Zadatak 1

Napišite parametriziranu strukturu **stablo** koja reprezentira stablo čiji čvorovi osim podatka imaju i identifikator (tipa **int**). Svaki čvor može imati nula ili više djece (čvorovi ne moraju imati jednak broj djece). Svaki čvor sadrži polje pokazivača na svoju djecu. Polja moraju biti dinamički alocirana, pri tome pripazite da se pri svakom dodavanju/izbacivanju čvora ne mora ponovno realocirati memorija. Dodajte sve potrebne varijable u strukturu da bi to mogli ostvariti. Struktura će se koristiti sa standardnim C++ tipovima nad kojima je definiran operator <.

Za navedenu strukturu implementirajte:

- konstruktor bez parametara
- konstruktor koji kao parametre prima element i njegov identifikator te stvara stablo koje se sastoji samo od korijena koji sadrži zadane podatke.
- Konstruktor koji kao ulazni parametar prima **string**. Konstruktor stvara stablo koje se sastoji od korijena, njegove djece i djece njegove djece. String na početku sadrži niz brojeva razmaknut razmakom, gdje prvi broj definira broj djece korijena a svaki sljedeći broj djece jednog djeteta od korijena. U nastavku se nalaze elementi stabla odvojeni znakom ';'. Djeca svakog čvora moraju biti sortirana uzlazno te im se slijedno dodjeljuje identifikator.
  Primjer: Za ulazni string `"2 3 1;4;1;12;5;6;9;555"` treba stvoriti stablo:
  (4 , 0)
  (1 , 1) (12 , 2)
  (5 , 3) (6 , 4) (9 , 5) (555 , 6)
  Uputa: provjeru parametra tipa unutar konstruktora možete raditi korištenjem:
  ```
  #include <type_traits>
  std::is_same<T, int>::value //primjer za int
  ```
- Napišite funkciju `int ubaci(int iden, T pod, int mid);` koja ubacuje element *pod* u stablo kao dijete čvoru s identifikatorom *iden*. Varijabla *mid* sadrži vrijednost najvećeg identifikatora čvora stabla. Funkcja vraća 1 ako je čvor uspješno ubačen, inače vraća 0 (ne postoji čvor s identifikatorom *iden*).
- Napišite funkciju `void maxId(int *m);` koja sprema vrijednost najvećeg identifikatora čvora stabla u varijabilni argument *m*.
- Napišite funkciju `int izbaci(int iden);` koja izbacuje čvor s identifikatorom *iden*, njegovu djecu i svu djecu njegove djece iz stabla (cijelo pod stablo). Funkcija vraća 1 ako je izbacivanje uspješno provedeno, inače vraća 0 (čvor nije bio element stabla).
- Napišite funkciju `void ispisi_stablo(std::queue<stablo<T>*> q);` koja ispisuje stablo razinu po razinu, čvorovi iste razine su odmaknuti razmakom, svaka razina je zapisana u svom retku (vidi primjer stabla gore).

Napomena: zabranjeno je korištenje STL-a za implementaciju svih točaka osim zadnje, u njoj smijete koristiti red kao ulazni parametar i još jedan takav red kao pomoćnu varijablu.

Zadatak 2

Napravite dvije strukture **Automobil** i **Automobili**.

Struktura **Automobil** sadrži **marka** (string) koji predstavlja marku automobila, **lokacija** (string) koji predstavlja lokaciju na kojoj se nalazi automobil, **cijena** (int) koja predstavlja cijenu automobila, **konjskaSnaga** (int) koja predstavlja konjsku snagu automobila. Struktura **Automobil** ima slijedeće funkcije članice:

- Konstruktor **Automobil (string marka, string lokacija, int cijena, int konjskaSnaga)**.
- Funkcija **Marka()** vraća marku automobila
- Funkcija **Lokacija()** vraća lokaciju na kojoj je automobil
- Funkcija **Cijena()** vraća cijenu automobila
- Funkcija **KonjskaSnaga()** vraća konjsku snagu automobila
- Funkcija **Razliciti (Automobil a)** vraća da li su dva automobila različita. Različiti su ako im je barem jedno od marka, lokacija , cijena ili konjska snaga različito, inače vraća false
- Funkcija **Ispis()** koja ispisuje jedan automobil tako da mu ispisuje marku, lokaciju, cijenu i konjsku snagu odvojeno zarezom u jednoj liniji, primjer ispisa je *AutomobilMarke1, Zagreb, 500000, 250*

Struktura **Automobili** sadrži polje **automobili** koje sadrži maksimalno 200 automobila i **brojAutomobila** (int) koji pokazuje koliko automobila imamo u polju **automobili**. Struktura Automobili ima slijedeće funkcije članice:

- Defaultni konstruktor **Automobili()** koji postavlja **brojAutomobila** na nulu
- Konstruktor **Automobili(Automobil a)** koji stvara automobile sa samo jednim automobilom **a**.
- Funkcija **Dodaj(Automobil a)** koja dodaje automobil **a** u polje **automobili** i povećava **brojAutomobila** za jedan. Moguće je imati u polju **automobili** više automobila sa istim vrijednostima (marka, lokacija, cijena, konjskaSnaga), tj. moguće je imati iste automobile više puta.
- Funkcija **BrojAutomobila()** vraća **brojAutomobila**.
- Funkcija **ItiAutomobil(int i)** vraća automobil na i-tom indeksu u polju **automobili**. Pretpostavka je da indeksi kreću od nule.
- Funkcija **SortirajPoCijeniUzlazno()** koja sortira automobile po cijeni od niže prema većoj. Ako ima više automobila sa istom cijenom nebitno koji će se prvi pojaviti u sortiranom prikazu.
- Funkcija **NadjiAutomobileSaKonjskomSnagom(int konjskasnaga)** koja vraća automobile (tj. vraća tip **Automobili**) koji imaju traženu konjsku snagu.
- Funkcija **NajmanjaCijenaNaLokaciji(string lokacija)** vraća na nekoj lokaciji najmanju cijenu automobila, a ako nema automobila na lokaciji vraća -1.
- Funkcija **Ispis()** koja ispisuje sve automobile iz polja **automobili** tako da je svaki automobil u svojoj liniji i elementi automobila su odvojeni zarezom npr.
  AutomobilMarke1, Split, 200000, 140
  AutomobilMarke2, Zagreb, 500000, 200

Napišite neki vaš main koji demonstrira upotrebu ovih funkcija.

U ovom zadatku ne smijete koristiti spremnike iz STL-a (vector, list, set itd.). Cjelokupni program spremite pod imenom **zadatak1.cpp**. Možete dodavati i dodatne svoje funkcije ako će vam pomoći u izradi zadatka.

Zadatak 3

Napravite dvije strukture **Proizvod** i **Proizvodi**.

Struktura **Proizvod** sadrži **naziv** (string) koji predstavlja naziv proizvoda, **adresa** (string) koji predstavlja adresu na kojoj se nalazi proizvod, **cijena** (int) koja predstavlja cijenu proizvoda, **tezina** (int) koja predstavlja težinu proizvoda u kilogramima. Struktura **Proizvod** ima slijedeće funkcije članice:

- Konstruktor **Proizvod (string naziv, string adresa, int cijena, int tezina)**.
- Funkcija **Naziv()** vraća naziv proizvoda
- Funkcija **Adresa()** vraća adresu proizvoda
- Funkcija **Cijena()** vraća cijenu proizvoda
- Funkcija **Tezina()** vraća težinu proizvoda
- Funkcija **Razliciti (Proizvod p1)** vraća da li su dva proizvoda različita. Različiti su ako im je barem jedno od naziv, adresa, cijena ili težina različito, inače vraća false.
- Funkcija **Ispis()** koja ispisuje proizvod tako da mu ispisuje naziv, adresu, cijenu i težinu odvojeno zarezom u jednoj liniji, primjer ispisa je *Proizvod1, Adresa1, 100, 2*

Struktura **Proizvodi** sadrži polje **proizvodi** koje sadrži maksima 100 proizvoda i **brojProizvoda** (int) koji pokazuje koliko proizvoda imamo u polju proizvodi. Struktura Proizvodi ima slijedeće funkcije članice:

- Defaultni konstruktor **Proizvodi()** koji postavlja brojProizvoda na nulu.
- Konstruktor **Proizvodi(Proizvod p)** koji stvara proizvode sa samo jednim proizvodom p.
- Funkcija **Dodaj(Proizvod p)** koja dodaje proizvod **p** u polje **proizvodi** i povećava **brojProizvoda** za jedan. Moguće je imati u polju **proizvodi** više proizvoda sa istim vrijednostima (naziv, adresa, cijena, tezina), tj. moguće je imati iste proizvode više puta.
- Funkcija **BrojProizvoda()** vraća brojProizvoda.
- Funkcija **ItiProizvod(int i)** vraća proizvod na i-tom indeksu u polju proizvodi. Pretpostavka je da indeksi kreću od nule.
- Funkcija **SortirajPoCijeniSilazno()** koja sortira proizvode po cijeni od veće prema nižoj. Ako ima više proizvoda sa istom cijenom nebitno koji će se prvi pojaviti u sortiranom prikazu.
- Funkcija **NadjiProizvode(string adresa, int maxCijena, int minTezina)** koja vraća proizvode koji su na danoj adresi, a kojima je cijena manja ili jednaka od **maxCijena** i tezina veća ili jednaka od **minTezina**. Pretpostavite da barem jedan takav proizvod postoji.
- Funkcija **NajvecaCijenaNaAdresi(string adresa)** vraća na nekoj adresi najveću cijenu proizvoda, a ako nema proizvoda na adresi vraća -1.
- Funkcija **Ispis()** koja ispisuje sve proizvode iz polja **proizvodi** tako da je svaki proizvod u svojoj liniji i elementi proizvoda su odvojeni zarezom npr.
  Proizvod1, Adresa1, 10, 2
  Proizvod1, Adresa2, 5, 1

Napišite neki vaš main koji demonstrira upotrebu ovih funkcija.

U ovom zadatku ne smijete koristiti spremnike iz STL-a (vector, list, set itd.). Cjelokupni program spremite pod imenom **zadatak1.cpp**. Možete dodavati i dodatne svoje funkcije ako će vam pomoći u izradi zadatka.

Zadatak 4

Napravite dvije strukture **Apartman** i **Apartmani**.

