1. Kreirati klasu Slagalica koja omogućava zadavanje tajne reči, slučajno permutovanje znakova tajne reči, kao i dve operacije – zameni(i, j), rotiraj(i, j). Operacija zameni(i, j) menja mesta *i* – tom i *j* – tom karakteru u transformisanoj reči, a rotiraj(i, j) rotira sve karaktere od *i* – tog do *j* – tog u desno ako je $i < j$, odnosno u levo ako je $i > j$.
   U glavnom programu omogućiti igranje sledeće igre – kompjuter učitava niz reči iz datoteke 'recnik.dat', odabira slučajno jednu od njih i permutuje joj karaktere. Posle toga prikazuje permutovanu reč korisniku i traži od njega komande. Posle svake komande treba odštampati promenjenu reč. Igra se završava kada igrač pogodi traženu reč ili posle 10 pokušaja.

2. Kreirati klasu `KompleksniBroj` koja omogućava rad sa kompleksnim brojevima. Klasa bi trebalo da ima:
   a. podatke `re` i `im`, za realni i imaginarni deo kompleksnog broja;
   b. konstruktor sa dva parametra koji postavlja vrednosti za realni i imaginarni deo kompleksnog broja;
   c. konstruktor bez parametara koji postavlja vrednost kompleksnog broja na 0;
   d. odgovarajuće getere i setere;
   e. javni metod `moduo()` koji kao rezultat vraća moduo kompleksnog broja;
   f. javne metode kojima se izvode osnovne operacije sa kompleksnim brojevima (sabiranje, oduzimanje, množenje, stepenovanje, deljenje).

3. Kreirati klasu `Razlomak` koja omogućava rad sa racionalnim brojevima. Klasa bi trebalo da ima:
   a. podatke `br` i `im`, za brojilac i imenilac razlomka;
   b. privatni metod skrati() koji dovodi razlomak na neskrativi oblik. Može se definisati i privatni metod `int nzd(int a, int b)` koji određuje najveći zajednički delilac dva cela broja;
   c. konstruktor sa dva parametra koji postavlja vrednosti za brojilac i imenilac. Voditi računa o tome da imenilac ne može biti 0;
   d. konstruktor bez parametara koji postavlja vrednost kompleksnog broja na 0. U tom slučaju je imenilac jednak 1;
   e. odgovarajuće getere i setere;
   f. javne metode kojima se izvode osnovne operacije sa razlomcima (sabiranje, oduzimanje, množenje, stepenovanje, deljenje).

4. Kreirati klasu `Automobil` koja ima:
   a. celobrojne podatke `trenutnaBrzina` i `maksimalnaBrzina` koji predstavljaju trenutnu i maksimalnu brzinu automobila u km/h. Trenutna brzina automobila ne može biti veća od maksimalne;
   b. realni podatak `predjeniPut` koji određuje koliko kilometara je automobil prešao;
   c. konstruktor sa jednim celobrojnim parametrom koji predstavlja maksimalnu brzinu automobila. Ona ne može biti negativna. Trenutna brzina i pređeni put se postavljaju na 0;

d. javni metod `void ubrzaj(int a)` koji povećava trenutnu brzinu za `a` km/h, ali ne preko maksimalne brzine;
e. javni metod `void uspori(int a)` koji smanjuje trenutnu brzinu za `a` km/h, ali ne ispod 0;
f. javni metod `void vozi(double t)` koji na promenljivu `predjeniPut` dodaje put koji auto pređe za `t` časova, vozeći trenutnom brzinom;
g. odgovarajuće getere i setere;
h. javni metod `int ucitajKomande(char * fileName)`, odnosno, u Javi, `int ucitajKomande(String fileName)` koji učitava i izvršava komande iz tekstualne datoteke čije je ime dato. Metod vraća 0 ako se pri čitanju desila neka greška, a inače 1. Komande su oblika `UBRZAJ A`, `USPORI A`, `VOZI T` i svaka je data u po jednom redu. U prvom redu datoteke se nalazi broj komandi.

5. Kreirati klasu `Robot` koja u svakom trenutku pamti poziciju robota u ravni. Klasa bi trebalo da ima metode `kreni(int t)`, `ubrzaj(int v)`, `uspori(int v)`, `levo()`, `levo(int x)`, `desno()`, `desno(int x)`. Robot se na početku nalazi u koordinatnom početku, a početni smer je u pozitivnom smeru *x* – ose. Trebalo bi omogućiti korisniku da pri kreiranju objekta zada početne koordinate robota, kao i početni smer. Brzina robota se meri u metrima u sekundi, a početna brzina je 0. Metod `kreni(t)` pokreće robota u trenutnom smeru *t* sekundi. Metod ubrzaj(v) (uspori(v)) povećava (smanjuje) brzinu robota za *v* metara u sekundi. Brzina robota nikada ne sme biti manja od 0 *m*/*s*, a ni veća od 30 *m*/*s*. Metodi `levo(x)` `desno(x)` menjaju smer kretanja robota za *x* stepeni u odgovarajuću stranu. Metodi `levo()` i `desno()` menjaju smer kretanja za 90 stepeni.

   Napisati metode `pozicijaX()` i `pozicijaY()` koji daju *x*, odnosno *y* koordinatu trenutne pozicije. Napisati metod `ucitajKomande(String filename)` koji iz tekstualne datoteke sa navedenim imenom učitava komande za robota i izvršava ih. U svakom redu datoteke se nalazi po jedna komanda. Komande mogu biti `KRENI #`, `UBRZAJ #`, `USPORI #`, `LEVO`, `LEVO #`, `DESNO`, `DESNO #` (# označava prirodan broj).

   U glavnom programu učitati početne koordinate robota, a zatim izvršiti spisak komandi iz datoteke 'robot.in'. Na kraju odštampati rastojanje robota od početne tačke.

