Minden feladatot kötelező beküldeni!

A beküldési határidő: ZH időpontja!

1.

a) Írj egy Katona nevű osztályt. Egy katonának tároljuk a támadó és védőerejét (egész számok). Ezek az adattagok legyenek **privát** láthatóságúak és legyenek lekérdezhetők és beállíthatók publikus metódusokon keresztül. Az osztály rendelkezzen egy olyan konstruktorral, mely beállítja a támadó és védőerejét az objektumnak 5,5-re, továbbá rendelkezzen egy olyan konstruktorral, mely a kapott paraméterek alapján állítja be ezeket. Definiáld felül az osztályban a toString() metódust úgy, hogy kiírja az egység támadó és védőerejét. pl.: ”TE: [támadóerő], VE: [védőerő]”. Továbbá definiálja felül az osztályban az equals metódust, amely az egyenlőséget a támadóerő és védőerő alapján vizsgálja.

b) Írj egy Nyilas és egy Landzsas osztályt, amely származik a Katona osztályból! A Nyilas osztálynak legyen egy privát láthatóságú lőtáv (egész szám) adattagja, mely publikus metódusokon keresztül legyen lekérdezhető és módosítható! A Nyilas támadóerejének lekérdezésekor a visszaadott érték legyen a támadóerő és a lőtáv összege! Mindkét osztály használja az ősosztály konstruktorát, a nyilas konstruktora várjon egy lőtáv paramétert is és ez alapján állítsa be a lőtáv értéket is! Definiáld felül ezekben az osztályokban is a toString() metódust. pl:”Nyilas: TE: [támadóerő], VE: [védőerő]” Itt a lőtáv már szerepel (a támadóerőhöz hozzáadva). Továbbá definiálja felül a Nyilas osztályban az equals metódust, amely az egyenlőséget a támadóerő, védőerő és lőtáv alapján vizsgálja.

c) Írj egy Teszt nevű futtatható osztályt! Az osztálynak legyen egy statikus megkuzd() metódusa amely két Katona objektumot vár paraméterként és a visszatérési értéke a győztes katona objektum. A megkuzd() metódusban kiszámoljuk mely érték nagyobb:

- az első katona támadóereje - a második védőereje, (ekkor az első katona nyer)
- vagy a második katona támadóereje - az első védőereje. (ekkor a második nyer)
- ha a két érték megegyezik, akkor az első katona nyer.

A metódus írja ki mely katonák küzdtek meg, és ki nyert.

A futtatható osztályban, a main metódusban állíts elő három katona objektumot (például két Nyilast és egy Lándzsást)! A létrehozott katonák segítségével valósíts meg egy csatát három katona részvételével: rendezz egy küzdelmet a megkuzd() metódus segítségével az első két katona között, majd a győztes és a harmadik katona között is! Írd ki mely katona maradt életben!

2. feladat

Fejesek 2

Az általános iskolai testnevelésórán a tornatanár nagyság szerint növekvő sorrendbe állítja a nebulókat. A sor elején áll a legalacsonyabb gyerek, a sor végén a legmagasabb. A sor elején álló gyerek kivételével a többieket megkéri, hogy mindegyikük jegyezze meg az előtte álló társa nevét (a gyerekek nevei szerencsére különböznek egymástól), hogy legközelebb könnyedén be tudjanak állni a sorba. Miután a gyerekek szépen elrendeződtek a sorban, a tanár a sor elejétől kezdve felváltva hol egy piros, hol egy kék színű sapkát húz az egymás után sorakozó gyerekek fejére, az első gyerekre piros sapkát húzva.

