1. feladat: Írjon egy programot, amely 2 db felhasználói függvényt használ. Az egyik az **int beker(int be_tb[], int *min, int *max)**, amely 10 db előjeles egész számot kér be úgy, hogy azokat egy tömbbe tárolja le.A tömb hosszát szimbolikus állandóval állítsa be. A tárolás csak akkor jöjjön létre, ha az alábbi feltételek teljesülnek. A tömb **0. indexén** csak a **nulla** érték, a páratlan indexein csak páratlan értékek és a *páros indexein* csak *páros* számok szerepelhetnek! Ha az adott érték nem teljesíti a rá vonatkozó feltételt, akkor új számot kell bekérni. Bekérési információk és hibaüzenetek kiírása nem kell! A feltételek teljesülése esetén a **beker** függvény határozza meg és tárolja el a legkisebb páratlan, illetve a legnagyobb páros számokat. Ezen értékek a függvény 2. és 3. paraméterei, amíg az 1. paraméter a tároló tömb. A függvény visszatérési értéke a 10 db szám összege legyen. A másik függvény a **void kiir(int szum, int min, int max)**, pedig írja ki a **beker** függvény visszatérési értékét illetve a 2. és 3. paraméterek értékeit a példában látott szövegezéssel.

2. Írjon egy programot, amely 2 db felhasználói függvényt használ. Az egyik az **int feltolt(char s_tb[])**, amely feltölt egy karakter tömböt (1. paraméter) ékezet nélküli szöveggel az ENTER billentyű leütéséig, vagy maximum 50 db karakterig. Bekérési információk és hibaüzenetek kiírása nem kell! A tömb hosszát szimbolikus állandóval állítsa be. A függvény visszatérési értéke a feltöltött tömb hossza legyen. A másik függvény a **void torol(char s_tb[], int h)**, pedig a feltöltött tömb (1. paraméter) minden második elemét törölje, majd az így módosított tömb tartalmát írja is ki. A **torol** függvény 2. paramétere a **feltolt** függvény visszatérési értéke.

3. Írjon egy **string tri_result(string fbe)** függvényt, amely kiírja egy adott triatlon verseny győztesének az azonosítóját (licence), és a cél-idejét; óra:perc:másodperc alakban. A cél-idő a következő részeredményekből tevődik össze: úszás + kerékpár + futás + depo. A kiírás pontos formátumát a példa mutatja! A célba érkezett versenyzők számát és a részidőket egy adat-file tartalmazza, amelynek a létezését ellenőrizni kell! A hibaüzenet formátumát a példa mutatja! Ennek az állománynak az azonosítója lesz a **tri_result** függvény paramétere. Az adat-file első sora egy pozitív egész szám, amely a célba érkezettek száma. A további mindenegyes sora egy-egy sportolót azonosít és tartalmazza a részidejeit másodpercben, a következők szerint:
   licence úszás-idő kerékpár-idő futás-idő depó-idő.
   Az adatokat szóközök választják el egymástól! Pl.:
   df-572ki 1500 4500 2500 125
   Az adat-file elemeit tárolja el egy struktúra-tömbbe, amihez használja a megadott **struct triathlete** típust! A struktúra-tömböt a dinamikus memóriába hozza létre! A versenyzők kiszámolt cél-idejeit (másodpercben) az **int sum** tagváltozókba mentsék el. Ezen adatokból kell megállapítani a győztes és az utolsó versenyző idejeit. Ugyanis a függvény visszatérési értéke az utolsó versenyző licence legyen. Ha az adat-file nem létezik, akkor a visszatérési érték az ”N/A!” karaktersorozat legyen! (A **cerr <<** utasítást NE használják! moodle…)

4. Írjon egy **double area(string fbe)** függvényt, amely kiírja az alábbi feltételeknek megfelelő általános háromszögek területeit. A kiírás pontos formátumát a példa mutatja! A 10 db háromszög adatait egy adat-file tartalmazza, amelynek a létezését ellenőrizni kell! A hibaüzenet formátumát a példa mutatja! Ennek az állománynak az azonosítója lesz az **area** függvény paramétere. Az adat-file egy-egy sora a következő adatokat tartalmazza:
   1._oldal 2._oldal a_2_oldal_által_bezárt_hegyesszög.
   Az oldalak mértékegysége méter, a szögé fok. Az adatokat szóközök választják el egymástól! Pl.:
   140 230 40
   Az adat-file soronkénti elemeit és a kiszámított területek értékeit tárolja el egy-egy dinamikus tömbbe, azaz 4 db tömböt kell szinkronban tartania. A tömbök hosszát szimbolikus állandóval állítsa be. Ebben az esetben a trigonometrikus terület-képletet kell használni, amely a következő: (1._oldal * 2._oldal * sin(a_2_oldal_által_bezárt_hegyesszög)) / 2
   A terület tizedespontosságának a meghatározását bízzák a fordítóra. Kiíratni csak azon területeket kell, amelyek a 2.000-től 8.000-ig tartó zárt intervallumba esnek. Azaz a határértékek is megengedettek. A függvény visszatérési értéke, a fenti feltételeknek megfelelő háromszög-területek számtani átlaga. Ha az adat-file nem létezik, akkor a visszatérési érték a ”-1” legyen! (A **cerr <<** utasítást NE használják! moodle…)

5. Írjon egy **string first_last(string fbe, string fki)** függvényt, amely kiírja az alábbi átalakításon átesett szerzők neveit a képernyőre és egy kimeneti file-ba (2. paraméter). A kimeneti file létrejöttét is ellenőrizni kell! A kiírások pontos formátumát a példa mutatja! A szerzők keresztneveit (több is lehet) és vezetéknevét egy adat-file tartalmazza, amelynek a létezését ellenőrizni kell! A hibaüzenet formátumát a példa mutatja! Ennek az állománynak az azonosítója lesz az **first_last** függvény 1. paramétere. DE! Az adat-file első sorában csak egy pozitív egész szám található, amely az írók számát adja meg, amíg a további sorok egy-egy szerzőt azonosítanak. Az adat-file egy-egy sora a következő adatokat tartalmazza:
   1._keresztnév 2._keresztnév … n._keresztnév vezetéknév
   A neveket szóközök választják el egymástól! Pl.:
   Douglas Noel Adams
   Az adat-file szerző-sorait tárolja el egy dinamikus 2 dimenziós tömbbe.Ezek után dolgozza fel ezen tömböt úgy, hogy a szerzők nevei a következő mintát kövessék, mind a képernyőn, mind a kimeneti file-ban:
   vezetéknév, 1._keresztnév 2._keresztnév … n._keresztnév
   Azaz a vezetéknév kerüljön előre, majd a vessző utáni szóköz után jöjjenek a keresztnevek szóközökkel elválasztva. (A kiírásokat úgy is megvalósíthatja, hogy az eredeti tömböt nem kell felüldefiniálnia, hanem segéd-dinamikus tömböket hoz létre!) A függvény visszatérési értéke, a ”Hibátlan futás!” illetve, ha az adat-file nem létezik, akkor a ”Sikertelen file-nyitás!” szöveg-információ legyen!