Struktura **Apartman** sadrži **naziv** (string) koji predstavlja naziv apartmana, **mjesto** (string) koji predstavlja mjesto u kojem se nalazi apartman, **cijena** (int) koja predstavlja cijenu apartmana po danu, **brojOsoba** (int) koja predstavlja koliko maksimalno osoba može primiti apartman. Struktura Apartman ima slijedeće funkcije članice:

- Konstruktor **Apartman (string naziv, string mjesto, int cijena, int brojOsoba)**.
- Funkcija **Naziv()** vraća naziv apartmana
- Funkcija **Mjesto()** vraća mjesto u kojem je apartman
- Funkcija **Cijena()** vraća cijenu apartmana
- Funkcija **BrojOsoba()** vraća broj osoba u apartmanu
- Funkcija **Jednaki (Apartman a)** vraća da li su dva apartmana jednaka. Jednaki su ako su im naziv, mjesto , cijena i brojOsoba jednaki, inače vraća false.
- Funkcija **Ispis()** koja ispisuje jedan apartman tako da mu ispisuje naziv, mjesto, cijenu i broj osoba odvojeno zarezom u jednoj liniji, primjer ispisa je *Apartman1, Zagreb, 500, 2*

Struktura Apartmani sadrži polje **apartmani** koje sadrži maksima 100 apartmana i **brojApartmana** (int) koji pokazuje koliko apartmana imamo u polju **apartmani**. Struktura Apartmani ima slijedeće funkcije članice:

- Defaultni konstruktor **Apartmani()** koji postavlja brojApartmana na nulu
- Konstruktor **Apartmani(Apartman a)** koji stvara apartmane sa samo jednim apartmanom **a**.
- Funkcija **Dodaj(Apartman a)** koja dodaje apartman **a** u polje **apartmani** i povećava brojApartmana za jedan. Moguće je imati u polju **apartmani** više apartmana sa istim vrijednostima (naziv, mjesto, cijena, brojOsoba), tj. moguće je imati iste apartmane više puta.
- Funkcija **BrojApartmana()** vraća brojApartmana.
- Funkcija **ItiApartman(int i)** vraća apartman na i-tom indeksu u polju apartmani. Pretpostavka je da indeksi kreću od nule.
- Funkcija **SortirajPoCijeniUzlazno()** koja sortira apartmane po cijeni od niže prema većoj. Ako ima više apartmana sa istom cijenom nebitno koji će se prvi pojaviti u sortiranom prikazu.
- Funkcija **NadjiApartmaneUMjestu(string mjesto)** koja vraća apartmane koji su u traženom mjestu. Pretpostavite da postoji barem jedan takav apartman.
- Funkcija **NajmanjaCijenaUMjestu(string mjesto)** vraća u nekom mjestu najmanju cijenu apartmana, a ako nema apartmana u mjestu vraća -1.
- Funkcija **Ispis()** koja ispisuje sve apartmane iz polja **apartmani** tako da je svaki apartman u svojoj liniji i elementi apartmana su odvojeni zarezom npr.

  Apartman1, Dubrovnik, 1000, 2

  Apartman1, Zagreb, 500, 2

Napišite neki vaš main koji demonstrira upotrebu ovih funkcija.

U ovom zadatku ne smijete koristiti spremnike iz STL-a (vector, list, set itd.). Cjelokupni program spremite pod imenom **zadatak1.cpp**. Možete dodavati i dodatne svoje funkcije ako će vam pomoći u izradi zadatka.

Zadatak 5

Napišite strukturu `trojka` koja pamti tri decimalna broja (`float`). Struktura treba imati:

- Konstruktor koji prima tri parametra tipa `float`.
- Funkciju `vrati` koja prima jedan parametar tipa `int`. Ukoliko mu je vrijednost 0, funkcija vraća prvi broj, ukoliko je 1 vraća drugi, a ako je 2 vraća treći. Možete pretpostaviti da ostalih mogućnosti neće biti.
- Funkciju `naj` koja vraća razliku najvećeg i najmanjeg od ta tri broja pomnoženu sa srednjim brojem po veličini (u slučaju npr. 1,2,1, najmanji i srednji broj je 1, a najveći 2).
- Destruktor koji ispisuje sva tri broja u oblim zagradama, odvojena zarezom, te prelazi u novi red. Npr. (`3.14,2.72,1.41`). Prilikom ispisa, broj decimala je proizvoljan (tj. ne trebate zaokruživati, ni formatirati ispis).
- Napišite i neki main koji demonstrira upotrebu strukture.

Cjelokupni program spremite pod imenom `zadatak1.cpp`.

Zadatak 6

Napišite parametriziranu strukturu `spoji` koja predstavlja ploču od 6 redaka i 7 stupaca na kojoj se igra modificirana verzija igre *Connect Four*, u kojoj dva igrača naizmjence stavljaju svoje pločice i svakome je cilj napraviti horizontalni niz od 4 uzastopne pločice. Struktura treba imati sljedeće članice:

- konstruktor: prima dva parametra koji označavaju oznaku za svakog igrača (onog tipa po kojem je struktura parametrizirana i kojeg `cout` zna ispisati).
- `ubaci` – prima indeks `i` $(1 \leq i \leq 7)$. U i-ti stupac odozgora ubacuje pločicu onog igrača koji je na redu. Pločica pada na najdonje slobodno mjesto u tom stupcu. Funkcija vraća 1 ako je element uspješno ubačen i u sljedećem koraku je na redu sljedeći igrač. Ako je taj stupac već bio popunjen, funkcija vraća 0 i isti igrač ostaje na redu u sljedećem koraku.
- `pobijedio` – vraća 1 ako je stanje na ploči takvo da je pobijedio prvi igrač, 2 ako je pobijedio drugi igrač, a inače 0. Možete pretpostaviti da će uvijek biti najviše jedan pobjednik.
- `ispis` – ispisuje ploču, i to svaki redak ploče u odvojeni redak na ekranu. Sve elemente jednog retka ispisati jedan iza drugog, a ako na nekom mjestu nema pločice, ispisati znak `'.'`, točno kao u primjeru niže.
- destruktor ispisuje konačno stanje na ploči. Također treba ispisati poruku je li pobijedio prvi ili drugi igrač ili nema pobjednika.

Primjer:

```
int main() {

spoji<char> A(4,'*','o');

A.ubaci(1);   A.ubaci(2);   A.ubaci(2);   A.ubaci(2);
A.ubaci(3);   A.ubaci(5);   A.ubaci(4);   A.ubaci(4);
A.ubaci(3);   A.ubaci(4);   A.ubaci(4);   A.ubaci(3);
A.ubaci(5);

A.ispis();

A.ubaci(4); A.ubaci(4); A.ubaci(4); A.ubaci(5);

return 0;

}
```

Ispis#1

```
.......
.......
...*...
.ooo...
.**o*..
*o**o..
```

Ispis#2

```
...*...
...o...
...*...
.oooo..
.**o*..
*o**o..

Pobijedio je drugi igrac.
```

U rješenju ne smijete koristiti STL-spremnike, a rješenje spremite pod imenom `zadatak1.cpp`.

U istoj datoteci napišite i `main` za testiranje ove strukture sa tipom `int`.

Napomena: Smijete se koristiti pisanim materijalima, web-stranicom kolegija i linkovima koji vode s nje.

Zadatak 7

Na natjecanju *Eurosong* natječu se države iz cijele Europe. Svaka od tih država ocjenjuje ostale države. Zaduženi ste za tehničku podršku na sljedećem *Eurosongu* pa morate napisati program koji vodi bodovnu evidenciju.

Evidencija bodova je preslikavanje koje državi (`string`) pridružuje vektor uređenih parova koji predstavljaju države za koje je ta država glasala. Na prvom mjestu u paru nalazi se ime države (`string`), a na drugom broj bodova (`int`) koji je dana država dobila. Država koja glasa daje bodove od 1 do 12, ne mora dati sve bodove, ali uvijek mora dati 12 bodova.

(a) Napišite glavni program u kojem ćete učitati podatke o bodovima u varijablu `B`. Podaci će biti unošeni u formatu kao u donjem primjeru. Imena država sastoje se samo od velikih i malih slova bez razmaka. Prva država u svakom retku je država koja glasa. Države za koje se glasa će u svakom retku biti poredane prema broju bodova koje su dobile.

(b) U glavnom programu ispišite državu koja ima najviše bodova i broj bodova koji je sakupila. Možete pretpostaviti da neće biti više država s istim brojem bodova na prvom mjestu. Smijete koristiti pomoćne *STL-containere* (vector, list, map, …), ali ne i pomoćna polja!

Primjer:

Ulazni podaci	Izlazni podaci

`Alb Nje 2 Fra 6 Hrv 12`
`Hrv Nje 1 Slo 3 BiH 5 Alb 12`
`BiH Slo 3 Fra 6 Hrv 10 Nje 12`
`Nje Hrv 2 BiH 4 Alb 10 Fra 12`
`Fra Nje 4 BiH 8 Alb 12`
`Slo Alb 3 Nje 6 BiH 12`
`kraj`
	`Alb 37`

Za učitavanje i ispisivanje koristite isključivo `cin` i `cout`.

Cjelokupni program spremite pod imenom `zadatak2.cpp`.

Zadatak 8

Napišite parametriziranu strukturu "CNLJS" koja predstavlja četiri figurice jednog igrača u igri Čovječe ne ljuti se. Figurice tog igrača zadane su s četiri indeksa (cjelobrojnog tipa), pozicije na ploči na kojima se nalaze te figurice. Pretpostavljamo da je svaki nenegativan cijeli broj moguće polje ploče (drugim riječima: start je nula, cilj je u plus beskonačnosti). Podsjećamo na jedno pravilo: figurica jednog igrača ne smije preskočiti drugu figuricu tog igrača, niti dvije figurice istog igrača ne mogu biti na istom polju. To će biti zadovoljeno i na ulazu. Svakom igraču karakteristični su i *vrijeme_razmisljanja* i *vrijeme_pomicanja*. Prilikom svakog poziva dolje navedenih funkcija (osim konstruktora i destruktora), igrač prvo razmišlja *vrijeme_razmisljanja* vremena, a zatim za pomicanje figura potroši vrijeme jednako umnošku broju koraka za koju pomakne figuru i *vrijeme_pomicanja* vremena. Sva vremena parametrizirana su nekim tipom *T* (int, double, float i sl.).

U ovom zadatku ne smijete koristiti STL. Unutar strukture **`CNLJS`** definirajte funkcije:

konstruktor – Prima niz integera duljine 4 (*int* niz*), te vremena *t1* i *t2* tipa T. U nizu su zapisane međusobno različite pozicije figurica na ploči. Vremena *t1* i *t2* redom označavaju *vrijeme_razmisljanja* i *vrijeme_pomicanja* igrača koji je vlasnik prethodno navedenih figurica.

`pomakni_max` – Figurica najbliža cilju pomiče se za jedno mjesto naprijed. Funkcija ništa ne vraća.

`pomakni_i –` Prima cijele brojeve *i* i *k*. Igrač pomiče *i*-tu figuricu po redu (sortirano po udaljenosti od cilja, počevši s onim najdaljima) za najviše moguće mjesta, a maksimalno *k* mjesta. Vraća *true*/*false*, ovisno o tome je li se figurica pomakla za *k* mjesta.

`max_korak –` Vraća najveći mogući korak koji neka figurica različita od one najbliže cilju može napraviti na legalan način. Ne prima ništa.

`vrijeme –` Prima vremena *t1* i *t2*. Vraća trenutno vrijeme igranja igrača koji je pozvao funkciju. Ako je broj *t1*, odnosno *t2* pozitivan, tada se *vrijeme_razmisljanja,* odnosno *vrijeme_pomicanja* mijenja na broj *t1*, odnosno *t2.*

`srusi` – Prima varijablu *C* tipa CNLJS (figurice drugog igrača) i poziciju *indeks* na ploči (mjesto neke figurice drugog igrača na ploči – pretpostavljajte da na toj poziciji postoji figurica drugog igrača). Igrač koji je pozvao funkciju nalazi figuricu kojom može srušiti figuricu na poziciji *indeks.* Na kraju izvođenja funkcije, igrač koji je pozvao funkciju pomiče svoju figuricu na mjesto indeks, te figurica koja je srušena odlazi na mjesto koje je najbliže startu, a da ne stane na isto polje kao neka druga figurica tog igrača. Funkcija vraća broj koraka koje je figurica koja je rušila napravila. Ako takav potez nije moguć, funkcija vraća -1. Pretpostavljamo da je potez od nula koraka također legalan potez. Vrijeme za ovaj potez je *vrijeme_razmisljanja* i broj koraka figure koja ruši puta *vrijeme_micanja.*

destruktor – Ispisuje pozicije figurica na kraju igre, odvojene razmakom.

Napišite i neki main koji testira strukturu koju ste napravili.