6. (JAVA) Igra *Mastermind* se sastoji u tome da kompjuter slučajno generiše niz od *n* znakova koje bira od *k* vrsta znakova. Igrač zatim pogađa kombinaciju koju je kompjuter zamislio, a kompjuter posle svako pogađanja daje izveštaj koliko je znakova pogođeno, kao i koliko je znakova na svom mestu.

   Kreirati klasu `Mastermind` kojoj se može zadati broj različitih znakova *k* ($3 \le k \le 8$), kao i dužina niza koji se generiše *n* ($4 \le n \le 10$). Definisati metod `generisi()` koji generiše slučajni niz znakova, kao i metod `proba(...)` čiji je parametar niz znakova (znak može biti predstavljen cifrom ili posebnom klasom – bolje rešenje), a čiji je rezultat klasa (ili

struktura) koja sadrži podatke o broju pogođenih znakova i broju znakova koji su na svom mestu.

U glavnom programu od korisnika uzeti podatke za *n* i *k*, a zatim simulirati igru *Mastermind* sa datim podacima.

7. Kreirati klasu `VelikiInt` koja omogućava rad sa velikim celim brojevima. Definisati metode za sabiranje, oduzimanje, množenje, celobrojno deljenje i izračunavanje ostatka pri deljenju dva velika cela broja. Takođe, napisati metode za ove osnovne operacije kada je drugi operand običan ceo broj. Rezultat svakog od ovih metoda je `VelikiInt`. Napisati konstruktor bez parametara koji postavlja broj na vrednost 0, konstruktor sa celobrojnim parametrom, kao i konstruktore kojima su parametri string, odnosno niz cifara.

8. Kreirati klasu `Soba` koja predstavlja hotelsku sobu. Bitni podaci za hotelsku sobu su broj sobe, broj kreveta, cena, kao i to da li je soba zauzeta ili ne. Kreirati klasu `Hotel` koja omogućava unošenje podataka o sobama u hotelu (sa tastature ili iz datoteke). Definisati metod `pronadji(int n)` koja proverava da li u hotelu postoji soba u koju se može smestiti *n* osoba, pa ako postoji, kao rezultat daje broj najjeftinije sobe u koju taj broj osoba može da se smesti. Ako ima više takvih soba, rezultat treba da bude broj sobe koja ima najmanji broj kreveta, a ako i takvih ima više, onda od njih vratiti najmanji broj sobe. Ukoliko takva soba ne postoji, rezultat je –1. Napisati i metod `zarada(int n)` koji izračunava kolika je ukupna zarada od *n* – tokrevetnih soba u hotelu. Ako je $n = 0$, onda se računaju sve sobe, bez obzira na broj kreveta. Naravno, računaju se samo zauzete sobe.

9. Kreirati klasu TesterLozinke koja proverava da li dati string prihvatljiv za lozinku. String je prihvatljiv za lozinku ukoliko:
    a. ima bar 7 znakova,
    b. sadrži i velika i mala slova,
    c. sadrži bar jednu cifru.
    d. Klasa bi trebalo da omogući zadavanje stringa, bilo preko konstruktora ili setera, kao i da ima javni metod prihvatljiv(String s) koji kao rezultat daje jedan od stringova:
    e. ‘u redu’, ukoliko su svi uslovi zadovoljeni,
    f. ‘kratka lozinka’, ukoliko nije zadovoljen uslov a),
    g. ‘lozinka mora da sadrzi bar jedno malo i bar jedno veliko slovo’, ukoliko nije zadovoljen uslov b) (a jeste a)),
    h. ‘lozinka mora da sadrzi bar jednu cifru’, ukoliko nije zadovoljen uslov c) (a jesu a) i b)).

10. Kreirati klasu Kutija koja ima polja visina, sirina i duzina.
    Napisati odgovarajuće getere i setere, kao i konstruktor sa 3 parametra.

Napisati javni metod stajeU(Kutija k) koji proverava da li kutija može da stane u kutiju k. Jedna kutija može da stane u drugu, ako joj je osnova manja (po širini i dužini) od osnove druge kutije (s' tim što kutije mogu da se rotiraju oko vertikalne ose), a visina joj je manja od visine druge kutije. Računati da su strane kutija uvek paralelne.

Kreirati klasu NizKutija koja sadrži polje kutije koje predstavlja niz kutija. U ovoj klasi napisati metod proveri() koji proverava da li je moguće rasporediti kutije tako da svaka staje u sledeću.

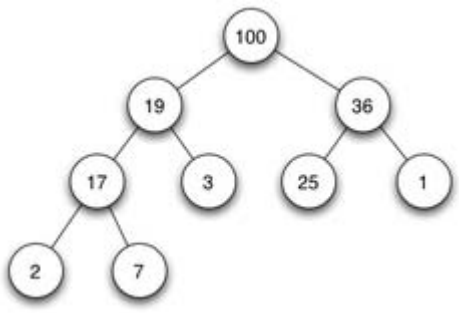
11. Kreirati klasu BinaryHeap koja kao podatak sadrži niz celih brojeva i koja omogućava rad sa strukturom binarni hip. Klasa bi trebalo da ima konstruktor i javne metode count(), get() i add(int x).

Objašnjenje: Binarni hip je struktura podataka koja se predstavlja binarnim stablom i ima osobinu da je svaki element veći (za maxHeap) od svih elemenata koji se u stablu nalaze ispod njega. Na slici je dat primer jednog binarnog hipa (slika levo).

Ovakva struktura se može predstaviti i nizom, tako što za *i* – ti element niza postavimo da je element levo od njega onaj sa indeksom (2*i* + 1), a desno element sa indeksom (2*i* + 2) (slika desno).

Element se u hip dodaje tako što se stavi na kraj niza, a onda se zamenjuje sa elementom iznad sebe, sve dok je veći od njega. Ovo bi trebalo da radi metod add(int x).

