Zadaci za Tutorijal 7.

NAPOMENA: Studenti bi trebali da razmisle o zadacima koji će se raditi na tutorijalu prije nego što dođu na tutorijal, tako da već u startu imaju osnovne ideje kako riješiti zadatke. U suprotnom, rad na laboratorijskim vježbama neće biti produktivan. Zadatke koje studenti ne stignu uraditi za vrijeme tutorijala, trebali bi ih samostalno uraditi kod kuće. Bez obzira što tutorijal izgleda glomazan, jer ima 10 zadataka, zadaci 2, 3, 4 i 5 su samo kratke prepravke Zadatka 1 (prepravlja se svega nekoliko linja koda), dok se funkcije u zadacima 7, 8, 9 i 10 (pa i u Zadatku 6 ukoliko se mudro napiše) implementiraju u ne više od 4 linije koda (ponekad i u samo jednoj liniji), mada je u nekim zadacima prilična "mozgalica" doći do rješenja.

1. Napišite program koji traži od korisnika da unese niz rečenica, pri čemu se broj rečenica prethodno unosi sa tastature (rečenice neće biti duže od 1000 karaktera). Za svaku unesenu rečenicu dinamički alocirajte prostor tačno onolike veličine koliko je minimalno potrebno da se zapamti ta rečenica, uz vođenje evidencije o adresi svake alocirane rečenice u dinamički alociranom nizu pokazivača na početke svake od rečenica (kojem se također pristupa preko odgovarajućeg pokazivača). Nakon toga, treba ispisati unesene rečenice sortirane u abecedni poredak (tačnije rečeno, u rastući poredak po ASCII kodovima). Sortiranje obavite *ručno*, nekim od jednostavnih postupaka za sortiranje koji Vam je poznat (npr. BubbleSort, SelectionSort, itd.). Drugim riječima, nemojte koristiti gotove funkcije za sortiranje, poput funkcije "`sort`" iz biblioteke "`algorithm`". Dijalog između korisnika i programa treba da izgleda poput sljedećeg:

```
Koliko zelite recenica: 3
Unesite recenice:
Otiso si sarme probo nisi
Hop cup na kalup cetir babe jedan zub
I jogurt ujutro goji
Sortirane recenice:
Hop cup na kalup cetir babe jedan zub
I jogurt ujutro goji
Otiso si sarme probo nisi
```

   U slučaju da neka od alokacija ne uspije, program treba tog trenutka da ispiše tekst "Problemi sa memorijom" i da prekine dalji rad. Svu memoriju koja je zauzeta tokom rada programa treba osloboditi prije završetka programa, bez obzira da li je program završio rad regularno, ili zbog bacanja izuzetka.

2. Izmijenite prethodni program, ali tako što ćete umjesto ručnog sortiranja koristiti funkciju "`sort`" iz biblioteke "`algorithm`", uz pogodnu funkciju kriterija implementiranu kao lambda funkcija.

3. Izmijenite prethodni program (dakle, program iz Zadatka 2), tako što ćete rečenice umjesto u dinamički alociranim nizovima znakova čuvati u dinamički alociranim objektima tipa "`string`". Također uklonite ograničenje da rečenice mogu biti duge najviše 1000 znakova.

4. Izmijenite prethodni program tako što ćete adrese dinamički alociranih objekata tipa "`string`" umjesto u običnim pokazivačima čuvati u dijeljenim pametnim pokazivačima, dok ćete adresu dinamički alociranog niza takvih pokazivača čuvati u odgovarajućem jedinstvenom pametnom pokazivaču (jedinstveni pametni pokazivač je ovdje odabran zbog činjenice da je pogodniji za čuvanje adresa dinamički alociranih nizova, a svakako neće više različitih pokazivača pokazivati na ovaj dinamički alocirani niz). Savjet: iskoristite "`typedef`" deklaraciju da uvedete prikladno ime za tip dijeljenih pametnih pokazivača na stringove, inače ćete imati vrlo glomazne i nečitke konstrukcije. Također, na jednom mjestu će Vam vjerovatno trebat i konverzija pametnih u "glupe" pokazivače.