Zadatak spremite pod imenom **`zadatak1.cpp`**

Napomene:

- Ne smijete koristiti STL (osim <string> i <iostream>)

Zadatak 9

Napišite parametriziranu strukturu „Brodovi“ koja predstavlja ploču jednog igrača u igri Potapanje brodova. Radi jednostavnosti, brodovi su prezentirani na ploči oblika 1x20. Na svakoj poziciji nalazi se ili more ili brod dimenzije 1x1. Različiti brodovi se smiju dirati (za razliku od originalne igre). Svaki igrač na svoj način zabilježava na kojim pozicijama je more odnosno brod. Svim je zajedničko da ploču reprezentiraju nizom tipa T duljine 20 (tipa int, string, char…). U zadatku će postojati mogućnost da će u nekom trenutku nekom igraču svi brodovi biti potopljeni – u tom slučaju ne pretpostavljajte da je igri kraj, nijedna funkcija se zbog toga ne mijenja. U ovom zadatku ne smijete koristiti STL.

Unutar strukture **`Brodovi`** definirajte funkcije:

konstruktor - Prima varijable *b* i *m* tipa T. Stvara praznu ploču za igru za jednog igrača (dakle, sva polja su more). Varijable *b* i *m* redom označavaju kojim znakom taj igrač zapisuje brod odnosno more na ploči.

`unesiBrod` – Prima indeks na koje upisujemo brod igraču koji je pozvao funkciju. Vraća *true* ako do tada nije postojao brod na toj poziciji, *false* u suprotnom.

`gadjamUNizu` – Prima varijablu *B* tipa Brodovi (koja označava ploču nekog drugog igrača) te neku poziciju *x*. Igrač koji je pozvao funkciju potapa protivnikov brod na poziciji *x*, ukoliko takav postoji. U slučaju uspjeha, nastavlja gađati redom na pozicijama *x+1, x+2, ..., 19, 0, 1, 2, ...* sve dok ne promaši. Igrač koji je potapao brodove na kraju ove funkcije dobije bodova (poena) u iznosu broja potopljenih brodova. Brodove u varijabli *B* na kraju programa treba potopiti (pretvoriti u more). Nemojte pretpostaviti da dva igrača zapisuju istim simbolima more i brod u ploču, no smijete pretpostaviti da su varijable tipa Brodovi koje reprezentiraju ploče tih igrača parametrizirane istim tipom.

`boljiSam` – Prima varijablu *B* tipa Brodovi. Vraća *true* ako igrač koji je pozvao funkciju ima strogo više bodova (poena) od igrača *B*, *false* u suprotnom.

`kolikoBrodova –` Vraća broj preostalih brodova na ploči.

`losaPozicija –` Ne prima ništa, a vraća *true* ili *false*, ovisno o tome može li igrač koji je pozvao funkciju izgubiti igru (odnosno doći u poziciju da su mu svi brodovi potopljeni) u jednom ili nula poteza nekog drugog igrača. Jedan potez drugog igrača je jedan poziv funkcije *gadjamUNizu*.

destruktor – Ispisuje stanje ploče na kraju igre. Elementi niza neka budu odvojeni razmakom.

Napišite i neki main koji testira strukturu koju ste napravili.

Zadatak spremite pod imenom **`zadatak1.cpp`**

Napomene:

Ne smijete koristiti STL (osim <string> i <iostream>)

Zadatak 10

Napišite parametriziranu strukturu „Karte“, koja sadrži podatke o kartama u ruci jednog igrača za kartašku igru objašnjenu u nastavku. Svaki igrač ima 5 karata u ruci. Karte su opisane parametriziranom tipom T (double, int, char, i sl.). Smijete pretpostaviti da će svi igrači imati karte parametrizirane istim tipom. Cilj igre je da u ruci, nakon određenih zamjena karata s ostalim igračima, igrač ima točno 4 iste karte. Ako vam pomaže, smijete pretpostaviti da će tip T imati definiran operator „<“. U ovom zadatku ne smijete koristiti STL.

Unutar strukture **`Karte`** definirajte funkcije:

konstruktor – Prima niz od 5 elemenata tipa T (*T* niz*), koji nisu nužno međusobno različiti. Taj niz označava 5 karata u ruci.

`imam_kartu` – Prima varijablu *x* tipa T. Vraća *true* ako se među 5 karata u ruci nalazi karta *x, false* inače.

`moj_rezultat` – Prima varijable *koliko* tipa int i *koje* tipa T. Funkcija provjerava koja karta se pojavljuje najviše u ruci, te ju vraća preko varijabilnog parametra kroz varijablu *koje,* a broj ponavljanja te karte vraća također preko varijabilnog parametra kroz varijablu *koliko*. Ukoliko postoji više takvih karata, sami izaberite koju ćete vratiti. Funkcija „normalnim putem“ ne vraća ništa.

`zamijeni_s_igracem –` Prima varijablu *K* tipa Karte, te dvije varijable tipa T: *moja* i *protivnikova*. Ukoliko varijabla koja je pozvala funkciju ima kartu *moja*, te *K* ima kartu *njegova*, izvršava se zamjena – u ruku varijable koja je pozvala funkciju dolazi karta *protivnikova*, a u *K* dolazi karta *protivnikova*, te funkcija vraća *true*. Inače funkcija ne radi ništa i vraća *false*.

`pobjeda –` Prima varijablu K tipa Karte. Vraća cijeli broj između -1 i 3 prema sljedećem pravilu. Vraća 3 ako oba igrača (koji poziva funkciju i igrač reprezentiran s K) imaju karte kojima pobjeđuju. Vraća 2 ako pobjeđuje samo igrač koji je pozvao funkciju, a vraća 1 ako pobjeđuje samo igrač *K*. Vraća 0 ako nitko ne pobjeđuje, ali jedan od igrača može pobijediti uz konačno mnogo poziva funkcije *zamijeni_s_igračem.* Vraća -1 inače.

`varam` – U ruku ubacuje kartu koja se pojavljuje najviše puta umjesto bilo koje druge karte. Ukoliko se to ne može napraviti (tj. igrač već ima sve karte iste), funkcija ne radi ništa. Vraća *true* ako će igrač imati sve iste karte nakon poziva funkcije, *false* inače.

destruktor – Ispisuje sve karte u ruci, odvojene razmakom.

Napišite i neki main koji testira strukturu koju ste napravili.

Zadatak spremite pod imenom **`zadatak1.cpp`**

Napomene:

- Ne smijete koristiti STL (osim <string> i <iostream>)

Zadatak 11

Restorani su mjesta gdje ljudi najčešće odlaze trošiti novce kada ogladne. Odlaske unosimo tako da prvo unesemo cijenu objeda (`int`, nenegativan), ime osobe (`string`) i naziv restorana (`string`). Moguće je da ista osoba više puta odlazi u isti restoran. Unos završava kada za cijenu objeda unesemo broj 0.

Unesene podatke zapamtite u spremnik **`map<string, map<string, int> > m`**.

Program 1

Napišite program koji ispisuje naziv restorana u kojem je potrošeno najviše novca i koliko je potrošeno, te za svaki restoran naziv restorana, te ime osobe koja je tamo najviše potrošila i koliko je potrošila. Ukoliko ima više istih (za bilo koju od ovih stvari), ispišite bilo koju.

Primjer:

```
10 ivica
kodPere
```

Restoranu kodPere pridružujemo ivicu, kojem tamo pridružujemo broj 10

`m = {`**`kodPere: {ivica: 10}`** `}`

```
20 jozo bistro
```

Restoranu bistro pridružujemo jozu, kojem tamo pridružujemo broj 20

`m = {`**`bistro: {jozo:20},`** `kodPere: {ivica:10} }`

```
30 jozo kodPere
```

```
m = {bistro: {jozo:20},
     kodPere: {ivica:10, jozo:30} }
```

```
50 ivica
kodPere
```

```
m = {bistro: {jozo:20},
     kodPere: {ivica:60, jozo:30} }
```

```
10 ivica bistro
```

```
m = {bistro: {ivica:10, jozo:20},
     kodPere: {ivica:60, jozo:30} }
```

```
0
kodPere 90
bisto jozo 20
kodPere   ivica
60
```

Program 2

Napišite program koji ispisuje ime osobe koja je ukupno najviše potrošila i koliko je potrošila, te ime osobe koja je bila na objedima sa najviše različitih osoba (i imena svih tih osoba). Ukoliko ih ima više istih, ispišite bilo koju.

Primjer:

```
10 3 a b c
```

Osobi a pridružimo par ({b,c}, 10), b -> ({a,c}, 10) i c -> ({a,b},10)

```
m = {a: ({b,c}, 10}
     b: ({a,c}, 10}
     c: ({a,b}, 10}}
```

```
20 2 a d
```

Osobi a u skup dodajemo d, i uvećavamo cijenu za 20, a osobi d pridružujemo par ({a},20)

```
m = {a: ({b,c,d}, 30}
     b: ({a,c}, 10}
     c: ({a,b}, 10}
     d: ({a}, 20}}
```

```
30 2 b c
```

```
m = {a: ({b,c,d}, 30}
     b: ({a,c}, 40}
     c: ({a,b}, 40}
     d: ({a}, 20}}
```

```
0
b 40
```

b i c su potrošili po 40, pa ispisujemo npr. b

```
a b c d
```

a je bio na objedima s 3 različite osobe, dok su b i c s 2, a d s jednom

Program 3

Napišite program koji za svako jelo ispisuje (njegov naziv), koliko se toga jela ukupno pojelo, naziv restorana gdje ga se pojelo najviše i količinu. Ukoliko ih ima više istih, ispišite bilo koji.

Primjer:

```
3 kodPere a b c
```

Jelu a pridružujemo restoran kodPere, gdje se pojelo jedno to jelo, a isto i s jelima b i c

```
m = {a: {kodPere:1},
     b: {kodPere:1},
     c: {kodPere:1}}
```

```
3 bistro a b a
```

Jelu a pridružujemo restoran bistro, gdje su se pojela dva ta jela, a jelo b se u tom restoranu pojelo jedan put

```
m = {a: {bistro:2, kodPere:1},
     b: {bistro:1, kodPere:1},
     c: {kodPere:1}}
```

```
3 bistro a b c   m = {a: {bistro:3, kodPere:1},
                     b: {bistro:2, kodPere:1},
                     c: {bistro:1, kodPere:1}}

2 kodPere b b    m = {a: {bistro:3, kodPere:1},
                     b: {bistro:2, kodPere:3},
                     c: {bistro:1, kodPere:1}}

0
a 4 bistro 3
b 5 kodPere 3
c 2 bistro 1
```

Zadatak 12

Porezna uprava namjerava uvesti porez na nekretnine. Zbog toga žele sistematizirati podatke iz katastra. U katastru su zapisani objekti u obliku naziv objekta (string), veličina (int), te ime vlasnika (string). Podatke učitavamo sve dok se za naziv objekta ne unese riječ „0“ (nula).

Unesene podatke zapamtite u spremnik **`map<string, vector<pair<string, int> > > m`**.

Porezna uprava želi za svaku osobu ispisati naziv i veličinu najvećeg objekta (ako ih ima više najvećih, bilo kojeg od njih), na taj objekt vlasnih neće plaćati porez, te sve preostale objekte koje ta osoba ima, sa njihovim veličinama, te ukupnu veličinu svih preostalih objekata, na koje će ta osoba onda plaćati porez.