Metod get() bi trebalo da kao rezultat vrati maksimalni elemen hipa (null ako je hip prazan), a zatim da taj element obriše iz hipa. Prvi element se briše tako što se na njegovo mesto dovede najmanji element, a zatim se taj najmanji element pomera 'na dole' tako što se zamenjuje sa većim od dva elementa koji su direktno ispod njega, sve dok ne bude veći od oba elementa koji su ispod njega.

	**100** **19** **36** **17** **3** **25** **1** **2** **7** 0 1 2 3 4

	5 6 7 8

Metod count() bi trebalo da daje trenutni broj elemenata u hipu.

12. Napisati klasu Vektor za rad sa vektorima u ravni; vektor je odredjen svojom početnom i krajnjom tačkom. Definisati: konstruktor na osnovu dve zadate tacke (pocetna i krajnja tacka vektora), konstruktor na osnovu jedne zadate tacke (pocetna tačka je koordinatni pocetak, a krajnja tačka vektora je zadata), konstruktor kopije i metode za sabiranje i oduzimanje vektora (rezultat je vektor). Takođe implementirati i metod za translaciju vektora u zadatu tačku, metod za translaciju vektora za zadati vektor, metod koji nalazi centralno simetricni vektor datom vektoru u odnosu na zadatu tacku i metod za rotaciju vektora za dati ugao. U drugoj klasi implementirati main() funkciju koja sa standardnog ulaza ucitava vektor i zatim ucitava tacku u koju transliramo taj vektor, a zatim i ugao za koji ga dalje rotiramo; onda ucitavamo drugi vektor, pa tacku u odnosu na koju nalazimo centralno simetricni vektor tom vektoru. Na kraju ispisujemo zbir i razliku ovako dobijenih vektora.

13. Napisati sledeće klase (klase opremiti onim konstruktorima i destruktorom koji su potrebni za bezbedno korišćenje klasa):

- ***Ocena*** sadrži ceo broj u opsegu od 5 do 10. Vrednosti izvan opsega prilikom kreiranja promene se u najbližu prihvatljivu vrednost. Može da se dohvati vrednost ocene brojčano i slovima kao i da se ocena ispiše na glavnom izlazu kada se pišu oba oblika ocene (na primer: 10(deset)).
- ***Ispit*** sadrži šifru ispita od najviše 6 znakova i ocenu. Može da se učita šifra ispita i ocena i da se ispit ispiše na glavnom izlazu u obliku "*šifra*:*ocena*".
- ***Student*** ima ime (tekst proizvoljne dužine), broj indeksa (dugačak ceo broj po šemi ggggrrrr, gde su g i r cifre godine upisa i registarskog broja) i niz ispita zadatog kapaciteta (podrazumevano 40). Stvara se bez ijednog ispita posle čega ispiti mogu da se dodaju jedan po jedan. Povratna vrednost pri dodavanju ispita pokazuje uspeh dodavanja (tj. da li je bilo mesta u nizu ispita). Ne sme da se pravi kopija studenta. Može da se izračuna srednja vrednost ocena položenih ispita i da se student ispiše na glavnom izlazu u obliku "*ime*[*godUp*/*regBr*:*srOcena*]".

  Napisati na jeziku *C++* program koji napravi jednog studenta, dodaje mu tri ispita i ispiše ga na glavnom izlazu. Koristiti samo konstantne podatke (ne treba ništa učitavati s glavnog ulaza).

14. Sastaviti na jeziku *C++* sledeće klase (klase opremiti onim konstruktorima i destruktorom koji su potrebni za bezbedno korišćenje klasa):

- ***Datum*** se zadaje pomoću broja dana, meseca i godine. Može da se proveri da li tri cela broja predstavljaju ispravan datum, da se stvara datum na osnovu tri cela broja (podrazumevano 7.11.2005. – pogrešan datum prekida program), da se dohvataju delovi datuma, da se datum uporedi s drugim datumom (rezultat je <0, =0 ili >0, zavisno od toga da li je tekući datum pre, jednak ili posle zadatog datuma), da se datum pročita sa glavnog ulaza i da se datum ispiše na glavnom izlazu.
- ***Lista*** datuma se stvara prazna, posle čega se datumi dodaju jedan po jedan na kraj liste. Može da se odredi dužina liste, da se dohvati najkasniji datum u listi i da se lista ispiše na glavnom izlazu.

  Napisati na jeziku *C++* glavni program koji čitajući datume s glavnog ulaza napravi listu datuma (čitanje se završava prvim neispravnim datumom), ispiše na glavnom izlazu dobijenu listu kao i najkasniji datum i ponavlja prethodne korake sve dok ne pročita praznu listu.

15. Sastaviti na jeziku *JAVA* sledeće klase (klase opremiti onim konstruktorima i destruktorom koji su potrebni za bezbedno korišćenje klasa):

- ***Datum*** se zadaje pomoću broja dana, meseca i godine. Može da se proveri da li tri cela broja predstavljaju ispravan datum, da se stvara datum na osnovu tri cela broja (podrazumevano 7.11.2005. – pogrešan datum prekida program), da se dohvataju delovi datuma, da se datum uporedi s drugim datumom (rezultat je <0, =0 ili >0, zavisno od toga da li je tekući datum pre, jednak ili posle zadatog datuma), da se datum pročita sa glavnog ulaza i da se datum ispiše na glavnom izlazu.
- ***Lista*** datuma se stvara prazna, posle čega se datumi dodaju jedan po jedan na kraj liste. Može da se odredi dužina liste, da se dohvati najkasniji datum u listi i da se lista ispiše na glavnom izlazu.

Napisati na jeziku *JAVA* glavni program koji čitajući datume s glavnog ulaza napravi listu datuma (čitanje se završava prvim neispravnim datumom), ispiše na glavnom izlazu dobijenu listu kao i najkasniji datum i ponavlja prethodne korake sve dok ne pročita praznu listu.