Írjon programot, amely a standard bemenet első sorából beolvas egy egész számot (*n*-et), ami a tornasorban álló gyerekek számát határozza meg, a tornasor legelején álló gyerek nevét, valamint egy újabb nevet (ami akár meg is egyezhet az előzővel, a későbbi hivatkozás kedvéért jelöljük most *X*-szel)! Ezt a három adatot egy-egy pontosvessző választja el egymástól. A következő $n-1$ sorban a tornasorban álló gyerekek adatai találhatók az alábbi formában:

név; *előtte*

A program írja a standard kimenetre azt, hogy az *X* nevű gyerek hány darab piros sapkás és kék sapkás fejet lát maga előtt, ha a sor elején álló legalacsonyabb társa felé tekint, feltételezve, hogy a gyerekek különböző magasságúak, és mindenki számára biztosított a „jó kilátás" a sor eleje felé! A két adatot egyetlen szóközzel válassza el egymástól!

Példa bemenet

```
5;Péter;Barnabás
Gábor;József
Samu;Barnabás
József;Péter
Barnabás;Gábor
```

letöltés szöveges állományként

A példa bemenethez tartozó kimenet

```
2 1
```

letöltés szöveges állományként

3. feladat

Budapest Kupa 1 (Java)

A Budapest Kupát a Teljesítménytúrázók Társasága írta ki. Elnyerésének feltétele, hogy a 2017-re meghirdetett 144 túrából legalább 10 túrán vegyen részt az ember, és igazoltan teljesítse is azt. Az ön feladata egy olyan program írása, amely az ebben a sorozatban szereplő túrák adatait dolgozza fel.

A standard bemenet első sorában a feldolgozandó túrák darabszáma szerepel (*N*). A következő *N* sor egy-egy túra adatait tartalmazza a következő formában:

év; *hónap*; *nap*; *túra_neve*; *résztáv*[; *résztáv*]...

Az *év*, a *hónap* és a *nap* az év egy napjának leírására szolgál: az *év* az évszámot, a *hónap* a hónap éven belüli sorszámát, a *nap* pedig a túra napját írja le az adott hónapon belül. Az *év*, *hónap*, *nap* adatok mindig egy valódi, létező dátumot jelölnek. A *túra_neve* egy sztring, a *résztáv* pedig egy pozitív egész szám. Az említett adatokat pontosvessző karakterek választják el egymástól egy soron belül. A pontosvessző karakter nem szerepel egyik túra nevében sem.

A túrák teljesítéséhez végig kell haladni a szervezők által kijelölt útvonalon, mely mentén ellenőrzőpontokat kell érinteni. Az ellenőrzőpontok egymástól való távolságát írják le az egyes sorokban szereplő résztávok. Ha egy túrán például a rajtból elindulva két ellenőrzőpontot kell érintenie a túrázónak, akkor három darab *résztáv* értéket fog a sor tartalmazni: a rajtból az első ellenőrzőpontba, az elsőből a másodikba, valamint a másodikból a célba vezető út hosszát. Így a túra teljesítendő útvonalát a résztávok összege fogja megadni.

A programjának a standard kimenetre kell írnia a leghosszabb teljesítendő útvonallal rendelkező túrák dátumát és nevét. Az adatokat rendezze dátum szerint növekvő sorrendbe! Ha két ilyen túrát is ugyanazon a napon rendeznének meg, akkor ezeket a túrákat a nevük alapján állítsa lexikografikusan növekvő sorrendbe (ábécérendbe)! Feltételezheti, hogy a bemeneti adatok között nem fog szerepelni két azonos nevű túra. A rendezett adatokat a példa kimenetben látható formában írja a programja a standard kimenetre!

Példa bemenet

```
5
2017;05;14;Budaorsi dolomitok;5;4;3;5;5
2017;02;04;Kitaibel Pal emlektura;2;3;2;3
2017;04;01;Baba;3;3;3;4;3;3;3
2017;01;21;Toldi Miklos emlektura;6;4;6;4
2017;04;01;Pipi;4;4;4
```

letöltés szöveges állományként

A példa bemenethez tartozó kimenet

```
2017;04;01;Baba
2017;05;14;Budaorsi dolomitok
```

letöltés szöveges állományként

4. feladat

Hullámvasutak (Java)

Egy zsúfolt vidámparkban olykor több órát is sorban kell ahhoz állni, hogy felülhessünk egy-egy hullámvasútra. A parkok méretére és a kígyózó sorokra való tekintettel érdemes megtervezni, hogy milyen sorrendben ülünk fel a játékaikra. A PortAventura World Európa egyik legnagyobb vidámparkja, mely a Costa Braván, Salou és Tarragona között található. Klasszikus területe (a kontinens egyetlen Ferrari parkján kívül) több tematikus világot tartalmaz, amelyek mindegyikében található legalább egy hullámvasút.