Primjer:

```
kuca 70 ivica
```
Ivica ima kuću od 70 kvadrata
```
m = {ivica: [(kuca, 70)] }
```

```
kuca 150 jozo
```
Jozo ima kuću od 150 kvadrata
```
m = {ivica: [(kuca, 70)], jozo: [(kuca,
150)] }
```

```
stan 50 ivica
```
Ivica ima i stan od 50 kvadrata
```
m = {ivica: [(kuca, 70), (stan, 50)]},
     jozo: [(kuca, 150)] }
```

```
kuca 100 ivica
```
Ivica ima još jednu kuću od 100 kvadrata
```
m = {ivica: [(kuca,70), (stan,50),
(kuca,100)],
     jozo: [(kuca, 150)] }
```

```
kuca 100 ivica
```
Ivica ima još jednu kuću od 100 kvadrata
```
m = {ivica: [(kuca,70),(stan,50),(kuca,100),
(kuca,100)],
     jozo: [(kuca, 150)]}
```

```
0
```

```
ivica: kuca 100 kuca 70 kuca 100 stan 50 220
jozo: kuca 150 0
```

Zadatak 13

Napišite parametriziranu strukturu "Novcici" koja predstavlja šest hrpa s novčićima na stolu jedne osobe. Na svakoj od šest hrpa nalazi se prirodan broj novčića (nikad neće biti nula, i na to ne treba paziti), koje su sve okrenute glavom prema gore ili pismom prema gore. Na različitim hrpama jedne osobe mogu se nalaziti novčići koji su različito okrenuti prema gore. Također, svaki novčić ima vrijednost. Pretpostavljamo da mu se vrijednost razlikuje ako je okrenut glavom prema gore (G) ili pismom prema gore (P), no svi novčići jedne osobe imaju te dvije vrijednosti iste. Te vrijednosti parametrizirane su nekim tipom *T* (int, double, float i sl.).

U ovom zadatku ne smijete koristiti STL. Unutar strukture **`Novcici`** definirajte funkcije:

konstruktor – Prima dva integera *a, b*, te dva podatka tipa T. Osoba čiji su ti novčići na stol postavlja redom hrpe od $|a|$, $|a+b|$, $|a+2b|$, ..., $|a+5b|$ novčića. Glavom prema gore okreće one za koje su vrijednosti *a, a+b, a+2b, ..., a+5b* pozitivne, ostale okreće prema dolje. Podatci tipa T redom označavaju vrijednost novčića okrenutog glavom prema gore i novčića okrenutog pismom prema gore. Primjer: za učitane vrijednosti „7 , -2, 1.5, 2.5“ osoba slaže 6 hrpa s 7, 5, 3, 1, 1, 3 novčića, okrenuti redom G,G,G,G,P,P. Ukupna vrijednost novčića na stolu je 34.

Uputa: u memoriji čuvajte vrijednosti kao u aritmetičkom nizu (bilo kojim redoslijedom), makar bile negativne, kako bi se znalo koja je hrpa kako okrenuta (u ovom primjeru pamtite 7,5,3,1,-1,-3).

`stavi_na_i` – Prima jedan cijeli broj *i* između 0 i 5. Na *i*-tu po visini hrpu (počevši s najvišom) postaviti jedan novčić. Primijetiti da to ne mora biti *i*-ta hrpa po redu kao u konstruktoru. Ukoliko više hrpa dijeli *i*-to mjesto, odaberite proizvoljnu. Funkcija vraća string „G“ ili „P“, ovisno o tome na koju su stranu okrenuti novčići na toj hrpi.

`okreni_jednu_hrpu` – U slučaju da su sve hrpe okrenute na istu stranu, funkcija ne radi ništa. Inače, pronalazi (bilo koju) hrpu koja je okrenuta drugačije od neke hrpe s najviše novčića te je okreće na stranu istu stranu kao ta maksimalna hrpa. Funkcija ne prima ništa, a vraća razliku u vrijednosti svih 6 hrpa prije i poslije transformacije (ukoliko hrpe sada vrijede više, broj na izlazu je pozitivan).

`izjednaci` – Ukoliko je moguće, rasporedite sve novčiće jednoliko po hrpama, tako da osoba na kraju ima jednako novčića kao i prije. Broj hrpa s novčićima okrenutim glavom prema gore (i pismom prema gore) neka ostane isti kao prije transformacije. Funkcija ništa ne prima. Ako je završila uspjehom, funkcija vraća nulu, u suprotnom vraća najmanji broj novčića koji bi se trebao maknuti s nekih hrpa da bi to postalo moguće.

`zamijeni_s_igracem` – Prima varijablu *N* tipa Novcici koja označava hrpe novčića neke druge osobe. Osoba koja je pozvala funkciju na kraju izvođenja funkcije imat će 6 hrpa novčića okrenutih na istu stranu, a to će biti postignuto zamjenama hrpa novčića između dvije osobe. Nijedan novčić neka ne bude preokrenut na drugu stranu. Od svih mogućih načina kako izvršiti ovu zamjenu, odaberite vama proizvoljan. Funkcija ne vraća ništa.

`bogatiji –` Prima varijablu *N* tipa Novcici koja označava hrpe istovjetnih novčića neke druge osobe. Vraća *true/false,* ovisno o tome ima li osoba koja je pozvala funkciju vrijednije hrpe od druge osobe.

destruktor – Ispisuje 6 visina hrpa novčića s predznakom koji označava na koju stranu su te hrpe okrenute, zapisane kao u memoriji, odvojene razmakom, bilo kojim redoslijedom.

Napišite i neki main koji testira strukturu koju ste napravili.

Napomene:

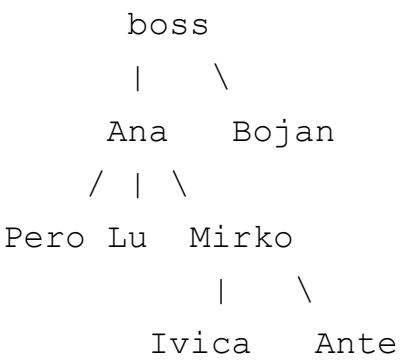
- Ne smijete koristiti STL (osim <string> i <iostream>).

Zadatak 14

U firmi radi nekoliko zaposlenika. Ime zaposlenika je tipa `string` i ne postoje dva zaposlenika s istim imenom. Svaki zaposlenik (osim zaposlenika s imenom `boss`) ima svog šefa.

Hijerarhija u firmi je preslikavanje (`map`) koje svakom zaposlenika pridružuje ime njegovog šefa (zaposleniku `boss` pridružen je prazan string).

a) Napišite `main` u kojem deklarirate varijablu koja sadrži hijerarhiju u firmi, implementiranu na gore opisan način. Koristite navedene STL-tipove. Nadalje, neka hijerarhija odgovara ovoj slici:



b) Napišite funkciju `maxSef` koji prima hijerarhiju u firmi. Funkcija treba pronaći onog zaposlenika (označimo ga sa *X*) koji je šef najvećem broju zaposlenika. Funkcija treba vratiti listu (`list`) svih zaposlenika kojima je *X* šef. U gornjem primjeru šef najvećem broju zaposlenika je `Ana`, pa treba vratiti listu koja se sastoji od `Lu`, `Mirko` i `Pero` (bilo kojim redom).

c) U firmi zaposlenici često šalju dopise jedan drugom. Da bi dopis stigao, mora proći putem po hijerarhiji. Na primjer, od `Ivice` do `Bojana` dopis treba proći kroz ruke još i `Mirku`, `Ani` i `bossu`. Da bi dopis došao od `Lu` do `Ante` treba proći kroz ruke još `Ani` i `Mirku`. Napišite funkciju `dopis` koja prima hijerarhiju i 2 imena zaposlenika koji šalju dopis jedan drugom. Funkcija vraća skup (`set`) svih dodatnih zaposlenika kojima dopis treba proći kroz ruke.

U glavnom programu pozovite funkcije iz (b) i (c) nad hijerarhijom zaposlenika definiranom u (a) i ispišite njihove povratne vrijednosti.

Zadatak 15

Napišite parametriziranu strukturu `nogometas` koja predstavlja igrače nogometa. Svaki nogometaš je predstavljen imenom, skupom pozicija na kojima igra i cijenom na tržištu. Cijena je parametrizirani tip. Moguće pozicije na kojima pojedini nogometaš može igrati su: *golman*, *obrana*, *veza*, *napad*. Sami odredite kako ćete prikazati skup pozicija. U ovom zadatku ne smijete koristiti STL (osim `<string>` i `<iostream>`).

Unutar strukture `nogometas` definirajte funkcije:

`jednak` – kažemo da su dva nogometaša jednaka ako im je jednaka cijena na tržištu i ako igraju na jednakom skupu pozicija. Funkcija vraća istinu ako su nogometaši jednaki, inače vraća laž.

`boljiOd` – kažemo da je nogometaš A bolji od nogometaša B, ako je A skuplji od B. Ukoliko su cijene jednake, tada je bolji onaj koji može igrati na više pozicija. Funkcija treba vratiti boljeg nogometaša. Ako niti jedan nogometaš nije bolji, vratite nogometaša s imenom "`Ne postoji`".

`dodajPoziciju` – nogometašu dodaje poziciju na kojoj može igrati.

`izbaciPoziciju` – nogometašu ukida poziciju na kojoj može igrati.

Implementirajte i barem 3 konstruktora. Destruktor neka ispiše informacije o objektu kojeg uništava.

Napišite i neki `main` koji testira strukturu koju ste napravili.

Rješenje spremite pod imenom `nogometas.cpp` .

Zadatak 16

Napišite parametriziranu strukturu `politicar` koja predstavlja političara u parlamentu. Svaki političar je predstavljen jedinstvenim imenom. Parametrizirani tip unutar strukture informacija je o političarevom broju javljanja za riječ. Političar može i upućivati replike nekom drugom političaru, i ima pravo na maksimalno 100 replika. Sami odredite kako ćete implementirati izrečene replike, tako da omogućite pravilno izvođenje dolje opisanih funkcija. U ovom zadatku ne smijete koristiti STL (osim `<string>` i `<iostream>`).

Unutar strukture `politicar` definirajte funkcije:

`jednak` – kažemo da su dva političara jednaka ako su se jednaki broj puta javljali za riječ i ako su uputili jednaki broj replika. Funkcija vraća istinu ako su političari jednaki, inače vraća laž.

`postenijiOd` – smatramo da je političar A pošteniji od političara B, ako mu je uputio manji broj "nepristojnih" replika, nego ovaj njemu. Replika se smatra nepristojnom ako sadrži neku od riječi "`konj`", "`laž`", "`majmun`" ili "`krađa`". Funkcija treba vratiti poštenijeg političara. Ako niti jedan političar nije pošteniji, vratite političara s imenom "`Nema`".

`uputiRepliku` – političar upućuje repliku nekom drugom političaru.

`ispisiReplike` – treba ispisati sve dosadašnje političareve replike. Ukoliko je replika nepristojna, umjesto teksta replike ispisuje se "`*****`".

Implementirajte barem 3 konstruktora. Destruktor neka ispiše sve replike političara kojeg se uništava.

Napišite i neki `main` koji testira strukturu koju ste napravili.

Rješenje spremite pod imenom `politicar.cpp` .

Zadatak 17

Napišite parametriziranu strukturu `dugovi` koja pamti do 100 dugova, na dolje opisani način. Svaki dug ima iznos (uvijek `int`) i datum (parametrizirani tip na koji se može ispisivati sa `cout` i uspoređivati). Funkcija

- `dodaj` ima parametre iznos i datum – dodaje dug koji je nastao određenog datuma. Ako je spremnik pun treba izbaciti najmanji dug (ili neki od najmanjih ako ih ima više) i dodati novi dug
- `oprosti` izbacuje sve dugove manje od navedenog iznosa, te vraća ukupno oprošteni dug
- `zastara` izbacuje sve dugove koji su bili prije navedenog datuma, te vraća ukupni iznos koji je otišao u zastaru
- `ukupno` vraća ukupni zbroj dugovanja
- destruktor treba ispisati sve dugove u obliku iznos, datum

Napišite i neki `main` koji testira strukturu sa tipom `string` - koji predstavlja datum u formatu `yyyymmdd`.

