**Konténerek jellemzői:**

* vector: elemeket index alapján azonosítjuk, ritkán bővítjük, akkor is a végére szúrunk be
* list: elemeknek csak a sorrendjét ismerjük, sosem egy adott indeyű elemre vagyunk kíváncsiak, hanem mindig az egész listát végigjárjuk, ha valamire kíváncsiak vagyunk. Gyorsan lehet beszúrni a közepére
* set: magyarul halmaz. Arra jó, hogy megnézzük, hogy az adott elemet (vagy akár egy másik konténert), már egyszer hozzáadtuk-e vagy sem. Ha iterátorosan járod be, akkor sorbarendezve kapod az elemeket, de ez a sorbarendezés csak primitív típusok (int, float, char, String) esetében triviális, listákat, vectorokat, map-eket úgy rendez sorba, ahogy épp neki jól esik.
* map: Nagyon hasonló a set-hez, gyakran set helyett map-re szoktatok gondolni. Annyiban több, mint a set, hogy valamilyen azonosító alapján (kulcs) tudod elérni az elemeket.

Megjegyzés: char-t nyugodtan kezelhetitek egy 0 és 255 közti számként.

**Példák:**

* Tornasor: érkezik egy új gyerek, és be kéne szúrni. >> *list<string>* vagy *list< pair<string, int> >*
* Szabad neptunkódok, ha ismerjük azokat, amiket kiosztottunk? >> *set<string>*
* Neptunkód->név konverter >> *map<string, string>*
* Bevásárlólistát írok, és úgy szeretném beszúrni az elemeket, hogy egymás után következnének a boltban egymás közelében található termékek >> *list<string>*
* Bevásárlólista, ahol csak arra vagyok kíváncsi, hogy felírtam-e már az adott terméket, és hogy abból, mennyit szeretnék venni: *set<map<String, int>>* (string a termék neve, int a mennyisége)
* Keletiben a pénztárak: egy sor áll, 5 ablak van, ahol jegyet lehet venni, és folyamatosan érkeznek az emberek >> *list <ember>* (queue lenne az igazi)
* Nyilvántartási hivatal (vagy valami ilyesmi): személyi igazolvány alapján egy ember összes adata >> *map<string, valami\_struct vagy azonosító>*
* Madárfajok katalógusa: név -> a madár mely kontinensen él >> *map<string, string>*
* Adott egy kontines. Kérdés, hogy él-e ott adott nevű madár: >> *set<string>*
* Bűnügyi nyilvántartás: név alapján az eddig elkövetett bűnök lajstroma >> *map<string, list<akármi>>*
* Teremfoglalás az ITK-n: termek 1-500-ig vannak sorszámozva, és arra vagyok kíváncsi, hogy foglalt-e az, amit le szeretnék foglalni >> *set<int>*
* Vizsganaptár készítése június hónapra: minden naphoz hozzátartozik egy felsorolás, hogy mely tárgyakból fognak aznap írni (azt is tárolni kell, hogy adott napon nincs egy vizsga sem) >> *vector<list<string>>*
* Egyszerű gráfot akarok eltárolni (szomszédsági-mátrix segítségével: az i.sor j.eleme megadja, hogy i csúcsból j-be mekkora súlyú élen keresztül juthatok el közvetlenül) >> *vector<vector<int>>*
* BKV ellenőr szeretné tudni, hogy a bediktált utcanév létezik-e az adott irányítószám és város mellett >> *vector<map<string, string>* vagy *map<int, map<string, string>>* Ha kevés irányítószámot kezelünk (pl csak Nyugat-Magyarországi irányítószámok), akkor a map éri meg jobban, ha az összes utcát felvesszük a rendszerbe (minden irányítószámot használunk), akkor a vector jobb.
* Billentyűkombinációk: az operációs rendszer statisztikát szeretne arról, hogy egy adott billentyűkombinációt hány program használ, és pl, hogy azok a programok, amik a Ctrl+C billentyűkombinációt használják, hány más billentyűkombinációt ismernek átlagosan: *set<set<set<int>>>*
	+ int: egy billentyű
	+ set<int>: billentyűkombinációt
	+ set<set<int>>: egy program billentyűkombinációi
	+ set<set<set<int>>>: telepített programok halmaza (a program neve nem fontos, mert csak statisztikát készítünk).
* Vízfesték-bolt: egy színt 3,4 vagy 5 szám kódol (ha RGB kódolást használunk (képernyőn való megjelenítéshez ez hasznos), akkor 3, ha CYMK kódolást (nyomtatók ezt használják általában), akkor 4, egyébként pedig képzeljük el, hogy a festéket 5 különböző komponensből keveri ki a cég, úgyhogy így is tárolhatják a festékeket). A színek palettákat alkotnak. Egy palettán 5-20 szín van, fontos, hogy milyen sorrendben (ha más sorrendben vannak, akkor az már egy másik palettának számít). És a boltban sok paletta van, egy palettát pedig a színei és azok sorrendje határoz meg. Ezen kívül fontos az, hogy ha egyszer csak gondolok egyet és minden paletta elejére be akarok szúrni mondjuk feketét (egy igazi festő nem használ feketét, de a rivális cég rak minden palettájába, és nekik jól megy az üzlet…), akkor az minél gyorsabban meg tudjam tenni. Vagy ha elfogy a bíborvörös, akkor törölni tudjam az összes palettának a közepéről is akár. >> *set<list<vector>>>*

