LINQ feladatok - 1

1. Tekintsük a következő, egész számokból álló listát:

   ```
   {1,-3,5,4,-2,-1,0,10,-10}
   ```

   Az alábbi feladatokban mindig készítsünk egy új gyűjteményt, és a tartalmát írassuk is ki!

   a) Állítsuk elő a számok négyzeteit **lekérdező kifejezés formátum** (*Query Expression Format*) segítségével! (select: kiválasztás)

   b) Oldjuk meg az előző kiválasztást **kiterjesztés formátum** (*Extension Format*) **és lambda kifejezés** segítségével! (Ehhez ún. **kiterjesztés metódus**, másképpen **bővítő metódus** kell, vagyis a Select.)

      b.1) Oldjuk meg a *Func* delegált használatával is!

   c) Válasszuk ki a számok négyzetei közül csak a különbözőket kiterjesztés metódust használva!

   ---

   d) Állítsuk elő a negatív számokat lekérdező kifejezés formátum segítségével! (Azokat választjuk ki, amelyek eleget tesznek valamilyen feltételnek: ez ún. szűrés.)

   e) Oldjuk meg az előző szűrést kiterjesztés metódus (Where) segítségével!

   f) Szűrjük ki a páros indexű elemeket kiterjesztés metódus segítségével!

   g) Szűrjük ki az 1-nél nagyobb számokat, és állítsuk elő a 10-szereseiket lekérdező kifejezés formátum segítségével! Figyeljük meg a késleltetett végrehajtást pl. úgy, hogy a kiíratás foreach ciklus előtt még adjuk hozzá a listához a 80-at!

2. Készítsünk egy *Person* nevű osztályt a következő tulajdonságokkal:

   ```
   public int ID {get; set;}
   public string FirstName {get; set;}
   public string LastName {get; set;}
   public string PhoneNumber {get; set;}
   ```

   Ezután készítsünk egy listát a következő objektumokkal:

   ```
   Person(1,"Lajos","Kis","123-4567"),
   Person(2,"Béla","Nagy","234-5678"),
   Person(3,"Miklós","Szabó","345-6789"),
   Person(4,"Gyula","Kovács","111-2222"),
   Person(5,"Jenő","Cipész","222-3333")
   ```

   Állítsuk elő a páratlan ID-jű személyek teljes nevét és telefonszámát egy új gyűjteménybe! A megoldáshoz lekérdező kifejezés formátumot és névtelen osztályt használjunk! (Ez projekció és szűrés, mert csak bizonyos adatokat választunk ki, és szűrőfeltételünk is van.)

3. Tekintsük a következő, karakterláncokból álló, *names* nevű listát:

```
{"Lajos","Béla","Miklós","Gyula","Jenő","Kálmán","Attila","Alajos",
"Elemér","Edina", "Ede", "Márton"}
```

Az alábbi feladatokban (kivéve az e)-t) mindig készítsünk egy új gyűjteményt, és a tartalmát írassuk is ki!

a) Rendezzük növekvőleg (azaz ábécé szerint) egy új gyűjteménybe a neveket, lekérdező kifejezés formátum segítségével! (orderby: rendezés)

a.1) Próbáljuk ki a csökkenő sorrendet is! (descending)

b) Rendezzünk növekvőleg kiterjesztés metódussal! (OrderBy)

b.1) Próbáljuk ki a csökkenő sorrendet is! (OrderByDescending)

b.2) Próbáljuk ki a növekvő sorrend megfordítását is! (OrderBy+Reverse)

c) Rendezzünk növekvőleg először a kezdőbetű, majd azon belül a 2. betű alapján lekérdező kifejezés formátum segítségével! Mi történik, ha van *null* elem a listában? Javítsuk a kódot!

d) Rendezzünk növekvőleg kiterjesztés metódusokkal! (OrderBy+ThenBy)

e) Rendezzük a `{4,6,3,1,2,11,13}` listát 5-tel való osztási maradék alapján növekvő sorrendbe kiterjesztés metódussal! (Sort. Szükség lesz a CompareTo metódusra is.)

---

f) Csoportosítsuk (új gyűjteménybe) a kezdőbetű szerint növekvőleg rendezett neveket a kezdőbetűjük szerint, lekérdező kifejezés formátum segítségével! (group: csoportosítás)

g) Csoportosítsunk kiterjesztés metódusokkal, annyi eltéréssel az előzőhöz képest, hogy most a csoporton belül is legyen ábécé sorrend! (OrderBy+ThenBy+GroupBy)