Írjon programot, amely a standard bemenet első sorából beolvas egy egész számot (*n*-et), amely a további feldolgozandó sorok (és hullámvasutak) darabszámát adja meg! A következő *n* sor mindegyikének a felépítése a következő:

hullámvasút_neve; *világ_neve*; *legkisebb_magasság*; *várakozási_idő*

A *hullámvasút_neve* és a *világ_neve* a hullámvasút, valamint a világ nevét tartalmazó sztringek, a *legkisebb_magasság* a felüléshez szükséges magasságot (cm-ben kifejezve), a *várakozási_idő* pedig az aktuális várakozási idő hosszát (percben kifejezve) tartalmazza.

A programja rendezze a beolvasott hullámvasutak adatait a várakozási idő szerint növekvő sorrendbe! Ha több azonos várakozási idővel rendelkező hullámvasút lenne a listában, akkor őket a legkisebb magasság szerint rendezze csökkenő sorrendbe! Ha ezek alapján sem tudna különbséget tenni két hullámvasút között, akkor őket a nevük szerint rendezze lexikografikus sorrendbe! Feltételezheti, hogy nincs két egyforma nevű hullámvasút.

A hullámvasutak adatait (név, világ és várakozási idő) a példa kimenetben látható formában írja a standard kimenetre!

Példa bemenet

```
6
Furius baco;Polynesia;140;120
Shambhala;China;140;120
Dragon Khan;China;140;80
Stampida;Far West;120;20
Tami Tami;SesamoAventura;100;20
El Diablo;Mexico;140;30
```

letöltés szöveges állományként

A példa bemenethez tartozó kimenet

```
Stampida (Far West): 20
Tami Tami (SesamoAventura): 20
El Diablo (Mexico): 30
Dragon Khan (China): 80
Furius baco (Polynesia): 120
Shambhala (China): 120
```

letöltés szöveges állományként

5. feladat

Izzasztó feladat (Java)

Írjon programot, amely a standard bemenet első sorából beolvas egy egész számot (*n*-et) és egy településnevet, amely nem tartalmaz szóköz karaktert! A bemenet következő *n* sora egy-egy település nevét és az adott településen mért hőmérsékleti adatot tartalmazza, az előbbit sztringként, az utóbbit egész számként. A két értéket minden sorban egy kettőspont karakter választja el egymástól.

A programja döntse el, hogy a bemenet első sorából beolvasott településnév szerepel-e a megadott felsorolásban, és ha igen, akkor írja a standard kimenetre mindazoknak a településeknek az adatait (nevét és hőmérsékletét) a példa kimenetben látható formában, amelyeknél a hőmérséklet értéke meghaladja az első sorból beolvasott településen mért hőmérsékletet! A kimeneten a hőmérsékleti értékek szerint csökkenő sorrendben jelenjenek meg a települések adatai! Amennyiben több településen is azonos hőmérsékletet mértek volna, akkor ezeket a településnevek lexikografikusan növekvő sorrendjében (ábécérendben) írja a standard kimenetre!

Amennyiben a bemenet első sorából beolvasott településnév nem szerepelne a felsorolásban, akkor a programja mindössze egy „`Missing data`" tartalmú sort írjon a standard kimenetre! Ne felejtse el ebben az esetben sem soremelés karakterrel zárni a sort!

