*Az alábbi feladatok megoldásához az előadáson bevezetett **osztálykönyvtárat kell használnia**. Az osztály-sablonok kódja megtalálható a http://people.inf.elte.hu/gt/oaf/lib.zip állományban. A megoldásokat az előadáson látott módon tevékenység objektumokkal kell megvalósítani, amelyeknek osztálya vagy az öt programozási tétel (Summation, Counting, Selection, LinSearch, MaxSearch) osztálysablonjának valamelyikéből származik, vagy az általános felsoroló (Enumerator> osztálysablonból. **Nem definiálhatja felül** a Run(), Do(), LoopCond() metódusokat, az Init()-et is csak akkor, ha a Summation osztályból származtat! A saját kódban **nem szerepelhet ifstream típusú objektum**, helyette használja a szekvenciális inputfájl felsoroló osztály-sablonját (SeqInFileEnumerator), és kezelje le annak OPEN_ERROR kivételét! A saját kódban egyáltalán **ne szerepeljen ciklus** és csak összefuttató felsoroló Next() műveletében **használhat rekurzív függvényhívást.***

1. Egy szöveges állományban neptunkód-osztályzat párokat tartalmazó sorokat helyeztünk el. (Az neptunkód 6 karakter hosszú, utána egy szóköz jön, azt követően pedig egy 0 és 5 közötti egész szám.) Az állomány neptunkód szerint növekedően rendezett (ugyanolyan neptunkódot tartalmazó sorból egymás után több is lehet). Igaz-e, hogy minden hallgatónak az átlaga legalább négyes? A választ a konzolablakba írjuk! (A kiíratáson kívül csak egyetlenegy üres „else” ágú elágazást használjon!)

   input:
   ```
   AAAAAA 3
   AAAAAA 5
   BBBBBB 2
   BBBBBB 5
   BBBBBB 5
   ```
   output:
   ```
   Eredmény: Igaz
   ```

2. Egy szöveges állományban számlaszám-befizetés párokat tartalmazó sorokat helyeztünk el. (A számlaszám 8 karakter hosszú, utána egy szóköz jön, azt követően pedig egy egész szám.) Az állomány számlaszám szerint növekedően rendezett. Igaz-e, hogy összességében egyik számláról sem vettek ki 100 000 Forint vagy annál több pénzt (azaz -100 000-nél nem nagyobb egy-egy számlaszám összesített bevétele)? A választ a konzol ablakba írjuk! (A kiíratáson kívül csak egyetlenegy üres „else” ágú elágazást használjon!)

   input:
   ```
   00000000 -10000
   11111111  77000
   11111111  -200000
   22222222   8000
   ```
   output:
   ```
   Eredmény: Nem
   ```

3. Egy szöveges állományban neptunkód-osztályzat párokat tartalmazó sorokat helyeztünk el. (Az neptunkód 6 karakter hosszú, utána egy szóköz jön, azt követően pedig egy 0 és 5 közötti egész szám.) Az állomány neptunkód szerint növekedően rendezett (ugyanolyan neptunkódot tartalmazó sorból egymás után több is lehet). Keressük meg azt a hallgatót, akinek az átlaga a legjobb (írjuk ki az neptun kódját és az átlagát a konzol ablakba)! Lehet, hogy nincs hallgató. (A kiíratáson kívül csak egyetlenegy üres „else” ágú elágazást használjon!)

   input:
   ```
   AAAAAA 3
   AAAAAA 5
   BBBBBB 2
   BBBBBB 4
   BBBBBB 5
   ```
   output:
   ```
   Eredmény: AAAAAA  4.0
   ```

4. Egy szöveges állományban számlaszám-befizetés párokat tartalmazó sorokat helyeztünk el. (A számlaszám 8 karakter hosszú, utána egy szóköz jön, azt követően pedig egy egész szám.) Az állomány számlaszám szerint növekedően rendezett. Írjuk ki egy szöveges állományba az egyes számlák összesített forgalmát számlaszám-forgalom párokat tartalmazó sorok formájában! (A kiíratáson kívül csak egyetlenegy üres „else" ágú elágazást használjon!)

input: ***output:***

```
00000000 10000          00000000 8000
00000000 -2000          11111111 7000
11111111  7000
```

5. Egy karakterenként olvasható szekvenciális fájl egy szöveget tartalmaz. Tömörítse ezt a szöveget – és helyezze el egy szekvenciális output fájlba – úgy, hogy egy olyan szövegrész helyett, amely kettőnél több azonos karakterből áll, a „#<karakter><darabszám>" formájú kódot írja le, ahol a darabszám a karakter ismétlődéseinek száma! (Az eredeti szövegben nincs # jel.) (A feladat megoldásához egy üres „else" ágú és egy kétágú elágazást használhat!)

input: `Az ilyen hosszúakat:    (ez itt négy szóköz), és az ilyeneket: aaaaaa tömörítjük.`

output: `Az ilyen hosszúakat: # 4(ez itt négy szóköz), és az ilyeneket: #a6 tömörítjük.`

6. Egy karakterenként olvasható szekvenciális fájl egy szöveget tartalmaz. Ha a közvetlenül egymás után álló kettőnél több azonos karakter helyett a „#<karakter><darabszám>" formájú kódot írnánk, ahol a darabszám a karakter ismétlődéseinek száma, akkor tömöríteni tudnánk a szöveget. Például:

eredeti: `Az ilyen hosszúakat:    (ez itt négy szóköz), és az ilyeneket: aaaaaa tömörítjük.`

tömörített: `Az ilyen hosszúakat: # 4(ez itt négy szóköz), és az ilyeneket: #a6 tömörítjük.`

