# Demonstrature 5 - dodatni zadaci

**Napomena 1.** Zadatke 1, 2, 3, 7 i 8 pokušajte riješiti koristeći generatore listi.

**Zadatak 1.** Implementirajte funkciju koja prima listu uređenih parova te vraća listu njihovih prvih elemenata. Npr. za listu `[(1,2),(4,7),(9,5)]` funkcija treba vratiti `[1,4,9]`.

**Zadatak 2.** Implementirajte funkciju koja prima string koji se sastoji isključivo od malih slova engleske abecede i razmaka te iz njega izbacuje sve samoglasnike.

**Zadatak 3.** Implementirajte funkciju *findSameChars* koja za dva stringa vraća listu pozicija na kojima se u stringovima nalaze isti znakovi. Npr. za stringove "rijec" i "recenica", funkcija treba vratiti listu `[0,3]`.

**Zadatak 4.** Implementirajte funkciju koja prima rečenicu, a vraća uređenu trojku $(a, b, c)$, pri čemu vrijedi:

- $a$ je lista riječi dane rečenice;
- $b$ je broj riječi u danoj rečenici;
- $c$ je `Bool` koji je nam govori je li broj riječi u danoj rečenici neparan.

**Zadatak 5.** Implementirajte funkciju *is_perm* koja prima dvije liste istog tipa te provjerava je li druga lista permutacija prve.

**Zadatak 6.** Implementirajte funkciju *maloSlovo* koja prima znak $x$ te ukoliko je $x$ veliko slovo engleske abecede, pretvara ga u malo. U suprotnom, funkcija vraća $x$.

**Zadatak 7.** Implementirajte funkciju *malaSlova* koja pretvara sva velika slova u stringu u mala.

**Zadatak 8.** Implementirajte funkciju *atLeast*2 koja prima listu $L$ te argument $e$ istog tipa kao elementi liste $L$. Funkcija vraća `True` ukoliko se u listi $L$ nalaze barem dvije kopije argumenta $e$, a u suprotnom vraća `False`.

**Zadatak 9.** Implementirajte funkciju *remove*2 koja prima listu $L$ te argument $e$ koji je istog tipa kao elementi liste $L$. Ukoliko se u listi nalaze barem dvije kopije elementa $e$, uklonite prve dvije. U suprotnom, ne činite ništa.

**Zadatak 10.** Implementirajte funkciju *toNumber* 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 11.** Implementirajte funkciju koja prima neprazan skup $S \subset \mathbb{Z}$ reprezentiran listom tipa `Int`, a vraća kardinalni broj najmanjeg podskupa skupa $S$ za kojeg vrijedi da je zbroj njegovih elemenata veći od zbroja preostalih elemenata skupa $S$. Pretpostavite da se članovi skupa smiju ponavljati, odnosno pravite se da svuda u zadatku piše *multiskup* umjesto *skup*.

Npr. za skup $\{3, 1, 7, 1\}$ funkcija treba vratiti 1 jer je $\{7\}$ najmanji podskup za koji to vrijedi $(7 > 3 + 1 + 1)$, dok za skup $\{2, 1, 2\}$ funkcija treba vratiti 2 jer su najmanji podskupovi koji zadovoljavaju traženi uvjet skupovi $\{2, 2\}$ i $\{1, 2\}$ (oba imaju kardinalni broj 2).

**Hint:** Sortirajte brojeve u listi od najvećeg prema najmanjem.

**Zadatak 12.** Implementirajte funkciju *intSchedule* koja prima listu uređenih parova $(s_k, f_k)$ duljine $n$, pri čemu vrijedi $(s_k, f_k \in \mathbb{N}_0) \wedge (s_k < f_k)$, $\forall k = 0, \ldots, n-1$, te rješava tzv. *Interval scheduling problem* sljedećim algoritmom:

1. Sortiraj intervale po njihovom vremenu završavanja $f_k$ od najmanjeg prema najvećem.
2. Postavi trenutno vrijeme $t$ na 0.
3. Prođi kroz sve intervale te radi sljedeće:
   - Ukoliko za interval $(s_k, f_k)$ vrijedi $s_k \geq t$, dodaj $k$ u rješenje te postavi trenutno vrijeme $t$ na $f_k$;
   - U suprotnom, preskoči interval.

Npr. za listu `[(0,6),(1,2),(3,4),(5,9),(5,7),(8,9)]`, funkcija treba vratiti `[1,2,4,5]`.

**Napomena:** Prije sortiranja sparite svaki interval s njegovim indeksom koristeći funkciju *zip*. Ne morate provjeravati ispravnost inputa niti sortirati output.