Komparativna analiza programskih jezika

Test 1

Napomena: Samo u drugom zadatku je potrebno navoditi tipove svih parametara funkcija. Ukoliko se tipovi ne navode, moguće je osvojiti najviše 80% bodova na tom zadatku.

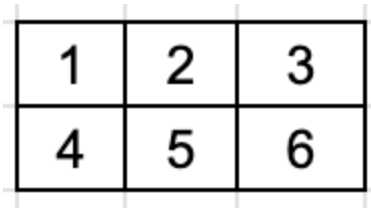
1. (9 bodova) Napisati tail recursive funkcije ***take: int * ‘a list -> ‘a list*** i ***drop: int * ‘a list -> ‘a list*** koje primaju kao parametre nenegativan cijeli broj *n* i listu. Funkcija *take* vraća listu od prvih *n* elemenata proslijeđene liste (ako je *n* veće od dužine liste, vraća čitavu listu). Funkcija *drop* vraća proslijeđenu listu u kojoj je izbačeno prvih *n* elemenata (ako je *n* veće od dužine liste, vraća praznu list). Ukoliko je *n* negativan broj funkcije trebaju baciti izuzetak sa adekvatnom porukom. Demonstrirati i poziv funkcija sa hvatanjem eventualnog izuzetka. Funkcije trebaju očuvati redoslijed elemenata unutar liste.
   Napomena: Ako napisane funkcije nisu tail recursive, moguće je osvojiti maksimalno 50% bodova.

2. (7 bodova) Tip za binarno stablo je definisan na idući način:

   ```
   type 'a binary_tree =
     | Empty
     | Node of 'a * 'a binary_tree * 'a binary_tree
   ```

   Implementirati sljedeće:
   a) funkciju ***binary_tree_map : 'a binary_tree * ('a -> 'b) -> 'b binary_tree*** koja prima binarno stablo i funkciju te vraća binarno stablo u kojem je nad svakim čvorom primijenjena proslijeđena funkcija (slično kao *List.map* funkcija za liste). Ovu funkciju iskoristiti za implementaciju funkcije ***promijeni_znak: int binary_tree -> int binary_tree*** koja promijeni znak svakom čvoru u binarnom stablu (npr. 5 postaje -5). Za implementaciju funkcije koja cijeli broj negira koristiti lambda funkciju;
   b) funkciju ***binary_tree_apply_maps: ('a -> 'a) list * 'a binary_tree -> 'a binary_tree*** koja prima listu funkcija i binarno stablo, a vraća binarno stablo u kojem su nad svakim čvorom primijenjene sve funkcije (u istom poretku kao što su navedene) iz proslijeđene liste funkcija. Demonstrirati korištenje ove funkcije.

3. (9 bodova) Matrica u OCaml jeziku se može čuvati u varijabli tipa ***int list list,*** pri čemu svaki element vanjske liste predstavlja jedan red, a svaki element unutrašnje liste predstavlja broj u tom redu. Npr. matrica sa slike ispod se čuva kao ***[[1;2;3]; [4;5;6]]***. Matrica je validna ukoliko ima bar jedan red, bar jednu kolonu, i svaki red ima jednak broj elemenata (npr. matrice ***[[1;2;3];[4;5]]*** i ***[]*** nisu validne).



Implementirati sljedeće:

a) funkciju ***is_valid: int list list -> bool*** koja provjerava da li je proslijeđena matrica validna;
b) funkciju ***map_two: 'a list * 'a list * ('a * 'a -> 'a) -> int list*** koja prima dvije liste *l1* i *l2*, te funkciju *f*, a vraća listu *l* u kojoj je *l[i] = f(l1[i], l2[i])*;
c) funkciju ***add_row_vectors: int list * int list -> int list*** koja prima dvije liste i vraća listu u kojoj su sabrani odgovarajući elementi te dvije liste (npr. ako su proslijeđene liste ***[1;2;3]*** i ***[4;5;6]*** funkcija treba vratiti listu ***[5;7;8]***);
d) funkciju ***add_matrices: int list list * int list list -> int list list*** koja prima dvije matrice i vraća matricu koja predstavlja zbir proslijeđene dvije matrice.

Napomena: U svim funkcijama u ovom zadatku možete pretpostaviti da su svi parametri odgovarajuće veličine tako da je moguće izvršiti traženu funkcionalnost.