**Tavalyi beadandóra készült megoldások:**

* + - map<int, vector<vector<int>>> utcak
* ELMŰ budapesti kirendeltségén új számítógépet vesznek, és ezen való örömükben elkezdik a lakossági adatokat digitális formában tárolni. Elsőként az átalánydíjak adathalmazát szeretnék rendszerezni, ezért a hozzájuk tartozó utcák irányítószámát (feltételezzük, hogy minden utca irányítószáma különböző), és az utca lakásainak átlagos fogyasztási adatait egy map-be rakják.
* Utcánként a házszámok alá tartozó lakások adatait vektorba rakják. Az egyes házszámok alatt található lakások (tömbházakról van ugyanis szó) adatait pedig ezen belül is egy újabb vectorban tárolják.
	+ - list<set<set<char>>> pizzeriak;
* A tarolóban egy város pizzeriáinak étlapjait tároljuk, árfekves szerinti sorrendben (egy pizzeria annal drágább, minel tobb fele pizza kaphato). A pizzákat a feltéteikkel azonosítjuk, a feltéteket pedig karakterek jelölik. Egy lehetséges pizza: trfg, erre a pizzára a t, az r, az f es a g feltétek kerülnek. A tarolóba bármikor lehet új pizzériát beszúrni az 'uj\_pizzeria' paranccsal, ekkor a program elkéri a hozzáadott pizzériában kapható pizzák kódjat. Ha tudjuk, hogy milyen pizzát szeretnénk enni, 'lekerdez' paranccsal lekérdezhetjuk annak az étteremnek az étlapját, ahol ez a tipusú pizza a legolcsóbb, hogy a partnerünknek is tudjunk rendelni valami finomat.
* A pizzériák nevét és más adatát az adatszerkezet nem tartalmazza, ezt opcionálisan egy másik adaszerkezettel (pl: map<set<set<char>>, pizzeria\_adatok\_struct> ahol set<set<char>> az étlap) tárolhatjuk és hívhatjuk elő, ha már megvan, hogy melyik étlapról szeretnénk választani.
	+ - list<list<vector<bool>>>
* Egy sportközpont új fitnesscsoportokat indít. Kezdetben 2 csoportot indítanak 5-5 fõvel. Minden jelentkezõnek ki kell töltenie egy 3 kérdésbõl álló igaz-hamis kérdõívet az egészségi állapotáról. Egy statisztikus csoport elemzi az emberek kérdésekre adott válaszait.
* Feladatok:
	+ Adatok betöltése, kiírása fájlba (a betoltendo fajl neve: adatok.txt) betolt, fajlbair parancsokkal
	+ Új jelentkezõk felvétele a csoportokba: a kérdésekre adott válaszaiknak csatolása az adatszerkezetet konzolra kell beírni a 2 új ember válaszait (pl: true true false false true false) hozzaad paranccsal
	+ Annak lekérése, hogy hányan veszélyeztetettek cukorbetegséggel(aki legalább két kérdésre igennel válaszolt), és ez az emberek hány százalékafeladat paranccsal
* Indoklás: A belsõ vektor tartalmazza a kérdésekre adott igaz-hamis válaszokat. Azért vectort használok, mert így lekérdezhetõ, hogy egy adott sorszámú kérdésre mi a válasz. A belsõ lista tárolja az embereket, akik jelentkeztek a csoportokba. Ezen a listán végig lehet menni, de nem kell egy adott elemére hivatkozni, mert a statisztikus csoportot csak az összeredmény érdekli. A külsõ lista a csoportokat tartalmazza. Ezt is csak be kell járni, illetve összehasonlíthatóak az egyik csoport válaszai a másik csoport válaszaival.