Zadatak spremite pod imenom **dugovi.cpp**.

Zadatak 18

Napišite program koji određuje sastav ručka u menzi. Ime jela se sastoji od jedne riječi (velika slova `A-Z`). Na početku se unosi sadržaj meina – imena jela – sve dok se ne unese string `"kraj"`. Nakon toga se unose imena jela koja su na tacni, sve dok se ne unese string `"kraj"`.

Program treba ispisati jela koja su na tacni, kao u primjeru dolje (meni napišite malim slovima `meni`). Ako se jelo nalazi u meniu, podrazumijeva se da je cijeli meni na tacni (ostala jela iz menia se uracunavaju u taj meni). Ako se neki jelo s menia ponavlja (više puta nego u meniu), računa se u (cijeli) novi meni.

Ulaz :	*Izlaz* :
`SARMA JUHA SALATA SALATA kraj`	`meni meni SOK`
`SARMA JUHA SALATA JUHA SOK kraj`	

Rješenje spremite pod imenom **klopa.cpp**.

Zadatak 19

U jednoj dalekoj zemlji PDV se obračunava na pomalo neobičan način. Mirko radi u državnoj financijskoj agenciji i mora napisati program koji računa koliko se PDV-a plati na svakom pojedinom računu. Svaki račun je spremljen u jednu listu `L`.

Svaki element liste je uređeni par koji opisuje jedan artikl na računu. Prvi element para je naziv artikla koji ima oblik `proizvod:grupa`, gdje su `proizvod` i `grupa` nizovi koji se sastoje samo od malih slova engleske abecede. Drugi par je cijena proizvoda (cijeli broj) koja ne uključuje PDV.

PDV se određuje ovako: za svaki račun se po abecedi sortiraju sve grupe koje se pojavljuju na računu. PDV na proizvode iz abecedno prve grupe je 5%, iz abecedno druge je 10%, iz treće 15% i tako dalje.

(a) Napišite glavni program u kojem ćete učitati podatke s jednog računa u varijablu `L`. Podaci će biti unošeni u formatu kao u donjem primjeru.

(b) Napišite funkciju `pdv` koja prima listu `L`, a vraća jedno preslikavanje (mapu) `M`. U tom preslikavanju, svakoj grupi pridružen je jedan par. Prvi člana para je ukupan iznos (onaj koji uključuje i PDV) koji treba biti plaćen za sve artikle koji pripadaju toj grupi. Drugi član para je skup koji se sastoji od svih proizvoda iz te grupe koji se nalaze na računu.

Možete pretpostaviti da će proizvodi imati različita imena.

U glavnom programu ispišite sadržaj varijable `M` na ekran, kao u primjeru dolje. Poredak niti broj decimala prilikom ispisa nisu bitni.

Smijete koristiti pomoćne *STL-containere* (vector, list, map, ...), ali ne i pomoćna polja!

Primjer:

Ulazni podaci	Izlazni podaci

`kruh:hrana 7`
`novine:tisak 8`
`mlijeko:hrana 6`
`knjiga:tisak 100`
`mis:racunala 50`
`sir:hrana 8`
`strip:tisak 150`
`kraj`
	`hrana 22.05 {kruh mlijeko sir}`
	`racunala 55.00 {mis}`
	`tisak 296.70 {knjiga novine strip}`

U gornjem primjeru, lista `L` izgleda ovako:

```
L = [ (kruh:hrana, 7), (novine:tisak, 8), (mlijeko:hrana, 6), (knjiga:tisak, 100),
      (mis:racunala, 50), (sir:hrana, 8), (strip:tisak, 150) ].
```

PDV na `hrana` je 5%, na `racunala` je 10%, a na `tisak` je 15%. Preslikavanje `M` izgleda ovako:

```
    M(hrana) = (22.05, {kruh, mlijeko, sir}), M(racunala) = (55.00, {mis}),
    M(tisak) = (296.70, {knjiga, novine, strip}).
```

Za učitavanje i ispisivanje koristite isključivo `cin` i `cout`.

Zadatak 20

U jednom društvu, prijatelji često jedni drugima posuđuju novac. Popis svih posudbi zapisan je u listi `L`.

Svaki element te liste je jedan uređeni par. Prvi element para je cijeli broj koji predstavlja posuđeni iznos. Drugi element para je ponovno jedan uređeni par: prvi član je ime osobe koja je posudila novac, a drugi član je ime dužnika.

(a) Napišite glavni program u kojem ćete učitati podatke o posudbama u varijablu `L`. Podaci će biti unošeni u formatu kao u donjem primjeru. Imena osoba se sastoje samo od velikih i malih slova, a ime osobe koja je posudila novac i ime dužnika odvojeni su točno jednim znakom minus.

(b) Uočimo da u listi `L` osoba A može imati više posudbi prema osobi B, te da posudbe mogu postojati i u suprotnom smjeru. Stoga sve dugove između A i B možemo "prebiti" i zamijeniti jednim dugom. Napišite funkciju `prebijanje` koja prima listu `L`, a vraća jedno preslikavanje `M`.

U tom preslikavanju svakoj osobi A je pridružen jedan skup. Elementi tog skupa su uređeni parovi. Prvi član para je ime neke osobe kojoj A duguje novce nakon prebijanja. Drugi element para je ukupan iznos duga. U glavnom programu ispišite sadržaj ovog preslikavanja na ekran, kao u primjeru dolje.

Smijete koristiti pomoćne *STL-containere* (vector, list, map, ...), ali ne i pomoćna polja!

Primjer:

Ulazni podaci	Izlazni podaci

`50 Ana-Mirko`
`20 Mirko-Petra`
`30 Mirko-Ana`
`40 Ana-Mirko`
`30 Petra-Ana`
`10 Mirko-Ana`
`20 Petra-Mirko`
`20 Slavko-Ana`
`kraj`
	`Ana: (Petra, 30) (Slavko, 20)`
	`Mirko: (Ana, 50)`
	`Petra:`
	`Slavko:`

U gornjem primjeru, lista `L` izgleda ovako:

```
L = [ (50, (Ana, Mirko)), (20, (Mirko, Petra)), (30, (Mirko, Ana)), (40, (Ana, Mirko)),
      (30, (Petra, Ana)), (10, (Mirko, Ana)), (20, (Petra, Mirko)), (20, (Slavko, Ana)) ],
```

a preslikavanje `M` ovako:

```
    M(Ana) = { (Petra, 30), (Slavko, 20) }, M(Mirko) = { (Ana, 50) },
    M(Petra) = {}, M(Slavko) = {}.
```

Za učitavanje i ispisivanje koristite isključivo `cin` i `cout`.

Zadatak 21

Jedna kraljevska obitelj je odlučila modernizirati pravila za određivanje nasljedstva. Umjesto da pravo nasljeđivanja dobije dijete koje je prvo rođeno, odlučili su da pravo dobije unuk ili unuka koji ima najviše *followera* na Twitteru.

Popis rodbinskih veza u toj obitelji zapisan je u preslikavanju `M`. U tom preslikavanju, svakoj osobi (označimo ju sa A) je pridružen jedan uređeni par. Prvi član tog para je ime jednog od roditelja osobe A (onog koji ima kraljevsko porijeklo). Drugi član tog para je cijeli broj koji predstavlja broj *followera* osobe A na Twitteru.

(a) Napišite glavni program u kojem ćete učitati podatke o rodbinskim vezama u varijablu `M`. Podaci će biti unošeni u formatu kao u donjem primjeru. Imena osoba se sastoje samo od velikih i malih slova, a ime osobe i njenog roditelja odvojeni su točno jednim znakom "veće".

(b) Napišite funkciju `kruna` koja prima preslikavanje `M`, a vraća jednu listu `L`.

Elementi liste `L` trebaju biti uređeni parovi. Prvi član para je ime osobe, a drugi član treba biti njezin unuk/unuka koji ima najviše *followera* na Twitteru. U listi se kao prvi članovi moraju pojaviti sve osobe iz preslikavanja `M` koje imaju unuke. Možete pretpostaviti da svi unuci neke osobe imaju različit broj *followera*. Poredak elemenata u listi je nebitan.

U glavnom programu ispišite sadržaj ovog preslikavanja na ekran, kao u primjeru dolje.

Uputa: Za svaku osobu pronađite njezinog djeda/baku (to je vrlo lako napraviti pomoću preslikavanja `M`), te provjerite je li ta osoba dosad najpopularniji unuk.

Smijete koristiti pomoćne *STL-containere* (vector, list, map, ...), ali ne i pomoćna polja!

Primjer:

Ulazni podaci	Izlazni podaci
`Charles>Elizabeth 100`	`(Elizabeth, Henry)`
`William>Charles 200`	`(Charles, Charlotte)`
`Henry>Charles 500`	`(William, Richard)`
`George>William 900`	`(Henry, Alexandra)`
`Charlotte>William 1500`	
`TheThirdChild>William 1100`	
`John>Henry 200`	
`Anabel>George 1200`	
`Richard>George 1500`	
`Alexandra>John 2700`	
`theend`	

U gornjem primjeru, preslikavanje `M` izgleda ovako:

```
M(Charles) = (Elizabeth, 100), M(William) = (Charles, 200), M(Henry) = (Charles, 500),
M(George) = (William, 900), M(Charlotte) = (William, 1500), ...
```

a lista `L` ovako:

```
L = [ (Elizabeth, Henry), (Charles, Charlotte), (William, Richard), (Henry, Alexandra) ] .
```

Za učitavanje i ispisivanje koristite isključivo `cin` i `cout`.

Zadatak 22

Implementirajte klasu `Pol3`, čiji objekti predstavljaju polinome sa cjelobrojnim koeficijentima stupnja najviše 3. . Klasa treba imati sljedeće konstruktore i operatore:

- `Pol3 P(a0,a1,a2,a3)`: stvara polinom `a0+a1*x+a2*x^2+a3*x^3`
- omogućiti poziv konstruktora sa jednim, dva ili tri argumenta. Jedan argument stvara polinom stupnja `0`, `a0+0*x+0*x^2+0*x^3`, dva argumenta polinom stupnja `1`, tri argumenta polinom stupnja `2`.

- Defaultni konstruktor stvara nulpolinom.
- `cout<<P:` ispis, u obliku `a0+a1*x+a2*x^2+a3*x^3`

- `P+Q, P-Q` : uobičajeno zbrajanje i oduzimanje polinoma
- `P*Q` : množenje modulo `x^4`, tj. pri uobičajenom množenju polinoma zanemarimo sve monome stupnja većeg od `3`. Primjer: `(1+2x+3x^2+x^3)*(1-2x+x^2+3x^3)=1+0x+0x^2+0x^3+7x^4+ +9x^5+3x^6` ali zanemarimo monome 4., 5., i 6. stupnja, pa `(1+2x+3x^2+x^3)*(1-2x+x^2+3x^3)=1`
- `P[ int i]` : vraća koeficijent uz `x^i` u polinomu `P`. Treba omogućiti mijenjanje tog koeficijenta
- `int i * P` : množenje polinoma cijelim brojem
- `P ( int i )` : izračunaa vrijednost polinoma u argumentu `i`
- ~P : unarni operator koji vraća derivaciju od P
- konverzija u `double` : vraća vrijednost integrala $\int_0^1 P(x)dx$
- `P^n` : potenciranje modulo `x^4` na `n`. potenciju `(int n, n>=0)`
- `!P`: računanje inverza s obzirom na množenje modulo `x^4`, ako inverz postoji. Npr. `!(1+2x+3x^2+x^3)=1-2x+x^2+3x^3` jer `(1+2x+3x^2+x^3)*(1-2x+x^2+3x^3)=1`

Ne morate pisati const kvalifikatore (umjesto const& možete prenositi po vrijednosti), i sve može biti public (dakle, ne trebate ni friend deklaracije). Omogućite ulančavanje gdje god je to moguće. Napišite i neki main koji testira bar 2 od gornjih operatora.

