Zadaća 1.

Ova zadaća nosi ukupno 4 poena (od 20 poena, koliko će ukupno nositi sve zadaće zajedno), pri čemu svaki zadatak nosi po 1 poen. Svi zadaci se mogu uraditi na osnovu gradiva sa prva tri predavanja i pretpostavljenog predznanja iz predmeta "Osnove računarstva". Rok za predaju ove zadaće je srijeda, 26. III 2014. (do kraja dana) i ne može se produžiti. Zadaće se predaju putem Zamgera.

1. Neka je dat neki niz brojeva. Pod *najdužim negativnim podnizom* tog niza brojeva podrazumijeva se najduži podniz *uzastopnih* brojeva tog niza koji su *svi negativni*. Na primjer, ukoliko imamo niz brojeva 3, 5, –2, 4, –6, –3, –1, –2, 7, 6, 3, –5, –1, –4, 6, 0, 9, njegov najduži negativni podniz glasi –6, –3, –1, –2. Napravite funkciju nazvanu "`NajduziNegativniPodniz`" koja kao parametar prima vektor cijelih brojeva, a koja vraća kao rezultat vektor koji se sastoji od elemenata koji predstavljaju najduži negativni podniz niza brojeva koji se nalaze u vektoru koji joj je prenesen kao parametar. Na primjer, ukoliko se izvrše naredbe

```
std::vector<int> v{3, 5, -2, 4, -6, -3, -1, -2, 7, 6, 3, -5, -1, -4, 6, 0, 9};
for(int x : NajduziNegativniPodniz(v)) std::cout << x << " ";
```

   na ekranu treba da se pojavi ispis "`-6 -3 -1 -2 `". U slučaju da postoji više najdužih negativnih podniza (svi su onda iste dužine), funkcija treba da pronađe *prvi od njih* (gledano slijeva). U slučaju da su svi elementi vektora *nenegativni* (tako da negativni podnizovi ne postoje), funkcija treba da vrati *prazan vektor* kao rezultat.

   Napisanu funkciju demonstrirajte u testnom programu koji traži da se sa tastature unese slijed brojeva u neki vektor, a koji nakon toga poziva napisanu funkciju da odredi najduži negativni podniz unesenog slijeda brojeva i ispiše ga na ekran (elementi podniza se trebaju ispisati međusobno razdvojeni zarezom, pri čemu iza posljednjeg elementa ne treba stajati zarez).

2. Neka su date dvije matrice A i B, formata $m \times n$ i $p \times q$ respektivno:

$$\mathrm{A} = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & & \dots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{pmatrix} \qquad \mathrm{B} = \begin{pmatrix} b_{11} & b_{12} & \dots & b_{1q} \\ b_{21} & b_{22} & \dots & b_{2q} \\ \dots & \dots & & \dots \\ b_{p1} & b_{p2} & \dots & b_{pq} \end{pmatrix}$$

   Pod *Kroneckerovim* (ili *tenzorskim*) *proizvodom* ovih matrica smatra se matrica $\mathrm{A} \otimes \mathrm{B}$ formata $mp \times nq$ definirana kao

