Demonstrature 7 - dodatni zadaci

Napomena 1. Ako u zadatku nije drugačije navedeno, onda nije potrebno provjeravati ispravnost inputa.

Zadatak 1. Implementirajte funkciju *dotProd* koja računa skalarni produkt dvaju vektora koristeći funkciju `zip` u kombinaciji s generatorom liste.

Zadatak 2. Implementirajte funkciju *vecSum* koja zbraja dva realna vektora jednakih dimenzija.

Zadatak 3. Implementirajte funkciju *derive* koja prima polinom s cjelobrojnim koeficijentima te ga derivira. Ako je polinom zadan s

$$p(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n, \quad n \in \mathbb{N}_0,$$

onda ga zapišite kao nepraznu listu $[a_0, a_1, a_2, \ldots, a_n]$.

Zadatak 4. Implementirajte funkciju *primes* koja prima prirodan broj n te algoritmom Eratostenovog sita pronalazi sve proste brojeve manje ili jednake n.

Zadatak 5. Implementirajte funkciju *scale* koja prima realan broj α i vektor v te računa vektor $\alpha \cdot v$ koristeći funkciju `map`.

Zadatak 6. Implementirajte funkciju *is_zero* koja prima realan vektor v te vraća `True` akko je v nul-vektor. Koristite funkciju `filter`.

Zadatak 7. Implementirajte funkciju *removeZeros* koja prima skup realnih vektora te iz njega uklanja sve nul-vektore.

Zadatak 8. Implementirajte funkciju *checkOrth* koja prima dva realna vektora te vraća `True` akko su oni okomiti.

Zadatak 9. Implementirajte funkciju *norm* koja računa normu realnog vektora.

Zadatak 10. Implementirajte funkciju *normalize* koja normalizira realan vektor. Nul-vektor nije moguće normalizirati te u tom slučaju vratite nul-vektor.

Zadatak 11. Implementirajte funkciju *allNdim* koja prima neprazan skup realnih vektora i prirodan broj n te provjerava jesu li svi vektori u danom skupu dimenzije n.

Zadatak 12. Implementirajte funkciju *checkOrthBase* koja prima neprazan skup od n vektora iz $\mathbb{R}^n$ te provjerava tvore li dani vektori ortogonalnu bazu za $\mathbb{R}^n$.
Napomena: skup od n vektora iz $\mathbb{R}^n$ tvori ortogonalnu bazu za $\mathbb{R}^n$ akko se među njima ne nalazi nul-vektor te su svi međusobno okomiti. Osim ova dva uvjeta, treba provjeriti i jesu li svi vektori dimenzije n, gdje je n broj vektora u danom skupu.

Zadatak 13. Implementirajte funkciju *traceFor* koja računa trag realne (kvadratne) matrice koristeći "*for* petlju" te funkciju *trace* koja radi isto što i funkcija *traceFor*, ali pomoću generatora liste ili funkcije `map`.

Zadatak 14. Implementirajte funkciju $forSum$ koja prima cjelobrojnu kvadratnu matricu A reda n s elementima a_{ij} te računa sumu

$$\sum_{i=0}^{n-1}\sum_{j=0}^{n-1} f\left(a_{ij}\right), \qquad f\left(a_{ij}\right) = \begin{cases} i\cdot j\cdot a_{ij}, & i=j \\ 1, & i\neq j,\ 3|a_{ij} \\ 0, & \text{inače} \end{cases}$$

koristeći "*for* petlju" ili funkcije višeg reda. Primijetite da indeksi idu od 0 do $n-1$.

Zadatak 15. Implementirajte funkciju *matSum* koja prima dvije realne matrice jednakih dimenzija i vraća njihov zbroj.

Zadatak 16. Po uzoru na funkciju *trace*, implementirajte funkciju *transpose* koja prima realnu matricu i transponira ju.

Zadatak 17. Implementirajte funkciju *matDotVec* koja prima matricu A i vektor v, a vraća rezultat množenja $A\cdot v$. Ako retke matrice A zamislimo kao vektore $(a_1,\ldots,a_m)$, onda je rezultat množenja $A\cdot v$ vektor s komponentama $(\langle a_1,v\rangle,\ldots,\langle a_m,v\rangle)$.

