4. heti feladatok

1. órai feladat Készítsen egy alkalmazást, ami képes megmondani egy számról, hogy az prím-e, majd tesztelje is ezt.

- Hozza létre a **PrimeTool** osztályt.
  - Az osztályban tárolja el, hogy mely számról lehessen megállapítani a prím tulajdonságot.
  - A számot a konstruktorban lehessen beállítani.
  - Legyen az osztályban egy `bool IsPrime()` metódus, ami meghatározza, hogy prím-e a szám.
- Írjon teszteket, amivel ellenőrzi, hogy minden lényegesen különböző esetben helyesen működik-e az osztálya.

2. órai feladat Írjon egy osztályt, ami különböző statisztikákat képes megadni egészeket tartalmazó tömbről.

- Az **ArrayStatistics** osztályban tároljon el egy egészeket tartalmazó tömböt. A tömb az osztály konstruktorán keresztül legyen megadható.
  - Legyen az osztálynak egy `int Total()` metódusa, ami visszaadja, hogy mennyi a tömbben lévő elemek összege.
  - Legyen az osztálynak egy `bool Contains(int number)` metódusa, ami megadja, hogy a keresett szám benne van-e a tömbben.
  - Legyen az osztálynak egy `bool Sorted()` metódusa, ami pontosan akkor tér vissza `true` értékkel, ha a tömb elemei növekvő módon rendezettek.
  - Legyen az osztálynak egy `int FirstGreater(int value)` metódusa, ami visszatér az első olyan tömbbeli elem indexével, amelynek értéke nagyobb a paraméterként átadott értéknél. Ha nincs ilyen elem a tömbben, akkor -1-gyel térjen vissza a metódus.
  - Legyen az osztálynak egy `int CountEvens()` metódusa, ami meghatározza, hogy hány páros szám van a tömbben.
  - Legyen az osztálynak egy `int MaxIndex()` metódusa, ami megadja a tömb legnagyobb elemének indexét.
  - Legyen az osztálynak egy `void Sort()` metódusa, ami valamely eddig megismert rendező algoritmus[1] segítségével rendezi a tömböt.
- Írjon teszteket az egyes metódusokhoz, lehetőség szerint minden határesetet tesztelve.

3. gyakorló feladat Valósítson meg egy ún. **Verem**[2] adatszerkezetet, majd tesztelje a megvalósítását.

- Hozza létre a **Stack** osztályt az alábbiak figyelembe vételével.
  - Legyen egy karaktereket tartalmazó tömbje, aminek a maximális elemszámát a konstruktorban lehet beállítani.
  - Legyen egy adattagja, ami a veremben lévő elemek számát tárolja.
  - Lehessen betenni egy új elemet a verem tetejére a `bool Push(char newItem)`. Ha a betevés sikertelen, mert a verem már tele van, akkor a függvény visszatérési értéke legyen `false`.

[1] Minimumkiválasztásos rendezés, buborékrendezés, beillesztéses rendezés

[2] Egy veremnek csak a tetejére lehet tenni új elemet és csak a tetejéről lehet kivenni elemet.

  - Lehessen kivenni egy elemet a verem tetejéről a `bool Pop(out char item)` metódus használatával. Ha üres a verem, akkor a függvény visszatérési értéke legyen `false`. A kivett elem a kimeneti paraméteren keresztül legyen elérhető.
  - A `bool Empty()` metódussal lehessen lekérdezni, hogy üres-e a verem.
  - A `bool Full()` metódussal lehessen lekérdezni, hogy tele van-e a verem.
- Írjon teszteket az összes megvalósított metódushoz. Ügyeljen arra, hogy minden lényeges esetet fedjen le tesztekkel.