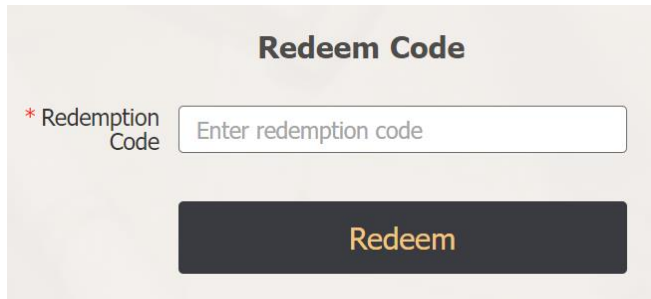


PARCIÁLIS II. - KUPONBEVÁLTÁS



Redeem Code

* Redemption Code

Redeem

Vásárlások során bizonyára lehetőségek volt kuponokat beváltani bizonyos ajándékért/kedvezményekért. A feladatod egy hasonló kuponbeváltó rendszer megvalósítása.

A felhasználók kuponkódokat fognak beváltani. Minden kupon használata limitált (csak az első n felhasználó tudja beváltani azt). Minden kupon

rendelkezik egy rövid leírással is, amely tájékoztatja a felhasználót, hogy mit nyert. Amikor valaki felhasznál egy kupont, a kuponnak csökken a még felhasználható alkalmak száma - tehát a kupon eggyel kevesebbszer használható fel. A kupon leírása egy mondat.

Példa bemeneti állományra:

```
COUPON2024A    100
10% kedvezmeny minden online vasarlasra
SAVE10        50
5000 Ft kedvezmeny 20 000 Ft feletti vasarlasokra
HAPPYNEWYEAR   200
Ingyenes szallitas az elso 3 rendelésre
SUMMERJOY20    150
20% kedvezmeny minden nyari kollekcios termekre
HOMEDECOR10    80
10% kedvezmeny otthoni dekoracios termekekre
TRAVEL15       60
15% kedvezmeny utazasi csomagokra
GENSHINGIFT    200
10% kedvezmeny minden termekre
KITCHENREADY   90
20% kedvezmeny minden konyhai keszulekre
```

(1p) Hozz létre egy új osztályt (**CouponCode**), amely a kuponkódot képviseli. Tartalmazzon minden olyan adatot, amely releváns a kuponkód modellezéséhez. Az adatokat a memóriában tároljuk!

- **Ötletek:** a kuponkód neve, a kuponkód leírása (mondat), hányszor használható még fel a kuponkód
- **Ajánlás:** Nem szükséges letárolni, hogy hányszor használható fel a kuponkód ÉS hogy hányszor használták fel: elég, ha letároljuk, hogy hány használata maradt a kuponnak

(1p) Az osztályban hozd létre a következő függvények leírását:

1. Legyen egy függvény, amely visszatéríti a kuponkód nevét.

2. Legyen egy függvény, amely ellenőrzi, hogy felhasználható-e még a kuponkód (több, mint 0 felhasználás maradt a kódon).
3. Legyen egy függvény, amely felhasználja a kuponkódot egyszer, **visszatérítve** a kuponkód leírását (egy mondatot) - vagy kivételt dobva ha már nem lehet felhasználni.

(2p) Implementáld az osztály függvényeit!

- **Vigyázat!** Ha a kuponkódot már nem lehet felhasználni, ne engedjük azt még egyszer kiváltani! Ebben az esetben dobjunk kivételt: **std::logic_error (1p)**

(1p) Hozz létre egy segédfüggvényt, amely létrehoz egy 10 alfanumerikus karakterből álló véletlenszerűen generált kupont egyszeri felhasználásra (például: **NA88ANTJL5**). További megkötés, hogy a kód csak nagybetűket és számokat tartalmazhat. Bármilyen stratégiát használhatsz, amennyiben elegendően véletlenszerű kuponkódokat generál!

- **Ajánlás:** használd a C++ által nyújtott véletlenszerű számgenerálást:

```
std::random_device dev;
std::mt19937 rng(dev());
std::uniform_int_distribution<int> numbers(0, 25); // [0, 25] tartományból
generál számokat

int number = numbers(rng);
char letter = 'A' + number; // ASCII kóddá alakítás
```

(1p) Hozz létre egy interfészt (**IRedeemService**) (C++-ban: absztrakt osztály tiszta virtuális függvényekkel), amely leírja a kuponbeváltó szolgáltatás funkcionálisait:

1. Legyen képes egy kuponkódot felhasználni, amennyiben megadjuk paraméterként a kuponkód nevét. Térítse vissza a kuponkód leírását (mondat), vagy dobjon kivételt ha nem létezik a kuponkód, vagy ha nem lehet már felhasználni!
2. Legyen képes létrehozni egy egyszeri használatú kupont. Használd az előző alponthan implementált segédfüggvényt a kuponok nevének véletlenszerű generálására!
3. Legyen képes megszámolni, hogy hány olyan kuponkód létezik, amely még felhasználható. Törekedj felhasználni a C++ standard könyvtárát az elemek megszámolására!
4. Legyen egy bemeneti állományból kiolvasni az összes kuponkódot a memóriába, amennyiben megadjuk a bemeneti állomány nevét. Használd a feladathoz tartozó példa bemeneti állományt.

(2p) Implementáld az előző alponthan kidolgozott interfészt egy **RedeemService** osztály keretén belül!

(1p) Teszteld le alaposan a megírt program összes függvényét a **main.cpp** állományban.