Zadatak 23

Implementirajte klasu `Kasica`, čiji objekti predstavljaju stanje u kasici malog Ivice. Ivica štedi u kovanicama od 5, 2, 1 kune i 50 lipa. Klasa treba imati sljedeće konstruktore i operatore:

- `Kasica K(pe, dv, je, po)`: stvara kasicu koja ima `pe` kovanica od 5kn, `dv` kovanica od 2kn, `je` kovanica od 1kn i `po` kovanica od 50lp.
- `Kasica K( string s, int br)`: stvara kasicu koja ima br kovanica s. Možete pretpostaviti da će s biti iz skupa "5kn" , "2kn" , "1kn" , "50lp"
- `cout<<K`: ispis, u obliku `pe*5+dv*2+je*1+po*.50=ukupno`
- `++K`: povećava broj kovanica od 5kn za jedan, omogućiti ulančavanje
- `K++`: povećava broj kovanica od 50lp za jedan
- `--K, K--`: analogno kao `++K` i `K++`, samo što smanjuje broj kovanica. Pazite da ne može ići u minus (tj. ako se -- poziva a broj odgovarajućih kovanica je već 0, ne događa se ništa).
- `K>>n` : Ivica pokušava platiti iznos od n kuna pohlepnim algoritmom. Ako može na taj način isplatiti iznos veći ili jednak n, i blagajnik mu vraća razliku također pohlepnim algoritmom (pretpostavljamo da blagajnik ima po volji mnogo svih kovanica). Npr. Ivica bi u situaciji da u kasici ima 3*5kn+3*2kn i plaća iznos 16, dao 5+5+5+2=17 i blagajnik bi mu vratio 1kn (iako bi pametnijim algoritmom 5+5+2+2+2 mogao platiti iznos bez razlike).
- `K<<n` : iznos u kasici se povećava pohlepnim algoritmom iz neke blagajne za koju pretpostavimo da ima dovoljno svih kovanica
- K1<<K2 Ivica je našao drugu kasicu i premješta sve iz nje u prvu.
- `K["5kn"], K["2kn"], K["1kn"], K["50lp"]`: vraća broj broj odgovarajućih kovanica, treba omogućiti mijenjanje tih podataka
- konverzija u `double`: trenutni iznos u kasici.
- konverzija u int: vraća broj transakcija u kasici. (K1<<K2 je transakcija za obe kasice).

Ne morate pisati const kvalifikatore (umjesto const& možete prenositi po vrijednosti), i sve može biti public (dakle, ne trebate ni friend deklaracije). Omogućite ulančavanje gdje god je to moguće. Napišite i neki main koji testira bar 2 od gornjih operatora.

Zadatak 24

Implementirajte klasu `Red`, čiji objekti predstavljaju red u banci:

- `Red Q(a)`: stvara prazan red kapaciteta `a`. Defaultni konstruktor neka postavi kapacitet na `20`.
- `Q << string S`: dodaje osobu `S` na kraj reda. `S` može (i ne mora) imati na kraju nekoliko kopija znakova '+' ili '-', koji označavaju prioritet osobe. Tako je `"baka+++"` prioriteta `3`, `"nepristojanLik--"` prioriteta `-2`, a `"slucajniProlaznik"` prioriteta `0`. `S` se pomiče prema naprijed po `Redu Q` dok god osoba ispred ima manji prioritet. Ako je `S` bila `kapacitet+1`. osoba po redu, jedna osoba (ona najmanjeg prioriteta) ispadne iz reda.
- Q1<<Q2 : prebaci sve osobe koje stanu sa Q2 u Q1, po redu. Pri tome u Q1 novo stanje također treba biti sortirano po prioritetu
- cout<< Q : treba ispisati na ekran sadržaj `Reda` od prve osobe na redu do zadnje. Npr ako su u Q osobe `"baka+++"`, `"nepristojanLik--"` i `"slucajniProlaznik"`, treba ispisati `"baka+++,slucajniProlaznik,nepristojanLik--"`
- `Q-= string S` : izbacuje osobu `S` iz reda `Q`. S se ne smatra posluženom.
- Q++, ++Q : posluži prvu osobu na redu, makne je iz reda, i pomjeri ostale za mjesto naprijed. Neka i prefiksni i postfiksni imaju isto ponašanje i omogućuju ulančavanje.
- `Q-= int n` : posluži prvih n osoba u redu. Ako n> broja osoba u redu, operator posluži onoliko koliko ih ima.
- `Q[int i]` : vraća osobu na mjestu i, omogućiti mijenjanje podataka. Ako se podatak promijeni (osoba ili prioritet) treba ažurirati stanje reda
- konverzija u `int` : vraća broj obrađenih osoba

Ne morate pisati const kvalifikatore (umjesto const& možete prenositi po vrijednosti), i sve može biti public (dakle, ne trebate ni friend deklaracije). Omogućite ulančavanje gdje god je to moguće. Napišite i neki main koji testira bar 2 od gornjih operatora.

Zadatak 25

Napišite parametriziranu strukturu `polje` koja ima može čuvati do 10x10 podataka određenog tipa (kojeg `cout` zna ispisati). Struktura predstavlja dvodimenzionalno polje i treba imati:

- Konstruktor s dva parametra (*m* i *n*). Prvi je najveći broj redova (≤10), a drugi najveći broj elemenata u jednom redu (≤10).
- `ubaci_red` – prima jedan argument – prvi element novog reda kojeg dodaje u strukturu. Funkcija vraća 1, ako je uspjela dodati novi red, a 0 ako nije (struktura je već sadržavala *m* redova)
- `ubaci` – prima indeks `i`, te novi element `k`. Na kraj `i`-tog reda strukture ubacuje element `k` (prvi element ima indeks 0). `i` će uvijek biti manji od trenutnog broja redova u strukturi. Funkcija vraća 1 ako je uspjela ubaciti element. Ako je taj red već sadržavao *n* elemenata, funkcija vraća 0.
- `izbaci_red` – prima indeks `i`, te izbacuje `i`-ti red iz strukture (prvi red ima indeks 0). `i` će uvijek biti manji od broja redova strukture. Redovi koji su bili iza njega, pomiču se za jedno mjesto naprijed.
- `izbaci(int i, int j)` – izbacuje `j`-ti element `i`-tog reda strukture (prvi element ima indeks 0). `i` će uvijek biti manji od broja redova strukture, a `j` manji od broja elemenata tog reda. Ako u strukturi nakon izbacivanja elementa nema niti jedan element, funkcija treba izbaciti taj red (te ostale pomaknuti).
- `izbaci_stupac` – prima indeks `i`, te izbacuje `i`-ti stupac iz strukture (prvi ima indeks 0), a elemente iza njega pomiče za jedno mjesto naprijed. Ukoliko u određenom retku nema toliko elemenata, funkcija s tim retkom ništa ne radi. Funkcija vraća koliko elemenata je uspjela izbaciti.
- `nadji(int i, int j)` – vraća `j`-ti element `i`-tog reda strukture (prvi element ima indeks 0). `i` će uvijek biti manji od broja redova strukture, a `j` manji od broja elemenata tog reda.
- Destruktor ispisuje sve elemente strukture, i to svaki red strukture u odvojeni redak na ekranu. Sve elemente jednog retka treba odvojiti zarezom, točno kao u primjeru dolje.

U istoj datoteci napišite i program za testiranje ove strukture sa tipom `string`.

Primjer:

`polje<int> p(2,3);`	
`p.ubaci_red(1);`	p={{1}}
`p.ubaci(0,2);`	p={{1,2}}
`p.ubaci_red(5);`	p={{1,2},{5}}
`p.ubaci(0,0);`	p={{1,2,0},{5}}
`p.ubaci(1,1);`	p={{1,2,0},{5,1}}
`p.izbaci(0,1)`	p={{1,0},{5,1}}
`cout<<p.nadji(1,0)`	Ispiše 5
`p.izbaci_red(0)`	p={{5,1}}
`p.ubaci_red(7);`	p={{5,1},{7}}
`p.ubaci(0,10);`	p={{5,1,10},{7}}
`cout<<p.izbaci_stupac(1);`	Ispiše se 1 (jer samo jedan red ima više od jednog elementa), p={{5,10},{7}}

	Ispiše se:
	`5,10`
	`7`

U rješenju ne smijete koristiti STL-spremnike, a rješenje spremite pod imenom `zadatak1.cpp`.

U istoj datoteci napišite i `main` za testiranje ove strukture sa tipom `string`.

Zadatak 26

Napišite parametriziranu strukturu `cudni` koja ima može čuvati do 100 podataka određenog tipa (kojeg `cout` zna ispisati). Struktura predstavlja niz u koji se elementi ubacuju i izbacuju na pomalo čudan način. Struktura treba imati sljedeće funkcije:

- Konstruktor s dva parametra tipa `int` (*m* i *n*). Prvi je maksimalan broj elemenata niza (≤100), a drugi prvi element kojeg treba izbaciti (<*m*). Niz je na početku prazan. Prvi element ima indeks 0.
- `izbaci` – izbacuje prvi element koji treba izbaciti iz strukture (na *n*-tom mjestu, prvi element ima indeks 0), elemente iza njega pomiče za jedno mjesto naprijed, uvećava *n* za 1, te vraća 1. Ako je u bilo kojem trenutku unutar ove funkcije *n* veći ili jednak broju elemenata, umanjite ga za trenutni broj elemenata. Ukoliko je struktura prazna, funkcija vraća 0.
- `ubaci` – ubacuje novi element na kraj niza i vraća 1. Ukoliko u nizu nema mjesta, treba izbaciti jedan element (na *n*-tom mjestu) i na njegovo mjesto staviti novi element, uvećati *n* (kao kod izbacivanja), te vratiti 0.
- `koliko` – prima jedan parametar, te vraća broj elemenata u strukturi koji su jednaki njemu.
- `nadji` – vraća koliko puta bi trebalo pozvati funkciju izbaci da bi se izbacili svi elementi koji imaju određenu vrijednost (međutim, ona ne izbacuje niti jedan element iz strukture). Ako taj element ne postoji u strukturi, funkcija vraća 0.
- destruktor treba ispisati sve elemente strukture, odvojene zarezom (točno kako piše u primjeru).

Napišite neki `main` za testiranje strukture sa tipom `string`.