16. Sastaviti sledeće klase (klase opremiti onim konstruktorima i destruktorom koji su potrebni za bezbedno korišćenje klasa):

- Materijalna ***tačka*** u prostoru se zadaje pomoću realne mase (podrazumevano 1) i tri realne koordinate (podrazumevano (0,0,0)). Može da se odredi rastojanje (r) do druge tačke, da se izračuna privlačna sila između tačke i zadate druge tačke ($F=\gamma \cdot m_1 \cdot m_2 / r^2$, $\gamma=6{,}67 \cdot 10^{-11}$) i da se tačka ispiše na glavnom izlazu.
- ***Niz*** materijalnih tačaka se stvara prazan zadatog početnog kapaciteta (podrazumevano 5), posle čega se tačke dodaju jedna po jedna na kraj niza. Ako se niz prepuni, kapacitet

mu se poveća za 5. Može da se dohvati broj tačaka u nizu, da se dohvati tačka u nizu koja najviše privlači zadatu tačku i da se niz ispiše na glavnom izlazu.

Napisati na jeziku *C++* program koji čitajući materijalne tačke s glavnog ulaza napravi niz materijalnih tačaka (čitanje se završava unosom negativne mase), ispiše na glavnom izlazu dobijeni niz kao i tačku koja najviše privlači tačku jedinične mase u koordinatnom početku i ponavlja prethodne korake sve dok ne pročita prazan niz (niz dužine 0).

17. Realizovati klasu trougao koja će kao podatke članove imati tri tačke koje predstavljaju objekte klase tačka. Klasa tačka treba da ima odgovarajuće konstruktore i destruktore kao i funkcije za računanje rastojanja dvije tačke i ugla između prave koju čine dvije tačke i x ose. Klasa trougao treba da posjeduje odgovarajuće konstruktore, destruktore kao i funkciju članicu koja će za tri date tačke provjeravati da li čine trougao ili su to tačke sa jedne prave i druga funkcija članica provjerava da li je trougao, na ovaj način zadat, jednakokraki. Napisati i glavni program u kojem će se unositi koordinate za željene tačke, inicijalizovati potrebni objekti i provjeravati da li date tačke čine trougao i da li je isti jednakokraki.

18. Projektovati na jeziku *Java* **paket klasa** sa sledćim opisom:

- Za apstraktnu ***funkciju*** može da se izračuna realna vrednost u nekoj tački x i može da se stvori funkcija koja predstavlja njen izvod.
- ***Monom*** je *funkcija* oblika ax^k (podrazumevano *x*), a njen izvod je kax^{k-1}. Konverzija u tip Sring je oblika ***a*x^k**, gde su *a* i *k* vrednosti parametara monoma.
- ***Eksponencijalna funkcija*** je funkcija oblika ae^{bx} (podrazumevano e^x), a njen izvod je abe^{bx}. Konverzija u tip Sring je oblika ***a*exp(b*x)**, gde su *a* i *b* vrednosti parametara eksponencijalne funkcije.
- ***Zbir*** funkcija je *funkcija* koja može da sadrži zadati broj (podrazumevano 2) funkcija. Stvara se prazan posle čega funkcije mogu da se dodaju jedna po jedna. Pokušaj stavljanja funkcije u pun zbir funkcija se prijavljuje izuzetkom tipa specijalne jednostavne klase. Vrednost zbira funkcije je zbir vrednosti sadržanih funkcija. Izvod zbira funkcija je zbir izvoda sadržanih funkcija. Konverzija u tip String je oblika **(***fun***)+**…**+(***fun***)**, gde su *fun* rezultati konverzije pojedinih sadržanih funkcija u tip String.

Sastaviti na jeziku *Java* klasu sa **glavnim programom** koji napravi jedan zbir funkcija kapaciteta koji se zadaje kao parametar glavnog programa, dodaje nekoliko funkcija čitajući potrebne podatke preko glavnog ulaza, ispiše dobijeni zbir funkcija i njegov izvod i posle vrši tabeliranje vrednosti zbira funkcija i njegovog izvoda za svako $x_{min} \le x \le x_{max}$ sa korakom Δx. Na raspolaganju stoji klasa Citaj u bezimenom paketu koja sadrži zajedničke metode za čitanje svih standardnih tipova podataka.

19. Projektovati sistem klasa sa sledećim opisom:

- ***Niz*** realnih brojeva može da se inicijalizuje zadatom dužinom (podrazumevano 10 elemenata, vrednosti elemenata su proizvoljne) i drugim nizom, da se uništi, da se dodeljuje vrednost jednog niza drugom (=), da se pristupi elementu sa zadatim indeksom ([]) i da se dohvati dužina niza (unarni +). U slučaju nedozvoljenog indeksa, prijavljuje se izuzetak celobrojnog tipa sa vrednošću jednakom tom nedozvoljenom indeksu.
- ***Funkcija*** je apstraktna klasa u kojoj je predviđeno izračunavanje vrednosti realne funkcije sa jednim realnim argumentom (()) i upisivanje simboličkog oblika te funkcije (na primer formulu) u neki izlazni tok (<<).
- ***Verižni razlomak*** je *niz* za koji može da se izračuna vrednost priložene *funkcije* $v(x)$, podrazumevano$v_0(x)$. Verižni razlomak se ispisuje u obliku niza njegovih koeficijenata, na primer: VRazl[a_0,a_1,...,a_{n-1}]. U slučaju pokušaja deljenja nulom, prijavljuje se izuzetak celobrojnog tipa sa vrednošću nula.

Sastaviti na jeziku *C++* glavni program koji pročita red verižnog razlomka *n*, stvori jedan verižni razlomak sa podrazumevanim koeficijentima, promeni vrednost nekih od koeficijenata čitajući preko glavnog ulaza parove $i - a_i$ sve dok ne pročita nedozvoljeni indeks, pročita vrednosti x_{min}, x_{max} i Δx, i na kraju, tabelira vrednost verižnog razlomka na glavnom izlazu za $x_{min} \leq x \leq x_{max}$ sa Δx korakom .

