1. feladat – Telefonminta ellenőrzése

Készíts egy `TelefonKezeles` osztályt, benne egy

```
static bool ervenyes_telefonszam(const std::string& s);
```

metódussal, ami elfogadja például:

```
+36 20 1234567 06-30-765-4321 +36(70)5554444
```

Teendők:

- Írd meg a regex-et (országkód +36 vagy 06, prefix 20|30|70, opcionális szóköz/kötőjel, 7 számjegy).
- Kommenteld soronként a regex minden részét.
- A `main`-ben próbálj ki 3 jó és 3 rossz mintát, `cout`-tal jelezd, melyik illeszkedik.

2. feladat – Prefix + hívószám bontása

Implementálj egy

```
struct Telefon  std::string orszagkod, prefix, hivoszam; ; static Telefon
bontott_telefonszam(const std::string& s);
```

függvényt, ami valid telefonból kiválasztja a három részt.

Teendők:

- Használj regex-csoportokat vagy `find_first_of` + `substr` technikát.
- Hibás bemenetre dobj `std::invalid_argument`-ot.
- A `main`-ben strukturált kötésben mutasd be:

```
auto t = bontott_telefonszam("+36 20 1234567"); auto [ok, pr, hv] = t; cout « ok
« " " « pr « " " « hv « endl;
```

3. feladat – CSV-szűrés és fájlba írás

Adott egy `contacts.csv`:

```
Név;Telefonszám Kiss Ádám;+36 20 1234567 Nagy Béla;06-20-765-4321 Tóth
Éva;+36(1)2345678
```

Írj egy `CsvTelefonFeldolgozo` osztályt, ami:

- beolvassa a CSV-t soronként,
- ha `ervenyes_telefonszam` → beírja `valid_telefonok.csv`-ba,
- különben `hibas_telefonok.log`-ba (sorszám + név + szám).

Nehézítés: Minden sor köré tegyél `try/catch`-et, hogy egy hiba ne állítsa le az egész feldolgozást.

4. feladat – Konfigurálható prefix-lista

Készíts egy `telefon.cfg` fájlt, amiben soronként egy engedélyezett prefix szerepel (pl. 20, 30, 70). A `CsvTelefonFeldolgozo` konstruktorában töltsd be ezt `vector<string>`-be, és csak ezekkel a prefixekkel fogadd el a számokat.

Kimenet:

- Tiltott prefixű számokat írd `tiltott_prefix.log`-ba.
- A konzolra írd ki: "engedélyezett: X db, tiltott: Y db."

5. feladat – Record (szöveges megfogalmazás)

Készíts egy `Record` nevű struktúrát és egy `RecordLista` nevű osztályt az alábbi célokra:

1. `Record` struktúra

- Legyen benne két tag: `int id;`, `std::string nev;`

2. `RecordLista` osztály

- Legyen egy privát `std::vector<Record> lista;` tagváltozója.
- A konstruktorában (`RecordLista(const std::string& fajlnev)`) nyissa meg az `adatok.txt` nevű szöveges fájlt (`std::ifstream`).
- Ha a fájl nem nyitható meg, dobjon `std::runtime_error` kivételt.
- Olvassa be soronként az `int id` és `std::string nev` párokat (pl. `in » id » nev;`), amíg van mit beolvasni.
- Minden beolvasott párral hozzon létre egy `Record{id, nev}` objektumot, és tolja be a `lista` vektorba.

3. `kiir()` metódus

- Írjon ki minden egyes `Record`-ot a `lista`-ból a konzolra ebben a formában:

```
ID: <id>, Név:  <nev>
```

4. `main()` függvény

- Próbálja meg létrehozni a `RecordLista`-t `"adatok.txt"`-ből.
- Ha kivétel (pl. nem olvasható a fájl), fogja el, és írja ki a hibaüzenetet (`e.what()`) a standard hibára.
- Ha sikeres, hívja meg a `kiir()`-t.