyek vektorral) és `Kurzus` osztályt, amely hallgatókat és jegyeiket kezeli. Legyen funkció hallgató hozzáadásához, jegy rögzítéséhez és statisztika készítéshez.

Dolgozz fel hallgatói adatokat CSV fájlból és validáld a neptun kódokat (regex-szel). Érvényes beiratkozásokat mentsd, hibásokat loggolj részletesen. Kezeld a duplikált beiratkozások kivételét.

Hozz létre template vagy sima függvényeket: jegyek átlagának, minimum és maximum értékének keresésére, két hallgatólista egyesítésére, jegy eloszlások mediánjának számítására.

Implementálj szövegfeldolgozást: hallgatói nevek formátum ellenőrzése, kurzusok neve alapú csoportosítás, adott szakos hallgatók keresése, névhosszúságok szerinti rendezés.

Szimulálj véletlenszerű vizsgaeredményeket különböző tantárgyakból. Készíts egyszerű statisztikai kimutatást a jegyeloszlásról és sikeres/sikertelen arányokról.

Általános tudnivalók a pontozásról

- **Pontozási rendszer:** Minden feladat **25 pontot** ér, és nagyjából **5 részfeladatra** bontható. A maximális pontszám eléréséhez nem szükséges minden részletet tökéletesen megvalósítani.
- **Első feladat prioritása:** Az első feladat a legfontosabb, itt jellemzően **két osztály** implementálása szükséges.

  Itt reflexből az alap CRUD funkciókat érdemes leimplementálni akkor is ha a feladat nem kéri explicit, konstruktor akár túlterhelve, destruktor, getter/setter, kiíratás, stb., lényeg, hogy lássak minél több C++ szintaktikát.

  Másoló konstruktor és értékadó operátor implementálása nem jellemző, de ha valaki megvalósítja, az plusz pontot érhet.

  Kimenetre küldő operátor gyakori feladatrész lesz, ha nem kéri, de megvan akár érhet plusz pontot is.
- **Második osztály implementációja:** Az első feladat második osztályánál az implementáció tetszőleges, de legyen logikus és átgondolt.

  Ajánlott adatszerkezetek:

  - Az első objektumokat tároló `vector`
  - `vector<pair<...,...> >`, ahol az egyik tag az első objektum, a második egyéb adat
  - Két párhuzamos `vector`: az egyikben az első objektumok, a másikban kapcsolódó adatok
  - `vector<tuple<...,...,...> >`, ahol az egyik tag az első objektum, a további tagok egyéb adatokat tárolnak
  - Beágyazott `pair` struktúra:
    `vector<pair<objektum1, pair<...,...> > >`

  Megjegyzés: Természetesen használhatók ettől eltérő adatszerkezetek is (akár C stílusúak), a lényeg, hogy az indexelés ne legyen túlságosan bonyolult és a megoldás logikus legyen.
- **Fájlkezelés:** A feladatok 50-50%-ban tartalmaznak **CSV** és **TXT** fájlkezelést. A tesztfájlokat a maguknak kell létrehoznunk.

  Követelmény: A programban minden funkciót le lehessen tesztelni a fájlokban szereplő adatokkal. A **validációs** mechanizmusok teszteléséhez szándékosan **hibás sorokat** is tartalmazniuk kell a fájloknak.
- **3-4. feladatok implementációja:** A funkciókat meg lehet valósítani külön függvényként, metódusként vagy **barát** (`friend`) függvényként. A cél a modern C++ szintaktikák bemutatása (pl. `template`, `auto`, **iterátorok**, **range-based for** ciklus stb.), de klasszikus C stílusú megoldások is elfogadottak.
- **5. feladat:** Az ötödik feladat mindig **véletlenszerű adatgenerálást** és esetleges **szimulációt** tartalmaz.