20. Projektovati na jeziku *C++* sistem klasa sa sledećim opisom:

- Apstraktni *podatak* može da se ispisuje na glavnom izlazu, da mu se formira kopija u dinamičkoj memoriji i da se uništava.
- *Skalar* je podatak koji ima neku realnu vrednost, može da se inicijalizuje običnom realnom vrednošću i da se dohvati njegova vrednost (+skal).
- *Niz* je podatak koji može da sadrži izvestan broj raznovrsnih podataka (uključujući i nizove), može da se inicijalizuje kao prazan niz zadatog kapaciteta ili da se inicijalizuje drugim nizom. Pored toga, postoji mogućnost dodele vrednosti jednog niza drugom (niz1=niz2), dohvatanje kapaciteta niza (+niz), dohvatanje vrednosti nekog elementa niza (niz[ind], greška je ako je indeks izvan dozvoljenog opsega ili ako je dato mesto prazno), stavljanje podatka na prvo slobodno mesto u nizu (niz+=pod, greška je ako u nizu nema slobodnog mesta), izbacivanje podatka sa datog mesta u nizu (niz-=ind, greška je ako je indeks izvan dozvoljenog opsega) i pražnjenje sadržaja niza (~niz). Greške signalizirati **izuzecima**.

Projektovati na jeziku *C++* glavni program koji pročita kapacitet niza, stvori prazan niz tog kapaciteta, napuni niz podacima koje čita preko glavnog ulaza (skalarnim podacima i nizovima koji sadrže samo skalarne podatke), ispiše sadržaj dobijenog složenog niza preko glavnog izlaza i ponavlja prethodne korake sve dok za dužinu niza ne pročita negativnu vrednost.

21. Projektovati na jeziku *C++* sistem klasa sa sledećim opisom:

- Svakom apstraktnom ***atomu*** može da se napravi kopija (klon) i da se izračuna celobrojna veličina koja predstavlja broj bajtova koje bi atom zauzeo u nekoj datoteci. Ova veličina ne može da se odredi operatorom sizeof, već se određuje na osnovu neophodnog sadržaja atoma u datoteci. Atom može da se ispiše na standardnom izlazu (operator <<).
- ***Znak*** je atom koji je opisan sa dva podatka: kodom znaka (karakter, jedan bajt), i kratkim celim brojem (dva bajta) kojim se opisuje stil znaka. Ispisuje se na standardnom izlazu po formatu "Z(*kod*, *stil*)".
- ***Piksel*** je atom opisan sa četiri bajta, od kojih prva tri predstavljaju komponente boje (crvenu, zelenu i plavu), a četvrti predstavlja faktor prozirnosti. Ispisuje se na standardnom izlazu po formatu "P(*crvena*, *zelena*, *plava*, *prozirnost*)".
- ***Element*** je atom koji sadrži niz atoma ograničenog kapaciteta. Stvara se prazan zadatog kapaciteta, a onda mu se dodaju atomi, redom (operator +=). Prekoračenje kapaciteta niza atoma izaziva izuzetak. Ispisuje se na standardnom izlazu u uglastim zagradama kao niz atoma, međusobno razdvojenih zarezima.
- ***Tekst*** je element koji sadrži niz znakova ograničenog kapaciteta i informaciju o dužini sadržaja (kratak ceo broj, dva bajta). Ispisuje se na standardnom izlazu po formatu "Tekst: (*dužina*) ", iza čega sledi niz znakova.
- ***Slika*** je element koji sadrži niz piksela ograničenog kapaciteta, kao i informacije o širini i visini slike, izražene u broju piksela (kratki celi brojevi po dva bajta). Stvara se prazna zadatog kapaciteta niza piksela i zadate širine. Ispisuje se na standardnom izlazu po formatu "Slika: (*širina*, *visina*) ", iza čega sledi niz piksela.
- ***Dokument*** je element koji ima svoje ime (konvencionalni niz karaktera završen znakom '\0'). Dokument može da sadrži tekstove, slike i druge dokumente. Dokument se ispisuje na standardnom izlazu po formatu "Dokument: <*ime*>", iza čega sledi niz elemenata.

Sastaviti na jeziku *C++* glavni program koji demonstrira formiranje jednog dokumenta sa nekoliko tekstova, slika i drugih dokumenata, zatim ispiše dokument i na kraju izračuna i ispiše veličinu dokumenta.

22. Napisati na jeziku *C++* sledeće klase (klase opremiti onim konstruktorima, destruktorom i operatorom za dodelu vrednosti, koji su potrebni za bezbedno korišćenje klasa):

- ***Boja*** se zadaje realnom talasnom dužinom izraženom u nanometrima u opsegu od 380 do 750 (podrazumevano 380), realnim zasićenjem u opsegu od 0 do 1 (podrazumevano 1) i realnim intenzitetom od 0 do 100 (podrazumevano 100). Može da se ispita da li su dve boje jednake (boja1==boja2) i da se boja upiše u izlazni tok (it<<boja) u obliku **(***talasna_dužina***,***zasićenje***,***intenzitet***)**.
- Obojen ***krug*** se zadaje poluprečnikom *r* (podrazumevano 1) i bojom (podrazumevano podrazumevana boja). Svaki krug ima jedinstven, automatski generisan identifikacioni broj *id*. Može da se dohvati boja, da se izračuna površina kruga, da se ispita da li su dva

kruga jednaka (krug1==krug2), da se ispita da li je jedan krug manji od drugog (krug1<krug2) i da se krug upiše u izlazni tok (it<<krug) u obliku **K*id*[*r*,*boja*]**.