5. Izmijenite prethodni program tako što ćete adresu dinamički alociranog niza dijeljenih pametnih pokazivača na stringove umjesto u jedinstvenom pametnom pokazivaču čuvati također u dijeljenom pametnom pokazivaču. Vodite računa da ovo, sve dok ne zaživi C++17 stamdard, nosi i izvjesne komplikacije. Prvo, ukoliko želimo da dijeljeni pametni pokazivač pokazuje na dinamički alocirani niz, neophodno je zadati i funkciju koja se poziva kada dođe vrijeme za brisanje objekta (tzv. *custom deleter*). Drugo, takvi pametni pokazivači ne mogu se neposredno indeksirati, nego je neophodna njihova konverzija u "glupe" pokazivače da bismo mogli koristiti indeksaciju.

6. Napišite generičku funkciju "`SortirajListu`" koja kao parametar prima neku listu (tj. objekat tipa "`list`") čiji su elementi proizvoljnog tipa, ali koji se mogu uspoređivati pomoću operatora "`<`". Funkcija treba da kao rezultat vrati novu listu čiji su elementi isti kao u listi navedenoj kao parametar, ali sortirani u rastući poredak. Pri tome, sortiranje treba obaviti "ručno", ne koristeći nikakve bibliotečke funkcije. Nije bitno da postupak sortiranja bude efikasan, bitno je samo da radi (sa znanjem koje trenutno imate, nećete moći napisati ikakav iole efikasniji postupak). Funkciju demonstrirajte u testnom programu koji će za listu cijelih brojeva unesenu sa tastature (broj elemenata se također zadaje putem tastature) kreirati novu sortiranu listu i ispisati njen sadržaj na ekran. Dijalog između programa i korisnika treba izgledati poput sljedečeg:

```
Koliko ima elemenata: 5
Unesite elemente: 3 1 7 6 2
Sortirana lista: 1 2 3 6 7
```

7. Napišite generičke funkcije "`Unija`" i "`Presjek`" koje kao parametre primaju dva skupa (tj. objekta tipa "`set`") proizvoljnog ali istog tipa elemenata, i koje kao rezultat daju novi skup koji predstavlja njihovu uniju, odnosno presjek. Napisane funkcije demonstrirajte u testnom programu koji nalazi uniju i presjek dva fiksna skupa čiji su elementi stringovi.

8. Napišite funkciju "`IzvrniBezRazmaka`" koja kao parametar prima neki string, a kao rezultat vraća taj string izvrnut naopačke, sa izbačenim razmacima (tj. ako je ulaz string "Evo pada kiša", izlazni rezultat treba da bude string "ašikadapovE"). Pri tome, funkcija treba da ima samo tri naredbe, od kojih je prva deklariranje pomoćnog stringa, a treća naredba "**`return`**", tako da sva "logika" funkcije treba biti izvedena u jednoj jedinoj (drugoj) naredbi. Uputa: poslužite se funkcijom "`remove_copy`", zatim obrnutim iteratorima te umetačima (inserterima). Napišite i kratki testni program u kojem ćete testirati napisanu funkciju.

9. Napišite funkcije "`plus2`" i "`plus3`" koje implementiraju respektivno funkcije f i g definirane na sljedeći način. f je funkcija takva da izraz $f(x)(y)$ treba da kao rezultat da vrijednost $x + y$. g je funkcija takva da izraz $g(x)(y)(z)$ treba da kao rezultat da vrijednost $x + y + z$. Ovdje su x, y i z realni brojevi. Napišite i kratki testni program u kojem ćete testirati napisane funkcije.

10. Napišite funkciju "`IteriranaFunkcija`" koja kao parametar prima neku funkciju f koja prima cijeli broj i vraća cijeli broj kao rezultat, te prirodan broj n. Ova funkcija kao rezultat treba vratiti novu funkciju (nazovimo je g) takvu da vrijedi da je $g(x) = f^{(n)}(x) = f(f(f(\ldots f(x)\ldots)))$, gdje se funkcija f primjenjuje na argument n puta uzastopno. Recimo, ukoliko je "`f`" neka funkcija koja prima i vraća cjelobrojnu vrijednost, tada isječak programa

```
auto g(IteriranaFunkcija(f, 10));
std::cout << g(5);
```

odnosno

```
std::cout << IteriranaFunkcija(f, 10)(5);
```

treba da proizvede isti rezultat kao i naredba

```
std::cout << f(f(f(f(f(f(f(f(f(f(5)))))))))));
```

Ukoliko je n negativan ili 0, funkcija vraćena kao rezultat (koju smo nazvali g) treba da bude identička funkcija, tj. za nju treba da vrijedi $g(x) = x$. Napišite i testni program u kojem ćete testirati ovu funkciju.