# Feladat

Készítsen egy halmaz típust! A halmazt rendezett láncolt listával ábrázolja! Implementálja a szokásos műveleteket (elem betétele, kivétele, benne van-e egy adott elem, üres-e), egészítse ki az osztályt a halmaz tartalmát kiíró operátor << -ral! Definiáljon olyan barát-operátorokat is, amely kiszámítja két halmaz szimmetrikus differenciáját és metszetét! A metszet műveletigénye: Ơ(m+n), ahol m és n a két halmaz elemszáma.

# Lista típus

A feladatot egy láncolt listával fogjuk ábrázolni. Ez a lista fejelem nélküli, egyirányú láncolt lista lesz. Egy elem értékei: – az elemben tárolt érték, – a következő listaelem. Ebben a listában az elemek mindig növekvő sorrendben helyezkednek el, a könnyű beszúrás és keresés érdekében.

Amennyiben a halmaz üres, úgy az első elemet jelölő mutató 0.

## Implementáció

### IsEmpty

Ez a függvény meghatározza, hogy a halmaz üres e. Itt csak annyit kell csinálni, hogy megnézzük, hogy az első elemre mutató mutató egy nulla mutató e.

### Beszúrás

Új elem beszúrásánál meg kell keresni az elem helyét. Két eset lehetséges: Első esetben még üres a lista, vagy a lista első eleme nagyobb, mint a baszúrni kívánt elem. Ezt a két lehetőséget azért veszem egy esetbe, mert ugyanazt kell csinálni: Új elemet az első helyre, és akármi volt az első elem, arra mutasson az új első elem. (Ha nullptr volt eddig, akkor a next nullptr lesz, ami helyes, ha pedig igazi elem volt, akkor az az elem lesz a next)

A második esetben egy belső elemhez kell helyezni. Ekkor elkezdjük bejárni a listát, és ha megtaláltuk az első olyan elemet, amely nagyobb, mint a beszúrni kívánt elem, az elé kell beszúrnunk. Ehhez folyamatosan tárolunk egy előző elem pointert is, mivel visszafele lépni nem tudunk.

### Keresés

Elem keresésénél csak elkezdjük bejárni a listát, és akkor állunk meg, ha megtaláltuk a keresett elemet, vagy találunk egy olyan elemet, amely nagyobb, mint a keresett elem.

### Törlés

Törlésnél, mint beszúrásnál, szintén két eset lehetséges: Ha az első elem a törlendő elem, vagy egy belső elem. A beszúráshoz hasonlóan járunk el, csak töröljük a keresett elemet, és nem beszúrjuk.

### Kiíró operátor

Kiírás esetén először egy „Set:” nyitással kezdünk, majd ezután felsoroljuk az elemeket. A legvégére kapcsos zárójelek között megjelenítjük a set elemeinek számát.

### Metszet

Metszet esetén két set -en kell végig mennünk, és az egyező elemeket kigyűjteni egy új setbe. Szerencsére tudjuk, hogy mindkét set elemei növekvő sorrendben helyezkednek el a listájukban.



### Szimmetrikus differencia



# Megoldás

## Set osztály

A Set osztály egy olyan osztály, amely kezeli a láncolt listára vonatkozó műveleteket. Az osztály deklarációját a set.h -ban helyezzük el, a metódusok implementációját pedig a set.cpp forrásban.

# Tesztelési terv

1. Üresség tesztje
   1. Kezdetben üres e
   2. Egy elem beszúrva üres e
   3. Az elem kivétele után üres e
2. Keresés & beszúrás & törlés tesztje
   1. Kezdetben ne találjon semmit
   2. Beszúrjuk az elemet, utána megtalálja e
   3. Kivesszük az elemet, most megtalálja e
3. Tisztítás tesztje
   1. Beszúrunk több elemet
   2. Tisztítjuk a listát
   3. Ezután talál e valamit?
4. Metszet
   1. Beszúrunk elemeket
      1. 2-vel osztható számok az egyik setben
      2. 3-al osztható számok a másik serben
   2. Metszetet számítunk
   3. A végső listában helyes metszet esetén csak 6-al osztható számok maradhatnak
5. Szimmetrikus differencia
   1. Beszúrunk elemeket (a metszethez hasonlóan)
   2. Szimmetrikus differenciát számítunk
   3. A végső listában helyes szimdiff esetén csak ((3-al osztható) XOR (2-vel osztható)) számok maradhatnak.
6. Másolás tesztje
   1. Beszúrunk elemeket
   2. Lemásoljuk a listát
   3. Megegyeznek az elemek?
7. Ön-Másolás tesztje
   1. Beszúrunk elemeket
   2. Saját magába másoljuk a listát
   3. Történt változás?