- ***Niz*** obojenih krugova može da sadrži zadat broj krugova (podrazumevano 10). Stvara se prazan, posle čega se krugovi dodaju jedan po jedan (niz+=krug; prepunjavanje niza prekida program). Može da se dohvati kapacitet niza i broj popunjenih mesta, da se dohvati krug sa zadatim rednim brojem (niz[ind]; indeks izvan opsega prekida program), da se dohvati adresa prvog kruga zadate boje (niz[boja]; rezultat je 0 ako nema takvog kruga) i da se sadržaj niza upiše u izlazni tok (it<<niz), jedan krug po redu.

Napisati na jeziku *C++* program koji napravi i popuni jedan niz obojenih krugova, ispiše niz na glavnom izlazu, pronalazi najmanji krug zadate boje u nizu i ispiše pronađeni krug na glavnom izlazu. Koristiti fiksne parametre (ne treba ništa učitavati s glavnog ulaza).

23. Napisati na jeziku *C++* sledeće klase (klase opremiti onim konstruktorima, destruktorom i operatorom za dodelu vrednosti, koji su potrebni za bezbedno korišćenje klasa; greške prekidaju program):

- ***Domina*** sadrži dva celobrojna polja (p_1,p_2) u opsegu od 0 do $n-1$ (greška je ako se navede vrednost izvan opsega). Parametar *n*se ne određuje za svaku dominu posebno, već za sve zajedno i može da se postavlja i dohvata. Može da se dohvati vrednost prvog i drugog polja domine, da se izvrši međusobna zamena polja (~dom), da se odredi da li su dve domine jednake (dom1==dom2; pri čemu su domine (p_1,p_2) i (p_2,p_1) jednake), da se domina pročita iz ulaznog toka (ut>>dom) i da se upiše u izlazni tok (it<<dom) u obliku **(**p_1**,**p_2**)** gde su p_1 ip_2 polja domine.
- ***Tabla*** se predstavlja nizom različitih domina tako da su susedna polja susednih domina jednaka (tj. drugo polje prethodne domine jednako je prvom polju naredne domine). Stvara se prazna kapaciteta $n(n+1)/2$ domina, nakon čega joj se domine dodaju jedna po jedna (tab+=dom) na odgovarajućem kraju uz okretanje domina po potrebi. Može da se dohvati broj domina na tabli, da se ispita da li neka domina sme da se stavi na tablu, da se tabla isprazni i da se tabla upiše u izlazni tok (it<<tab). Greška je pokušaj stavljanja nedozvoljene domine.

Napisati na jeziku *C++* program koji, čitajući podatke s glavnog ulaza postavi opseg vrednosti polja domina (*n*), napravi odgovarajuću tablu, stavlja domine na tablu dok može, ispiše sadržaj table na glavnom izlazu i ponavlja prethodne korake sve dok za *n* ne pročita nedozvoljenu vrednost.

24. Napisati na jeziku *C++* sledeće klase (klase opremiti onim konstruktorima, destruktorom i operatorom za dodelu vrednosti, koji su potrebni za bezbedno korišćenje klasa):

- ***Etapa*** vožnje se zadaje pomoću realne dužine i brzine. Na jednoj etapi se ne menja brzina. Mogu da se dohvate atributi etape i da se izračuna vreme kretanja u etapi.

- ***Vožnja*** sadrži niz etapa zadatog kapaciteta (podrazumevano 10). Stvara se prazna posle čega se etape dodaju jedna po jedna. Ako se niz prepuni, program se prekida. Može da se odredi ukupna dužina vožnje, ukupno trajanje i srednja brzina kretanja u toku vožnje.
- Apstraktno ***vozilo*** ima zadatu sopstvenu težinu. Može da se dohvati naziv vrste vozila, da se odredi težina vozila i da se vozilo upiše u izlazni tok (it<<v). Piše se naziv vrste vozila i sopstvena težina vozila.
- ***Bicikl*** je vozilo.
- ***Kamion*** je vozilo zadate sopstvene težine i nosivosti. Stvara se bez tovara posle čega može da se doda tovar zadate težine (k+=t) i da se skine tovar zadate težine (k-=t). Ako se kamion pretovari, višak tereta se odbacuje. Skidanjem previše tovara težina tovara postane jednaka nuli. U izlazni tok se piše i trenutna težina tovara na kamionu.
- Generički ***niz*** može da sadrži elemente nekog tipa. Stvara se prazan zadatog kapaciteta (podrazumevano 20) posle čega elementi mogu da se dodaju jedan po jedan (niz+=e). Greška je ako se niz prepuni. Može da se dohvati broj elemenata u nizu, da se dohvati element zadatog rednog broja (niz[i]) i da se niz isprazni. Greška je ako se pokuša dohvatiti nepostojeći element.
- ***Trkački auto*** je vozilo koje sadrži generisan niz vožnji. Stvara se s praznim nizom kapaciteta 10 vožnji. Može da se započne nova vožnja kapaciteta 100 etapa, da se poslednje započetoj vožnji doda nova etapa, da se odredi vožnja sa najvećom srednjom brzinom. U izlazni tok se piše i dužina vožnje sa najvećom srednjom brzinom.

Sastaviti na jeziku *C++* program koji napravi trkački auto sa dve vožnje koje sadrže po tri etape, ispiše auto na glavnom izlazu i za vožnju s najvećom srednjom brzinom ispiše dužinu, trajanje i srednju brzinu.