Zadatak 18. Implementirajte funkciju *matMul* koja prima dvije ulančane matrice, A i B, te računa njihov produkt. Ako stupce matrice B zamislimo kao vektore $[b_1,\ldots,b_r]$, tada je rezultat množenja $A\cdot B$ matrica čiji su stupci vektori $[A\cdot b_1,\ldots,A\cdot b_r]$.

Zadatak 19. Implementirajte funkciju *listToInt* koja prima listu znamenaka te vraća prirodan broj s istim znamenkama. Npr. za listu `[1,8,2,5]` funkcija treba vratiti rezultat 1825. Ukoliko lista sadrži više od 18 elemenata ili je prazna, vratite -1.

Zadatak 20. Koristeći funkciju `foldl` (ili `foldr`) i funkciju `zip`, implementirajte funkciju *binom* koja prima redak iz Pascalovog trokuta reprezentiran listom cijelih brojeva, a vraća sljedeći redak.

Zadatak 21. Koristeći funkcije *binom* i `foldl` (ili `foldr`), implementirajte funkciju *pascal* koja prima $n\in\mathbb{N}_0$ te računa n-ti redak Pascalovog trokuta. Pretpostavite da je nulti redak Pascalovog trokuta jednak `[1]`.

Zadatak 22. Implementirajte funkciju *bubbleSort* koja prima listu podataka iz klase `Ord` te ih sortira od najmanjeg prema najvećem koristeći algoritam *Bubble sort*. Preporučeno je korištenje funkcije `foldr` i pomoćne funkcije *bubble* koja radi jedan prolazak kroz podlistu prvih k članova dane liste. Iskoristite svojstvo funkcije *bubble* koje nam osigurava da se nakon jednog prolaska kroz neku podlistu najveći element te podliste nalazi na njenom desnom kraju.

Zadatak 23. Implementirajte funkciju *opseg* koja prima listu točaka (točke su zadane kao uređeni parovi realnih brojeva) te računa opseg mnogokuta kojem pripadaju dane točke. Pokušajte riješiti ovaj zadatak koristeći funkciju `foldr`.

Zadatak 24. Implementirajte funkciju *stackCommand* koja prima listu cijelih brojeva i cijeli broj c te radi sljedeće:

- Ukoliko je $c = 0$ i lista je prazna, vraća praznu listu;
- Ukoliko je $c = 0$ i lista je neprazna, uklanja prvog člana iz liste;
- Ukoliko je $c \neq 0$, dodaje c na početak liste.

Zadatak 25. Implementirajte funkciju *stack* koja prima listu cijelih brojeva *cs* te iterira kroz nju i primjenjuje funkciju *stackCommand* na trenutno stanje s. Neka je s na početku prazna lista.

Npr. za input `[1,2,3,0,4]` funkcija vraća `[4,2,1]`, dok za input `[1,2,0,0]` vraća praznu listu.

Zadatak 26. Implementirajte funkciju *insert* koja prima podatak n tipa `a` i sortiranu listu tipa `a` te ubacuje n u listu, ali tako da lista ostane sortirana.

Zadatak 27. Implementirajte funkciju *insertionSort* koja sortira listu koristeći algoritam *Insertion sort*.

Zadatak 28. Implementirajte funkciju *maxlen* koja prima dvije liste istog tipa te vraća onu koja je dulja. Ukoliko su liste iste duljine, funkcija vraća drugu.

Zadatak 29. Implementirajte funkciju *lss* koja prima niz elemenata tipa `a` i vraća njegov najveći sortirani podniz. Ukoliko je više takvih, vratite onog koji se prvi pojavio.

Zadatak 30. Deklarirajte tip podatka `HMS` kao uređenu trojku cijelih brojeva u kojem članovi označavaju redom broj sati, minuta i sekunda te implementirajte funkciju *sumHMS* koja prima listu tipa `HMS` i računa "zbroj" njenih članova.

Napomena: ovdje nije nužno deklarirati tip podatka `HMS`, ali ako ga deklariramo, imat ćemo dosta sažetiji kod.