1. példa bemenet

```
6 Debrecen
Budapest:19
Debrecen:24
Esztergom:25
Miskolc:26
Szeged:20
Szolnok:25
```

letöltés szöveges állományként

Az 1. példa bemenethez tartozó kimenet

```
Miskolc (26)
Esztergom (25)
Szolnok (25)
```

6. feladat

Másfél millió lépés (Java)

Írjon programot, amely a standard bemenet első sorából beolvas egy *n* pozitív egész számot, amely a további sorok számát adja meg! A következő *n* sorban turisták vándorlásainak leírásai szerepelnek a következő formában:

turista_neve;*útvonalleíró_sztring*

A fenti sorban pontosvesszők karakter választja el egymástól a turista nevét az általa megtett útvonalat leíró sztringtől. Az útvonalleíró sztringben `K`, `P`, `S` és `Z` karakterek jelzik azokat a szakaszokat, amelyeket a turista az országos kék, illetve a helyi jelentőségű piros, sárga és zöld jelzéseken tett meg. A pont karakterek (`.`) ebben a sztringben a turistajelzés nélküli útszakaszokat jelölik.

Minden turista esetén számítsa ki, hogy hány szakaszt tett meg a turista az országos kék jelzésen, és hányat tett meg a helyi jelentőségű piros, sárga és zöld jelzéseken összesen! Rendezze a turisták adatait csökkenő sorrendbe aszerint, hogy melyikük tette meg a legtöbb szakaszt az országos kék jelzésen! Ha ez az érték több turista esetén is megegyezne, akkor őket a helyi jelentőségű egyéb színű jelzéseken megtett szakaszok száma szerint állítsa csökkenő sorba! Ha e szerint az érték szerint sem tudna dönteni két turista sorrendjéről, akkor őket állítsa nevük alapján lexikografikusan növekvő sorrendbe! A program írja az ily módon sorba rendezett turisták nevét a standard kimenetre, soronként egyet-egyet!

Példa bemenet

```
4
Teszt Elek;K.K.K.
Gipsz Jakab;.Z.PK..S.SKZ...P.Z..P.K.S
Vizi Palma;KSKPKZK
Bodon Odon;...K....K...K...
```

letöltés szöveges állományként

A példa bemenethez tartozó kimenet

```
Vizi Palma
Gipsz Jakab
Bodon Odon
Teszt Elek
```

letöltés szöveges állományként

7.feladat

Angol–magyar szótár 2

Az angolórákon minden alkalommal új szavakat tanulnak a diákok. Az új szavak darabszámát (*N*-et) a standard bemenet első sora tartalmazza. A következő *N* sor mindegyike az alábbi formában tartalmazza a megtanulandó angol szavakat és magyar megfelelőiket:

angol_szó:*magyar_szó*

Az *angol_szó* és a *magyar_szó* minden esetben tetszőleges, betű karaktereket tartalmazó sztring lehet, amely nem tartalmaz szóköz és kettőspont karaktereket. A kettőspont karakter kizárólag az angol és a magyar szó elválasztására szolgál.

Gyakran előfordul, hogy egy angol szónak több magyar megfelelője is van, illetve fordítva: hogy egy magyar szót többféle szóval is ki lehet fejezni angolul. Például a *take* angol szó magyarul *vesz*, *visz* és *fog* értelemben is használható, vagy a magyar *galamb* szót angolul *dove*-nak és *pigeon*-nak is mondhatjuk. Ha ilyen szavak szerepelnének a felsorolásban, akkor természetesen minden egyes jelentés külön-külön, önálló sorban szerepel.

Készítsen egy olyan programot, amely a standard bemenetről beolvassa a megtanulandó szópárokat! A program rendezze a szópárokat az angol szavak szerint lexikografikusan növekvő sorba (ábécérendbe)! Ha több szópárnál is azonos lenne az angol szó, akkor ezeket a szópárokat a magyar jelentésük szerint tegye lexikografikusan növekvő sorrendbe (ábécérendbe)! Az ily módon sorba rendezett szópárokat aztán a bemeneti formátumhoz hasonló alakban írja ki a standard kimenetre!

Példa bemenet

```
5
take:visz
dove:galamb
take:vesz
pigeon:galamb
take:fog
```

letöltés szöveges állományként

A példa bemenethez tartozó kimenet

```
dove:galamb
pigeon:galamb
take:fog
take:vesz
take:visz
```

letöltés szöveges állományként