25. Korišćenjem priloženih gotovih klasa tačaka u ravni i geografskih mesta čija je kontura zadata nizom tačaka u ravni, sastaviti na jeziku *C++* sledeće klase (klase opremiti onim konstruktorima, destruktorom i operatorom za dodelu vrednosti, koji su potrebni za bezbedno korišćenje klasa; greške prijavljivati izuzecima tipa klasa koje su osposobljene za ispisivanje teksta poruke):

- Apstraktni ***čvor*** grafa ima jednoslovnu oznaku. Može da se dohvati oznaka čvora, da se odredi realna veličina čvora i da se oznaka čvora upiše u datoteku (dat<<cvor).
- ***Geografski čvor*** je čvor koji sadrži jedno geografsko mesto. Veličina čvora je obim konture mesta. Može da se dohvati centar sadržanog mesta. U datoteku se piše ime mesta i obim konture.
- Apstraktna ***grana*** grafa ima jednoslovnu oznaku i spaja dva čvora grafa. Može da se dohvati oznaka grane, da se odredi realna dužina grane i da se grana upiše u datoteku (dat<<grana) u obliku *ozn***(***poc***,***kra***)**, gde su *ozn* – oznaka grane a *poc* i *kra* – oznake početnog i krajnjeg čvora grane. Grana ne sme da se kopira.
- ***Geografska grana*** je grana koja spaja dva geografska čvora. Dužina grane je rastojanje između centara mesta na krajevima grane.
- Generički ***niz*** se stvara prazan zadatog kapaciteta (podrazumevano 10) posle čega se elementi dodaju jedan po jedan (niz+=elem; prepunjavanje niza je greška). Može da se

dohvati broj elemenata niza i da se pristupa elementu sa datim rednim brojem (niz[ind]; greška je ako je indeks izvan opsega).

- ***Geografska karta*** sadrži niz geografskih čvorova i niz geografskih grana. Stvara se prazan sa zadatim kapacitetima za nizove čvorova i grana posle čega se čvorovi i grane dodaju pojedinačno. Može da se dohvati čvor, odnosno grana sa datom oznakom (greška je ako ne postoji čvor, odnosno grana sa tom oznakom). Karta ne sme da se kopira niti da se promeni na bilo koji način, osim dodavanjem čvorova i grana.

26. Sastaviti na jeziku *C++* sledeće klase (klase opremiti onim konstruktorima, destruktorom i operatorom za dodelu vrednosti, koji su potrebni za bezbedno korišćenje klasa; u slučaju greške prekidati program):

- Apstraktna teritorijalna ***jedinica*** ima naziv (znakovni niz). Može da se dohvati jednoslovna oznaka vrste, da se odredi broj stanovnika i da se upiše u izlazni tok (it<<jed) u obliku *naziv*:*vrsta*:*brojSt*:.
- ***Naselje*** je jednostavna teritorijalna jedinica u kojoj živi zadati broj stanovnika. Oznaka vrste je '**N**'.
- Apstraktna ***oblast*** je teritorijalna jedinica koja sadrži zadat broj drugih teritorijalnih jedinica. Stvara se prazna nakon čega joj se jedinice dodaju jedna po jedna (obl+=jed; prekoračenje kapaciteta je greška) uz proveru da li se jedinica sme dodati. Proveru određuju konkretne oblasti. Dodata jedinica ne postane vlasništvo oblasti već se samo pamti njena adresa (pokazivač). Broj stanovnika oblasti je jednak zbiru broja stanovnika sadržanih jedinica. Može da joj se odredi površina. U izlazni tok se piše u obliku*naziv*:*vrsta*:*brojSt*:*povrs*[*jed*,…,*jed*], gde jed šredstavlja rezultat pisanja jedne jedinice.
- ***Opština*** je oblast koja sadrži samo naselja i ima zadatu površinu. Oznaka vrste je '**O**'. ***Okrug*** je oblast koja sadrži samo opštine. Površina oblasti je jednaka zbiru površina sadržanih opština. Oznaka vrste je '**K**'. Pokušaj dodavanja neodgovarajuće jedinice je greška.

Napisati na jeziku *C++* program koji formira primer jednog okruga sa dve opštine, od kojih svaka opština ima po dva naselja, a zatim ispiše okrug na glavnom izlazu. Koristiti konstantne podatke (ne treba ništa čitati s glavnog ulaza).

27. Projektovati na jeziku *C++* sledeći sistem klasa (klase opremiti onim konstruktorima, destruktorom i operatorom za dodelu vrednosti, koji su potrebni):

- Zarubljena ***kupa*** se zadaje pomoću poluprečnika donje (r_1) i gornje (r_2) osnovice i visine (h) sa podrazumevanim vrednostima$r_1 = 2$, $r_2 = 1$, $h = 1$. Mogu da se dohvate dimenzije kupe, da se izračuna zapremina kupe ($V = \pi h (r_1^2 + r_2^2 + r_1 r_2) / 3$), da se ispita da li je zapremina jedne kupe manja od zapremine druge kupe (k1<k2), da se dimenzije kupe pročitaju iz datoteke (dat>>k) i da se kupa upisuje u datoteku (dat<<k) u obliku **K(**r_1**,**r_2**,**h**)**.
- ***Niz*** zarubljenih kupa može da sadrži zadati broj (podrazumevano 5) kupa. Niz se stvara prazan, posle čega se kupe mogu dodavati iza poslednjeg popunjenog mesta (niz+=k),

dok se niz ne napuni (ako se niz prepuni, program se prekida). Može da se dohvata kapacitet niza, broj popunjenih mesta i može da se ispita da li je niz popunjen do kraja. Kupe sadržane u nizu mogu da se dohvataju (niz[i]) i da se pronalazi indeks kupe najmanje zapremine (!niz).

Sastaviti na jeziku *C++* glavni program koji sa glavnog ulaza pročita niz zarubljenih kupa, pronalazi kupu najmanje zapremine u nizu, ispiše pronađenu kupu i njenu zapreminu na glavnom izlazu i ponavlja prethodne korake dok za dužinu niza ne pročita nedozvoljenu vrednost.