$$\mathrm{A} \otimes \mathrm{B} = \begin{pmatrix} a_{11}b_{11} & a_{11}b_{12} & \dots & a_{11}b_{1q} & a_{12}b_{11} & a_{12}b_{12} & \dots & a_{12}b_{1q} & \dots & a_{1n}b_{11} & a_{1n}b_{12} & \dots & a_{1n}b_{1q} \\ a_{11}b_{21} & a_{11}b_{22} & \dots & a_{11}b_{2q} & a_{12}b_{21} & a_{12}b_{22} & \dots & a_{12}b_{2q} & \dots & a_{1n}b_{21} & a_{1n}b_{22} & \dots & a_{1n}b_{2q} \\ \dots & \dots & & \dots & \dots & \dots & & \dots & & \dots & \dots & & \dots \\ a_{11}b_{p1} & a_{11}b_{p2} & \dots & a_{11}b_{pq} & a_{12}b_{p1} & a_{12}b_{p2} & \dots & a_{12}b_{pq} & \dots & a_{1n}b_{p1} & a_{1n}b_{p2} & \dots & a_{1n}b_{pq} \\ a_{21}b_{11} & a_{21}b_{12} & \dots & a_{21}b_{1q} & a_{22}b_{11} & a_{22}b_{12} & \dots & a_{22}b_{1q} & \dots & a_{2n}b_{11} & a_{2n}b_{12} & \dots & a_{2n}b_{1q} \\ a_{21}b_{21} & a_{21}b_{22} & \dots & a_{21}b_{2q} & a_{22}b_{21} & a_{22}b_{22} & \dots & a_{22}b_{2q} & \dots & a_{2n}b_{21} & a_{2n}b_{22} & \dots & a_{2n}b_{2q} \\ \dots & \dots & & \dots & \dots & \dots & & \dots & & \dots & \dots & & \dots \\ a_{21}b_{p1} & a_{21}b_{p2} & \dots & a_{21}b_{pq} & a_{22}b_{p1} & a_{22}b_{p2} & \dots & a_{22}b_{pq} & \dots & a_{2n}b_{p1} & a_{2n}b_{p2} & \dots & a_{2n}b_{pq} \\ & & & & & & & & & & & & \\ \dots & \dots & & \dots & \dots & \dots & & \dots & & \dots & \dots & & \dots \\ & & & & & & & & & & & & \\ a_{m1}b_{11} & a_{m1}b_{12} & \dots & a_{m1}b_{1q} & a_{m2}b_{11} & a_{m2}b_{12} & \dots & a_{m2}b_{1q} & \dots & a_{mn}b_{11} & a_{mn}b_{12} & \dots & a_{mn}b_{1q} \\ a_{m1}b_{21} & a_{m1}b_{22} & \dots & a_{m1}b_{2q} & a_{m2}b_{21} & a_{m2}b_{22} & \dots & a_{m2}b_{2q} & \dots & a_{mn}b_{21} & a_{mn}b_{22} & \dots & a_{mn}b_{2q} \\ \dots & \dots & & \dots & \dots & \dots & & \dots & & \dots & \dots & & \dots \\ a_{m1}b_{p1} & a_{m1}b_{p2} & \dots & a_{m1}b_{pq} & a_{m2}b_{p1} & a_{m2}b_{p2} & \dots & a_{m2}b_{pq} & \dots & a_{mn}b_{p1} & a_{mn}b_{p2} & \dots & a_{mn}b_{pq} \end{pmatrix}$$

Na primjer, ukoliko su matrica A i B date kao

$$A = \begin{pmatrix} 3 & 5 & 2 \\ 4 & 0 & -1 \end{pmatrix} \qquad B = \begin{pmatrix} 1 & 4 & -2 & 3 \\ 0 & 5 & 4 & 1 \\ 2 & 0 & 0 & 3 \end{pmatrix}$$

njihov Kroneckerov proizvod glasi

$$A \otimes B = \begin{pmatrix} 3 & 12 & -6 & 9 & 5 & 20 & -10 & 15 & 2 & 8 & -4 & 6 \\ 0 & 15 & 12 & 3 & 0 & 25 & 20 & 5 & 0 & 10 & 8 & 2 \\ 6 & 0 & 0 & 9 & 10 & 0 & 0 & 15 & 4 & 0 & 0 & 6 \\ 4 & 16 & -8 & 12 & 0 & 0 & 0 & 0 & -1 & -4 & 2 & -3 \\ 0 & 20 & 16 & 4 & 0 & 0 & 0 & 0 & 0 & -5 & -4 & -1 \\ 8 & 0 & 0 & 12 & 0 & 0 & 0 & 0 & -2 & 0 & 0 & -3 \end{pmatrix}$$

Napravite funkciju "`KroneckerovProizvod`" koja kao parametre prima dvije matrice A i B organizirane kao vektore vektôrā. Ukoliko parametri nemaju ispravnu formu matrice, odnosno ukoliko svi redovi proslijeđenih vektôrā vektôrā (da, ovaj put dva puta zaredom genitiv množine) nemaju isti broj elemenata, treba baciti izuzetak tipa "`domain_error`" koji sadrži jedan od sljedećih tekstova "Prvi parametar nema formu matrice", "Drugi parametar nema formu matrice" ili "Parametri nemaju formu matrice", ovisno od toga o čemu je zaista riječ.