Primjer:

`cudni<int> p(4, 3);`	p={}, n=3
`p.ubaci(7); p.ubaci(8); p.ubaci(9);`	p={7,8,9}, n=3, i funkcija vraća 1
`p.izbaci();`	n=3, a u strukturi imamo samo 3 elementa, pa je n=3-3=0, pa izbacujemo broj 7, funkcija vraća 1, p={8,9}, te n uvećavamo na broj 1.
`p.izbaci();`	p={8}, n=2, ali je veći od broja elemenata strukture, pa ga smanjujemo do 0.
`p.ubaci(8);p.ubaci(9);p.ubaci(10);`	p={8,8,9,10}, n je i dalje 0.
`p.koliko(8);`	vraća 2
`p.nadji(8);`	3 – prvim izbacivanjem bi izbacili nultu 8-icu, sljedećim broj 9, n bi bio 2, pa prelazi u nula, te bi još jednim izbacivanjem izbacili i prvu osmicu. Međutim funkcija ne izbacuje elemente iz strukture.
`p.ubaci(1);`	p={1,8,9,10}, n=1, vraća 0
`p.ubaci(2)`	p={1,2,9,10}, n=2, vraća 0
	Ispiše se: `1,2,9,10`

U rješenju ne smijete koristiti STL-spremnike, a rješenje spremite pod imenom `zadatak1.cpp`

Zadatak 27

Autoprijevoznik ima 10 autobusa s istim brojem sjedala kojima prevozi grupe turista. Na početku se na stajalištu nalazi jedan prazni autobus. Broj sjedala se može parametrizirati nekim cjelobrojnim tipom (`int`/`short`/`char`/`long long`…). Napišite parametriziranu strukturu `autobus` koja ima funkcije:

- konstruktor pomoću kojeg se zadaje broj sjedala u autobusu.
- `dolazi` – prima broj turista u grupi (neće biti veći od broja sjedala). Turisti ulaze u najprazniji autobus na stanici; ako je više takvih, ulaze u prvi po redu najprazniji, te funkcija vraća redni broj autobusa (prvi ima indeks 1). Ako grupa ne stane niti u jedan autobus na stanici, dovozi se novi, grupa ulazi u njega, te funkcija vraća 0. Možete pretpostaviti da će svi turisti stati u sve autobuse prevoznika.
- `koliko` – prima jedan parametar – indeks autobusa, te vraća koliko još ima slobodnih mjesta u određenom autobusu (indeks će uvijek biti broj od 1 do 10).
- `broj` – vraća broj autobusa na stanici.
- `uravnotezi` – preraspoređuje turiste po autobusima na stanici, tako da nakon poziva funkcije svi autobusi imaju podjednako turista (odstupanje smije biti najviše za 1!).
- destruktor ispisuje broj putnika po busevima, odvojenih zarezom, točno kao u primjeru dolje.

Napišite i neki `main` koji testira strukturu sa tipom `short`.

Primjer:

`autobus<int> p(50);`	
`cout<<p.dolazi(30);`	1 – 30 turista dolaze u prvi bus
`cout<<p.dolazi(40);`	0 – potrebno je dovesti novi bus
`cout<<p.dolazi(20);`	1 – turisti ulaze u prvi bus
`cout<<p.dolazi(20);`	0 – dolazi treći bus
`cout<<p.koliko(3);`	30 – u treći stane još 30 turista
`cout<<p.broj();`	3
`p.uravnotezi();`	Sada su u autobusima recimo po 36, 37 i 37 putnika
	Ispisuje se: `36,37,37`

U rješenju ne smijete koristiti STL-spremnike, a rješenje spremite pod imenom `zadatak1.cpp`.

Zadatak 28

Napišite parametriziranu strukturu `SInviders` koja predstavlja ekran iz klase igara Space Inviders. U svakom polju tablice dimenzija `r x s` (r,s ≤ 20) nalazi se slobodno polje ili protivnički brod. Igrač u svakom potezu bira stupac u kojem ispucava metke, te ako se u tom stupcu nalazi protivnički brod, uništava onaj koji se nalazi u najnižem retku. Svakih `s+1` poteza svi protivnički brodovi pomiču se za jedno mjesto prema dolje.

Implementirajte sljedeće članske funkcije:

- Konstruktor s pet parametara (`r,s,k,znak0,znak1`). Prva dva označavaju broj redaka i stupaca ekrana, redom. Treći označava da na početku igre protivnički brodovi ispunjavaju prvih `k` redova ekrana. Zadnja dva argumenta istog su tipa kojim je parametrizirana struktura, te označavaju simbol kojim prezentiramo slobodno polje, odnosno protivnički brod, redom.
- Destruktor ispisuje stanje ekrana, koristeći `znak0` i `znak1` koji su primljeni u konstruktoru. Također, ispisuje odgovarajuću poruku (sâmi ju smislite) ako je igrač pobijedio (u slučaju da više nema protivničkih brodova), izgubio (u slučaju da se barem jedan protivnički brod nalazi u najdonjem retku ekrana) ili da igra i dalje traje (u ostalim slučajevima).
- `pucaj` – prima indeks `i`. Igrač odigrava svoj potez: uništava najniže postavljeni brod u stupcu s indeksom `i`. Ako je `i<0` negativan ili je `i>=s` smatramo da je igrač odigrao svoj potez, ali bez uništavanja brodova.
- `bomba` – prima indeks `j`. Uništava sve brodove u `j`-tom retku, a sve brodove koji se nalaze u retcima ispod `j`-tog pomiče za jedno mjesto prema gore. Poziv ove funkcije ne smatra se potezom (u kontekstu pomicanja protivničkih brodova prema dolje svakh `s+1` poteza). Ne radi ništa ako je `j<0` ili `j>=r`.
- `ispis` – ne prima ništa. Ispisuje stanje ekrana, koristeći `znak0` i `znak1` koji su primljeni u konstruktoru.
- `je_li_igra_gotova` – ne prima ništa. Vraća `1` ako je igrač pobijedio, `-1` ako je izgubio, te `0` inače (vidi opis destruktora za detalje).

Primjer:

```
1   SInviders<char> M(5,3,2,'.','#');
2   M.pucaj(1);
3   M.ispis();
4   M.pucaj(-1);M.pucaj(3);
5   M.ispis();
6   M.pucaj(0);
7   M.ispis();
8   M.bomba(1);
9   M.ispis();
10  cout<<M.je_li_gotovo(); // 0
11  M.pucaj(2);
```

Ispis nakon linija 3 i 5:

```
###
#.#
...
...
...
```

Ispis nakon linije 9:

```
...
..#
...
...
...
```

Ispis nakon linije 7:

```
...
###
..#
...
...
```

Ispis destruktora:

```
...
...
...
...
...
Igrac je pobijedio.
```

U rješenju **ne smijete** koristiti STL-spremnike, te u istoj datoteci napišite i `main` za testiranje ove strukture s tipom `char`.

Zadatak 29

Jedan lanac supermarketa uvodi internet trgovinu. Zato žele stvoriti bazu podataka kako bi iskontrolirali koji korisnik ima u košarici koje artikle i u kojoj količini. U bazi pamtimo imena korisnika (`string`), artikle (`string`) i količinu naručenih artikala (`int`).

Podatke spremamo u `map < string , set < pair < string , int > > >` (`string` u domeni `map`-a označava ime korisnika, dok `string` u `pair`-u označava artikl u košarici tog korisnika).

Imena korisnika i artikala stringovi sa znakovima iz engleske abecede. Bazu stvaramo i kontroliramo upisujući s tipkovnice naredbe (vidjeti primjer na dnu):

- `<korisnik> <kolicina> <artikl>` – u košaricu korisnika korisnik ubacuje `<artikl>` u količini `<kolicina>` (ako korisnik u košarici nema taj artikl), odnosno uvećava broj tog artikla u njegovoj košarici za `<kolicina>`.
- `koliko <artikl>` – ispisuje ukupnu količinu naručenih artikala imena `<artikl>`, za sve korisnike.
- `ispis <korisnik>` – ispisuje košaricu korisnika `<korisnik>`, ispisujući redom artikl i količinu artikla, odvojene razmacima

Učitavanje s tipkovnice traje dok se ne učita `0`. Tada se na ekran ispisuju svi korisnici u bazi odvojeni razmacima, te se završava s izvođenjem programa.

Možete smatrati da se nijedan korisnik ni artikl neće zvati `koliko` ili `ispis`. Također, možete smatrati da će sve linije koje se učitaju s tipkovnice biti legalne i isključivo gore navedenih formata. Količine artikala bit će prirodni brojevi.

Primjer:

Upis s tipkovnice	Stanje u spremniku	Ispis na ekran
`Ivan 7 banana`	`Ivan → {(banana,7)}`	
`Marko 5 krumpir`	`Ivan → {(banana,7)},` `Marko → {(krumpir,5)}`	
`Ivan 2 krumpir`	`Ivan → {(banana,7), (krumpir,2)},` `Marko → {(krumpir,5)}`	
`koliko krumpir`	*(isto kao gore)*	`7`
`Ivan 2 banana`	`Ivan → {(banana,9), (krumpir,2)},` `Marko → {(krumpir,5)}`	
`ispis Ivan`	*(isto kao gore)*	`banana 9 krumpir 2`

`Ante 3 salama`	`Ante → {(salama,3)},`
	`Ivan → {(banana,9), (krumpir,2)},`
	`Marko → {(krumpir,5)}`	

`0`		`Ante Ivan Marko`

Rješenje spremite pod imenom `zadatak2.cpp`, i pošaljite ga na `mljulj@math.hr`. U mailu se obavezno potpišite.

Zadatak DZ 1

Napišite sučelje i implementaciju za strukture `stranica` i `knjiga`.
Svaki string je standardni C-ovski string. Svaka riječ je string (niz malih slova engleske abecede) duljine do 20 znakova.
Struktura `stranica` predstavlja stranicu knjige. Na svakoj stranici može biti po volji mnogo riječi, ali će među njima postojati **maksimalno 100 različitih**.

Struktura `knjiga` predstavlja kolekciju (od maksimalno 100) podataka tipa `stranica`.
Stanice su numerirane brojevima od 0 do 99 (možete pretpostaviti da će uvijek biti brojevi stranice u tom rasponu). Uočite da neke stranice knjige mogu nedostajati, odnosno, da stranice ne moraju biti numerirane redom.

Sučelje (deklaraciju) za obje strukture spremite u datoteku **`knjiga.h`**, a implementaciju u datoteku **`knjiga.cpp`**.

Strukture moraju imati sljedeće elemente (a smijete dodati i elemente po želji, a neke ćete i morati dodati):

`stranica` - Konstruktori

funkcija	opis djelovanja
`stranica();`	Stvara stranicu u kojoj nema teksta.
`stranica(char* tekst);`	Stvara stranicu čije riječi opisuje string tekst. Tekst se sastoji od riječi (odvojenih jednim ili više razmaka).

Funkcije članice

funkcija	opis djelovanja
`int sadrzi_rijec(char* r)`	Vraća broj pojavljivanja riječi r na stranici.
`int broj_rijeci()`	Vraća broj **različitih** riječi na stranici.
`int get_rijec(int n, char* r)`	Vraća broj pojavljivanja n-te riječi (riječi trebaju biti poredane po abecedi), te kopira sadržaj te riječi u riječ r. Abecedno prva riječ ima indeks 0. Ukoliko na stranici ima manje ili jednako od n riječi, funkcija vraća 0.
`int dodaj_rijec(char* r)`	Dodaje riječ na stranicu. Ukoliko na stranici ima manje od 100 različitih riječi, dodaje riječ, i vraća broj pojavljivanja te riječi na stranici, a ukoliko bi dodavanjem te riječi bilo više od 100 riječi, funkcija vraća 0.
`int izbaci_rijec(char* r)`	Izbacuje jedno pojavljivanje riječi r na stranici. Funkcija vraća koliko je riječi r preostalo na stranici. Ukoliko je preostali broj pojavljivanja riječi r jednak 0, riječ r se izbacuje u potpunosti pa se ukupan broj riječi na stranici smanjuje. Ukoliko riječi nije ni bilo, funkcija vraća -1.

`knjiga` - Konstruktori

funkcija	opis djelovanja
`knjiga()`	Stvara knjigu s praznim stranicama.