28. Napisati na jeziku *C++* sledeće klase (klase opremiti onim konstruktorima, destruktorom i operatorom za dodelu vrednosti, koji su potrebni za bezbedno korišćenje klasa; greške prijavljivati izuzecima tipa jednostavnih klasa koje su opremljene pisanjem teksta poruke):

- ***Lista*** sadrži neograničen broj podataka nekog tipa. Može da se doda jedan podatak na kraj liste, da se uzima jedan podatak s početka liste i da se ispita da li je lista prazna. Greška je ako se pokuša uzeti podatak iz prazne liste.
- Apstraktan ***element*** ima naziv modela (niska znakova) i realnu težinu koji mogu da se dohvate. Može da se dohvati naziv vrste elementa (niska znakova) i da se element upiše u izlazni tok (it<<elem) u obliku **(***model***,***vrsta***,***težina***)**.
- ***Motor*** je element koji ima broj cilindara i jednoznačan automatski generisan celobrojni identifikator motora. Naziv vrste je **motor**. U izlazni tok se piše u obliku **(***model***,***vrsta***,***težina***)[***idBroj***,***brojCilindara***]**.
- ***Šasija*** sa karoserijom je element koji ima jednoznačan automatski generisan celobrojni identifikator šasije i boju iz skupa: bela, žuta, crvena, zelena, plava i crna. Naziv vrste je **sasija**. U izlazni tok se piše u obliku**(***model***,***vrsta***,***težina***)[***idBroj***,***bojaSlovima***]**.
- ***Automobil*** sadrži šasiju sa karoserijom i motor. Greška je ako se pokuša sklopiti od elemenata različitog modela. Može da se upiše u izlazni tok (it<<auto) u obliku **auto{***šasija***,***motor***}**.
- Proizvodna ***traka*** sadrži listu pokazivača na automobile. Može da se stavi (traka<<&auto) i da se uzme (traka>>&auto) jedan automobil sa trake. Traka ne može da se kopira ni na koji način.

Napisati na jeziku *C++* program koji napravi proizvodnu traku, stavi na nju tri automobila i na kraju uzme sa nje dva automobila i ispiše ih na glavnom izlazu. Koristiti fiksne parametre (ne treba ništa učitavati s glavnog ulaza).

29. Sastaviti na jeziku *C++* sledeće klase (klase opremiti onim konstruktorima, destruktorom i operatorom za dodelu vrednosti, koji su potrebni za bezbedno korišćenje klasa):

- ***Osoba*** ima ime i godine starosti. Može da se ispita da li je jedna osoba starija od druge (osoba1>osoba2) i da se osoba upiše u izlazni tok (it<<osoba) u obliku *ime***(***god***)**, gde su: *ime* – ime osobe i *god* – godine starosti. Osoba ne sme da se kopira ni na koji način.

- ***Student*** je osoba koji može da polaže najviše zadati broj ispita (podrazumevano 20) za koje se pamte samo ocene. Stvara se bez ijedne ocene, posle čega se ocene dodaju jedna po jedna (student+=ocena). Pokušaj dodavanja previše ocena prekida program. Može da se ispita koliko ocena može još da se doda, da se izračuna srednja vrednost položenih ispita. U izlazni tok se piše u obliku*ime***(***god***)/***sred*, gde je *sred* – srednja ocena položenih ispita.
- ***Imenik*** može da sadrži zadat broj (podrazumevano 10) osoba proizvoljne vrste. Stvara se prazan posle čega se osobe dodaju jedna po jedna (imenik+=&osoba). Prepunjavanje imenika prekida program. Može da se dohvati kapacitet imenika, broj osoba u imeniku, da se imenik uredi po opadajućem redosledu starosti osoba u njemu i da se upiše u izlazni tok (it<<imenik), tako što se svaka osoba piše u posebnom redu. Imenik ne sme da se kopira ni na koji način.

Napisati na jeziku *C++* **program** koji čitajući potrebne podatke s glavnog ulaza napravi imenik koji sadrži izvestan broj osoba proizvoljne vrste, uredi imenik po opadajućem redosledu starosti osoba u njemu i ispiše imenik na glavnom izlazu.

30. Sastaviti na jeziku *C++* sledeće klase (klase opremiti onim konstruktorima, destruktorom i operatorom za dodelu vrednosti, koji su potrebni za bezbedno korišćenje klasa; u slučaju greške prekidati program):

- ***Osoba*** ima zadato ime proizvoljne dužine (podrazumevano prazno) i zadatu težinu (podrazumevano 0). Može da se dohvati ime i težina i da se osoba upiše u izlazni tok (it<<osoba) u obliku *ime***(***težina***)**.
- ***Vozilo*** ima zadatu sopstvenu težinu i zadatog vozača (osobu). Vozilo ne sme da se kopira ni da se dodeljuje. Može da se odredi ukupna težina vozila i da se vozilo upiše u izlazni tok (it<<vozilo) u obliku **[***vozač***,***sopTež***]**.
- ***Teretno vozilo*** je vozilo koje ima zadatu nosivost. Može da se utovari (vozilo+=tezina) i da se istovari (vozilo-=tezina) teret zadate težine (greška je ako se vozilo pretovari pri utovaru, odnosno ako na vozilu nema dovoljno tereta pri istovaru). U izlazni tok se piše u obliku **[***vozač***,***sopTež***](***tovar***/***nosivost***)**.
- ***Putničko vozilo*** je vozilo koje ima zadat maksimalni broj putnika (osoba). Može da se dohvati trenutni broj putnika u vozilu, da u vozilo ulazi novi putnik (vozilo+= putnik; greška je ako je vozilo puno) i da iz vozila izlazi putnik sa datim rednim brojem (preostali putnici se pomeraju tako da popune upražnjeno mesto; greška je ako traženi putnik ne postoji). U izlazni tok se piše u obliku **[***vozač***,***sopTež***]{***putnik***|**…**|***putnik***}**.

Napisati na jeziku *C++* program koji napravi niz od 2 vozila: jednog teretnog sa nešto tereta i jednog putničkog sa jednim putnikom i posle ispiše taj niz na glavnom izlazu zajedno sa njihovim težinama. Koristiti konstantne parametre (ne treba ništa učitavati s glavnog ulaza).