Napisanu funkciju testirajte u glavnom programu na primjeru matrica čije se dimenzije i elementi unose sa tastature i koji za unesene matrice ispisuje njihov Kroneckerov proizvod. Napomena: u ovakvom konkretnom programu, izuzeci nikada neće biti bačeni, s obzirom da će se funkciji iz glavnog programa uvijek prenositi parametar koji zaista ima strukturu matrice. Međutim, s obzirom da dobro napisana funkcija ne treba ništa da zna o tome u kakvom će okruženju biti korištena, funkcija mora da se zna "odbraniti" u slučaju da u nju ipak "uđu" neispravni podaci.

3. Potrebno je napisati funkciju "`NadjiSimetricneRijeci`" koja u nekoj rečenici nalazi sve simetrične riječi koje se isto čitaju s obe strane (kao što su "kapak", "pop", "bob", "Aziza", "kajak", "monom", "neven", "rotator", itd.). Funkcija kao parametar prima rečenicu koja se analizira (koja je tipa "`string`"), a kao rezultat vraća vektor stringova (tj. vektor čiji su elementi tipa "`string`") koji sadrži spisak simetričnih riječi pronaenih u rečenici (svaka riječ kao zaseban string). Na primjer, ukoliko se kao argument funkciji pošalje string "Pop je podigao kapak na peći", funkcija kao rezultat treba da vrati vektor koji se sastoji od dva stringa "Pop" i "kapak".

   Radi jednostavnosti, pretpostavite da string koji se transformira sadrži samo slova i razmake (i ništa drugo), odnosno da ne sadrži interpunkcijske znake (u suprotnom, otežava se prepoznavanje koje skupine znakova tvore jednu riječ). Ovu pretpostavke ne morate provjeravati, već je samo uzmite kao takvu. Međutim, vodite računa da je moguće da riječi sadrže i velika i mala slova (koja se tretiraju kao jednaka prilikom testa simetričnosti, odnosno ne pravi se razlika između recimo velikog slova "A" i malog slova "a"). Dalje, vodite računa da je moguće da riječi budu razdvojene sa više razmaka, kao i da se na početku odnosno kraju stringa mogu također nalaziti razmaci, kao recimo u rečenici " Aziza vesla kajak " (u kojoj kao simetrične riječi moraju biti prepoznate "Aziza" i "kajak".

4. Položaj zamišljenog robota koji može da se kreće kroz koordinatni sistem sa cjelobrojnim koordinatama opisuje se pomoću tri globalne promjenljive "`x`", "`y`" i "`orjentacija`" (globalne su one promjenljive koje su definirane na početku programa izvan svih funkcija, i koje su vidljive i dostupne u svakoj od funkcija). Promjenljive "`x`" i "`y`" su tipa "**`int`**" i one čuvaju *x* odnosno *y* koordinatu pozicije na kojoj se robot trenutno nalazi. Promjenljiva "`orjentacija`" sadrži informaciju o pravcu u kojem robot trenutno gleda. Ona je tipa "`Pravci`", koji predstavlja pobrojani tip definiran kao

```
enum class Pravci {Sjever, Istok, Jug, Zapad};
```

   Za upravljanje robotom koriste se četiri funkcije, čija su zaglavlja (prototipovi) sljedeći:

```
void Nalijevo();
void Nadesno();
void Idi(int korak);
void IspisiPoziciju();
```

Funkcije "`Nalijevo`" odnosno "`Nadesno`" obrću robota nalijevo odnosno nadesno za 90^o (one zapravo utiču na stanje globalne promjenljive "`orjentacija`". Funkcija "`Idi`" pomjera robota za broj koraka zadan parametrom u smjeru u kojem robot trenutno gleda. Konačno, funkcija "`IspisiPoziciju`" ispisuje na ekran trenutno stanje robota u sljedećem formatu (šiljaste zagrade označavaju da ispis na tom mjestu zavisi od vrijednosti odgovarajuće promjenljive:

Robot se nalazi na poziciji (<*x*>**,** <*y*>**) i gleda na** <*orjentacija*>**.**

Napisane funkcije treba iskoristiti u glavnom programu koji korisniku nudi sljedeće komande za upravljanje robotom koje se zadaju putem tastature: L – Nalijevo; D – Nadesno; I – Idi; K – Kraj. Komande L odnosno D obrću robota nalijevo odnosno nadesno za 90^o. Komanda I obavezno je praćena jednim nenegativnim cijelim brojem (npr. I 20) i ona pomjera robota za navedeni broj koraka u smjeru u kojem je robot trenutno usmjeren. Komanda K završava program. Sve druge komande su ilegalne, i trebaju dovesti do prijave greške i ponovnog izbora komande. Na početku rada, robot se nalazi na poziciji (0,0) i gleda na sjever. Dijalog između programa i korisnika trebao bi izgledati poput sljedećeg:

```
Robot se nalazi na poziciji (0,0) i gleda na sjever.
Unesi komandu (L – Nalijevo, D – Nadesno, I – Idi, K – Kraj): D
Robot se nalazi na poziciji (0,0) i gleda na istok.
Unesi komandu (L – Nalijevo, D – Nadesno, I – Idi, K – Kraj): I
Pogrešna komanda (nedostaje parametar)!
Unesi komandu (L – Nalijevo, D – Nadesno, I – Idi, K – Kraj): I5
Robot se nalazi na poziciji (5,0) i gleda na istok.
Unesi komandu (L – Nalijevo, D – Nadesno, I – Idi, K – Kraj): D
Robot se nalazi na poziciji (5,0) i gleda na jug.
Unesi komandu (L – Nalijevo, D – Nadesno, I – Idi, K – Kraj): I4
Robot se nalazi na poziciji (5,-4) i gleda na jug.
Unesi komandu (L – Nalijevo, D – Nadesno, I – Idi, K – Kraj): D
Robot se nalazi na poziciji (5,-4) i gleda na zapad.
Unesi komandu (L – Nalijevo, D – Nadesno, I – Idi, K – Kraj): S
Pogrešna komanda!
Unesi komandu (L – Nalijevo, D – Nadesno, I – Idi, K – Kraj): IXY2
Pogrešna komanda (neispravan parametar)!
Unesi komandu (L – Nalijevo, D – Nadesno, I – Idi, K – Kraj): I2XY
Pogrešna komanda (neispravan parametar)!
Unesi komandu (L – Nalijevo, D – Nadesno, I – Idi, K – Kraj): I0
Robot se nalazi na poziciji (5,-4) i gleda na zapad.
Unesi komandu (L – Nalijevo, D – Nadesno, I – Idi, K – Kraj): I-1
Pogrešna komanda (neispravan parametar)!
Unesi komandu (L – Nalijevo, D – Nadesno, I – Idi, K – Kraj): I2
Robot se nalazi na poziciji (3,-4) i gleda na zapad.
Unesi komandu (L – Nalijevo, D – Nadesno, I – Idi, K – Kraj): L2
Pogrešna komanda (suvišan parametar)!
Unesi komandu (L – Nalijevo, D – Nadesno, I – Idi, K – Kraj): L
Robot se nalazi na poziciji (3,-4) i gleda na jug.
Unesi komandu (L – Nalijevo, D – Nadesno, I – Idi, K – Kraj): KK
Pogrešna komanda (suvišan parametar)!
Unesi komandu (L – Nalijevo, D – Nadesno, I – Idi, K – Kraj): K
Dovidjenja!
```

Treba predvidjeti i testiranja različitih vrsta greški koje mogu nastati (tipovi grešaka i način kako na njih treba reagirati vidljivi su iz prethodnog dijaloga). Budite sigurni da ste predvidjeli ispravne reakcije na bilo šta što bi eventualno mogao unijeti korisnik (program ne smije da "crkne" ma šta korisnik unio). Napomena: funkcija "`peek`" će Vam vjerovatno biti od velike koristi.