Számolja ki és írja ki a konzol ablakba, hogy mekkora mértékű lenne a fenti módszer melletti tömörítés: (eredeti hossz/tömörített hossz)*100! (A feladat megoldásához egy üres „else" ágú és egy kétágú elágazást használhat!)

input: `Az ilyen hosszúakat:    (ez itt négy szóköz), és az ilyeneket: aaaaaa tömörítjük.`

output: `Tömörítés mértéke: 93.2203%`

7. Egy szöveges állomány egy csoport hallgatóinak megajánlott gyakorlati jegyeit tartalmazza neptunkód-osztályzat alakú sorok formájában (az neptunkód 6 karakter hosszú, utána egy szóköz jön, azt követően pedig egy 0 és 5 közötti egész szám). Azok a hallgatók, akik sikerrel szerepeltek tehetséggondozási programozási versenyen, a megajánlott jegyüknél eggyel jobbat kapnak, pontosabban azoknak a hallgatóknak nő eggyel a gyakorlati jegye, akik a programozási versenyen 40 pontnál jobb eredményt értek el, de a megajánlott jegyük nem elégtelen vagy jeles. A verseny eredményeit egy másik szöveges állomány tartalmazza neptunkód-pontszám alakú sorok formájában (az neptunkód 6 karakter hosszú, utána egy szóköz jön, azt követően pedig egy 0 és 100 közötti egész szám). Készítse el a csoport hallgatóinak végleges gyakorlati jegyeit tartalmazó szöveges állományt

neptunkód-osztályzat alakú sorok formájában! Mindkét állomány neptunkód szerint növekedően rendezett: az gyakorlati jegyeket tartalmazó állomány neptunkód szerint szigorúan növekedően rendezett, de a második állomány a programozási verseny részeredményeit tartalmazza, nem az elért összpontszámot. Így ebben az állományban egy neptunkód, többször is szerepelhet egymás után, ezért egy-egy hallgató összpontszámát ki kell számolni. (Kizárólag a saját felsorolók Next() metódusaiban használhat elágazásokat!)

input1:	***input2:***	***output:***
AAAAAA 3	AAAAAA 14	AAAAAA 3
CCCCCC 2	BBBBBB 40	CCCCCC 2
DDDDDD 4	BBBBBB 48	DDDDDD 4
EEEEEE 5	EEEEEE 39	EEEEEE 5
FFFFFF 1	EEEEEE 48	FFFFFF 1
	FFFFFF 41	

8. Egy szöveges állomány egy bank ügyfeleinek számlaegyenlegeit tartalmazza számlaszám-egyenleg alakú sorok formájában (a számlaszám 8 karakter hosszú, utána egy szóköz jön, azt követően pedig egy egész szám). Egy másik szöveges állomány az ügyfelek aznapi tranzakcióit tartalmazza számlaszám-tranzakció alakú sorok formájában (a számlaszám 8 karakter hosszú, utána egy szóköz jön, azt követően pedig egy egész szám). Készítse el az ügyfeleknek a tranzakciók alapján időszerűsített egyenlegeit tartalmazó szöveges állományt számlaszám-egyenleg alakú sorok formájában! A fő tevékenység osztályát a Summation-ból kell származtatnia úgy, hogy csak az Add() metódust és a konstruktort definiálja újra, felsorolónak pedig egy saját gyártású összefuttató felsorolót használjon. Mindkét állomány számlaszám szerint növekedően rendezett: a számlaegyenlegeket tartalmazó állomány számlaszám szerint szigorúan növekedően rendezett, de a tranzakciók állományában egy számlaszám többször is szerepelhet egymás után, azaz egy-egy ügyfél több tranzakciót is végezhet egy nap, ezért az ügyfél tranzakcióit és eredeti egyenlegét kell összegezni. Olyan tranzakció is előfordulhat, amelyik nem létező számlára vonatkozik. Adjon ilyenkor hibajelzést a konzolra! (Kizárólag a saját felsorolók Next() metódusaiban használhat elágazásokat!)

input1:	***input2:***	***output:***
11111111 100	11111111 50	11111111 150
33333333 -20	44444444 -40	33333333 -20
44444444 500	44444444 -20	44444444 440
55555555 500	66666666 400	55555555 500
66666666 1000		66666666 1400

9. Egy *x* szekvenciális inputfájl egy könyvtár nyilvántartását tartalmazza. Egy könyvről ismerjük az azonosítóját, a szerzőjét, a címét, a kiadóját, a kiadás évét, az aktuális példányszámát, az ISBN számát. A könyvek azonosító szerint szigorúan növekvően rendezettek. Egy *y* szekvenciális inputfájl az aznapi könyvtári forgalmat mutatja: melyik könyvből hányat vittek el, illetve hoztak vissza. Minden bejegyzés egy azonosítót és egy előjeles egészszámot - ha elvitték: negatív, ha visszahozták: pozitív, ha leselejtezték: nulla - tartalmaz. A bejegyzések azonosító szerint növekvően rendezettek, azaz egy azonosítóra több bejegyzés is lehet az aznapi forgalomban. Aktualizáljuk a könyvtári nyilvántartást! A feldolgozás során keletkező hibaeseteket egy *h* sorozatba írjuk bele!

10. Egy vállalat dolgozóinak a fizetésemelését kell végrehajtani úgy, hogy az azonos beosztású dolgozók fizetését ugyanakkora százalékkal emeljük. A törzsfájl dolgozók sorozata, ahol egy dolgozót három adat helyettesít: a beosztáskód, az egyéni azonosító, és a bér. A törzsfájl beosztáskód szerint növekvően, azon belül azonosító szerint szigorúan monoton növekvően rendezett. A módosító fájl beosztáskód-százalék párok sorozata, és beosztáskód szerint monoton növekvően rendezett. Adjuk meg egy új törzsfájlban a dolgozók emelt béreit.