Funkcije članice

funkcija	opis djelovanja
`void postavi(int broj_stranice, stanica s)`	Dodaje stranicu s na zadano mjesto u knjigu. (Ako je neka stranica već ranije bila postavljena na to mjesto, zanemaruje to ranije postavljanje).
`stranica get_stranica(int broj_stranica)`	Vraća određenu stranicu.
`int izbaci_rijec(char* r)`	Izbacuje riječ r (u potpunosti) sa svih stranica knjige, te vraća koliko riječi je izbačeno.
`int rijec(char* r)`	Vraća redni broj stranice na kojoj se pojavljuje riječ r. Ukoliko se radi o prvoj pretrazi neke riječi, ili je promijenjena riječ pretrage, funkcija vraća redni broj prve (najmanji broj) stranice na kojoj se pojavljuje. Ukoliko ponovimo pretragu za istom riječi, funkcija vraća redni broj sljedeće stranice na kojoj se ona pojavljuje (neovisno o broju pojavljivanja riječi na prethodnoj stranici). Ukoliko riječ ne postoji, ili je već pronađeno zadnje pojavljivanje neke riječi, funkcija vraća -1. Možete pretpostaviti da između uzastopnih poziva funkcije s istom riječi neće biti ubacivanja/izbacivanja riječi/postavljanja stranica.

Primjer klijentskog programa

```
#include "knjiga.h"
#include <iostream>
using namespace std;

void ispisi_stranicu(stranica s) {
	int n = s.broj_rijeci();
	for (int i = 0; i < n; ++i) {
		char c[21];
		int x = s.get_rijec(i, c);
		cout << c << " " << x << endl;
	}
	cout << endl;
}

int main() {
	stranica s("ima neki tekst u kojem ima dvije iste rijeci");
	ispisi_stranicu(s);
/*
dvije 1
ima 2
iste 1
kojem 1
neki 1
rijeci 1
tekst 1
u 1
*/

	cout << s.izbaci_rijec("ima") << endl; // 1
	cout << s.izbaci_rijec("ima") << endl; // 0
	cout << s.izbaci_rijec("ima") << endl; // -1
```

```
        cout << s.dodaj_rijec("neki") << endl; // 2

        cout << s.dodaj_rijec("neka") << endl; // 1
        cout << s.sadrzi_rijec("abc") << endl; // 0
        cout << s.dodaj_rijec("abc") << endl; // 1
        cout << s.sadrzi_rijec("abc") << endl; // 1
        ispisi_stranicu(s);

/*
abc 1
dvije 1
iste 1
kojem 1
neka 1
neki 2
rijeci 1
tekst 1
u 1
*/

        knjiga k;

        k.postavi(2, s); // prve dvije stranice su prazne
        k.postavi(10, s); // i 10. stranica je ista 2.

        cout << s.izbaci_rijec("neki") << endl; // 1
        k.postavi(5, s); // sad je to 5. stranica

        cout << s.izbaci_rijec("neki") << endl; // 0
        k.postavi(3, s); // sad je to 3. stranica

        cout << k.rijec("neko") << endl; // -1

        cout << k.rijec("kojem") << endl; // 2
        cout << k.rijec("kojem") << endl; // 3

        cout << k.rijec("neki") << endl; // 2 .. krecemo od pocetka
        cout << k.rijec("neki") << endl; // 5 .. bez obzira sto imamo 2 te rijeci na
stranici 2, idemo dalje
        // sa stranice 3 je ta rijec izbacena, pa je sljedeca stranica 5.
        cout << k.rijec("neki") << endl; // 10

        cout << k.rijec("neki") << endl; // -1 .. rijeci vise nema u knjizi
        cout << k.rijec("neki") << endl; // -1

        cout << k.izbaci_rijec("neki") << endl; // 5

        return 0;

}
```

Opće napomene

- Struktura, funkcije i datoteke koje šaljete moraju se zvati *točno* onako kako je zadano u zadatku. Pazite na mala i velika slova!
- Trebate poslati samo sučelje i implementaciju. U datotekama koje šaljete *ne smije* se nalaziti funkcija `main()`!
- nijedna funkcija *ne smije* ništa učitavati s tipkovnice ili neke datoteke, niti išta ispisivati na ekran ili u neku datoteku.
- Svaki od main-ova pomoću kojih testiramo ispravnost vašeg programa neće pozivati sve gore navedene funkcije. Stoga, ako neku od funkcija ne znate napisati ipak možete dobiti

koji bod (u tom slučaju tu funkciju nemojte navesti niti u `.h` niti u `.cpp` datoteci ili napravite neku trivijalnu implementaciju).

Ispravnost implementacijâ koje napišete bit će provjerena tako da ćemo mi napisati razne klijentske programe koji će deklarirati nekoliko varijabli zadane strukture, i na njima pozivati funkcije koje ste trebali napisati. Ako se poslani programi ne budu uspješno povezivali (*linkali*) s našim klijentskim programima, smatrat će se neispravnima.

Neki klijentski programi provjeravat će samo neke jednostavnije funkcije, dok će neki provjeravati sve funkcije koje trebate napisati. Provjera je potpuno automatska, tako da je od presudne važnosti da se pridržavate specifikacije. Nepridržavanje lako može uzrokovati osvojenih 0 bodova iz zadaće!

Naravno, za provjeru radi li implementacija prije nego što je pošaljete, preporučuje se da je testirate pomoću nekog klijentskog programa. No taj klijentski program ne šaljete!

Zadatak DZ 2

Napišite sučelje i implementaciju za strukturu `relacija`.
Relacija predstavlja relaciju među brojevima 0…n-1 (n<=100).

Sučelje (deklaraciju) spremite u datoteku **`relacija.h`**, a implementaciju u datoteku **`relacija.cpp`**.

Struktura mora imati sljedeće elemente (a smijete dodati i elemente po želji, a neke ćete i morati dodati):

`relacija` - Konstruktori

funkcija	opis djelovanja
`relacija(int n);`	Na skupu 0…n-1 stvara praznu relaciju.

Funkcije članice

funkcija	opis djelovanja
`int velicina()`	Vraća broj n.
`int dodaj(int a, int b)`	Funkcija dodaje uređeni par (a,b) u relaciju r. Ako su a ili b >=n, funkcija vraća -1, ako su već bili u relaciji vraća 0, a inače vraća 1.
`int ukloni(int a, int b)`	Funkcija uklanja uređeni par (a,b) iz relacije r. Ako su a ili b >=n, funkcija vraća -1, ako nisu bili u relaciji vraća 0, a inače vraća 1.
`int rel(int a, int b)`	Ako su a i b u relaciji, funkcija vraća 1, inače vraća 0.
`int refleksivna()`	Vraća 1 ako je relacija refleksivna, inače 0.
`int simetricna()`	Vraća 1 ako je relacija simetrična, inače 0.
`int antisimetricna()`	Vraća 1 ako je relacija antisimetrična, inače 0.
`int tranzitivna()`	Vraća 1 ako je relacija tranzitivna, inače 0.
`int urelaciji(int a)`	Vraća najmanji broj koji je u relaciji s a. Ukoliko nitko nije u relaciji, ili je već pronađeni zadnji, ili je a>=n, funkcija vraća -1. Ukoliko se radi o prvoj pretrazi za nekim tko je u relaciji s a, ili je promijenjen element za čijim relacijama se traga, funkcija vraća najmanji broj koji je u relaciji s brojem a. Ukoliko ponovimo pretragu za istim brojem, funkcija vraća sljedeći broj koji je u relaciji s njim. Možete pretpostaviti da između uzastopnih poziva funkcije s istim brojem neće biti dodavanja/uklanjanja veza među brojevima (tj. dodavanjem/izbacivanjem neke veze pretraga počinje od početka).

Primjer klijentskog programa

```
#include "relacija.h"

#include <iostream>
using namespace std;

void ispis(relacija r) {
	for (int i = 0; i < r.velicina(); ++i) {
		for (int j = 0; j < r.velicina(); ++j)
			if (r.rel(i, j)) cout << "*"; else cout << "_";
			cout << endl;
	}
	cout << endl;
}

int main() {
	relacija r(5);
	cout << r.refleksivna() << r.simetricna() << r.antisimetricna() << r.tranzitivna()
<< endl; // 0111

	cout << r.dodaj(1, 2) << endl; // 1
	cout << r.dodaj(2, 4) << endl; // 1
	cout << r.dodaj(1, 2) << endl; // 0
	ispis(r);
/*
_____
__*__
___*
_____
_____
*/

	cout << r.refleksivna() << r.simetricna() << r.antisimetricna() << r.tranzitivna()
<< endl; // 0010

	cout << r.dodaj(2, 1) << endl; // 1
	cout << r.refleksivna() << r.simetricna() << r.antisimetricna() << r.tranzitivna()
<< endl; // 0000

	cout << r.dodaj(4, 2) << endl; // 1
	cout << r.refleksivna() << r.simetricna() << r.antisimetricna() << r.tranzitivna()
<< endl; // 0100

	for (int i = 0; i < 5; ++i) r.dodaj(i, i);
	cout << r.refleksivna() << r.simetricna() << r.antisimetricna() << r.tranzitivna()
<< endl; // 1100

	r.dodaj(1, 4); r.dodaj(4, 1);
	cout << r.refleksivna() << r.simetricna() << r.antisimetricna() << r.tranzitivna()
<< endl; // 1101


	ispis(r);

	while (true) {
		int t = r.urelaciji(2);
		if (t < 0) break;
		cout << t << " ";
	}
	cout << endl;
	// 1 2 4

	r.ukloni(4, 2);
	while (true) {
		int t = r.urelaciji(2);
		if (t < 0) break;
```

```
		cout << t << " ";
	}
	cout << endl;
	// 1 2 4

	r.ukloni(2, 2);
	while (true) {
		int t = r.urelaciji(2);
		if (t < 0) break;
		cout << t << " ";
	}
	cout << endl;
	// 1 4

	while (true) {
		int t = r.urelaciji(4);
		if (t < 0) break;
		cout << t << " ";
	}
	cout << endl;
	// 1 4
	return 0;

}
```

Opće napomene

- Struktura, funkcije i datoteke koje šaljete moraju se zvati *točno* onako kako je zadano u zadatku. Pazite na mala i velika slova!
- Trebate poslati samo sučelje i implementaciju. U datotekama koje šaljete *ne smije* se nalaziti funkcija `main()`!
- nijedna funkcija *ne smije* ništa učitavati s tipkovnice ili neke datoteke, niti išta ispisivati na ekran ili u neku datoteku.
- Svaki od main-ova pomoću kojih testiramo ispravnost vašeg programa neće pozivati sve gore navedene funkcije. Stoga, ako neku od funkcija ne znate napisati ipak možete dobiti koji bod (u tom slučaju tu funkciju nemojte navesti niti u `.h` niti u `.cpp` datoteci ili napravite neku trivijalnu implementaciju).

Ispravnost implementacijâ koje napišete bit će provjerena tako da ćemo mi napisati razne klijentske programe koji će deklarirati nekoliko varijabli zadane strukture, i na njima pozivati funkcije koje ste trebali napisati. Ako se poslani programi ne budu uspješno povezivali (*linkali*) s našim klijentskim programima, smatrat će se neispravnima.

Neki klijentski programi provjeravat će samo neke jednostavnije funkcije, dok će neki provjeravati sve funkcije koje trebate napisati. Provjera je potpuno automatska, tako da je od presudne važnosti da se pridržavate specifikacije. Nepridržavanje lako može uzrokovati osvojenih 0 bodova iz zadaće!

Naravno, za provjeru radi li implementacija prije nego što je pošaljete, preporučuje se da je testirate pomoću nekog klijentskog programa. No taj klijentski program ne šaljete!