**Opis funkcija**

**Funkcija *primjeri***

Unutar funkcije definirano je zadanih 10 primjera koje korisnik može koristiti. Primjeri su spremljeni unutar niza *baza*. Funkija poziva funkciju *print\_primjeri* i *izbor\_primjera*. Funkcija ne prima parametre.

**Funkcija *izbrisi\_primjer***

Funkcija traži od korisnika da unese broj primjera kojeg želi izbrisati. Uklanja primjer iz niza *baza* prema indeksu stoga uneseni broj umanjujemo za jedan. (Primjeri su numerirani od 1, a niz je numeriran od 0.) Na kraju poziva funkciju *print\_primjeri*. Funkcija prima bazu kao parametar.

**Funkcija *novi\_primjer***

Traži od korisnika da unese relaciju i sprema je u privremenu varijablu *temp\_rel*. Za svaki unos provjerava da li je unesenu realciju sačinjavaju samo slova, ako uneseno nisu samo slova izbacuje prikladnu poruku pogreške i traži ponovan unos. Kreira listu naziva *rel*. Relaciju spemljenu u *temp\_rel* postavlja cijelu u velika slvova i potom sprema kao listu stringova u *rel*. Kreira prazan listu *fo* u koji će se spremati funkcijonalne ovisnosti i brojač koji prati koliki je broj f.o. Od korisnika traži da unese broj funkcijonalnih ovisnosti. U slučaju pogreške ispisuje prikladnu poruku i traži ponovni unos. Korisnika zatim pita da unese f.o. onoliko puta koliki je broj unio radeći provjeru za svaku f.o.(poziva funkciju *provejra\_unosa\_fo*). Sprema ih u *temp\_fo* i u slučaju pogreške ispisuje prikladnu poruku i traži ponovan unos. Na kraju sprema sve f.o. u listu *fo* kao stringove. Kreira novi primjer koji sadrži unesenu relaciju i f.o. te ih pridodaje u *bazu*. Funkcija prima bazu kao parametar.

**Funkcija *provjera\_unosa\_fo***

Funkcija provjerava da li funkcijonalna ovisnost sadrži “strelicu“(- i >), provjerava da li se elementi s lijeve i strane strelice nalaze unutar relacije i provjerava da li elementi s lijeve i desne strane strelice slova. U slučaju da se jedan od uvjeta ne ispuni vraća vrijednost *False* u protivnom vraća vrijednost *True.*Funkcija prima relaciju i funkcijonalne ovisnosti kao parametre.

**Funkcija *print\_primjeri***

Sadrži brojač koji numerira primjer i printa njihove relacije i funkcijonalne ovisnosti. Kao parametar prima bazu.

**Funkcija *izbor\_primjera***

Funkcija traži od korisnika unos prema kojem selektira radnju. If-ovima se provjerava unos. U slučaju da se unio broj, unos se cast-a u int vrijednost koja mora biti jedna od vrijednosti primjera iz *baze*. Printa se koji je odabrani primjer, a u funkciju *key\_search* se šalje upisani primjer (zadnji indeks u *bazi*). Na kraju se ponovno poziva funkcija *izbor\_primjera* za nastavak rada. Ako se unese slovo „x“ izaći će se iz programa, ako se unese „b“ pozvati će se funkcija *brisanje\_primjera*, ako se unese „a“ poziva se funkcija *print\_primjeri*, a za unos slova „p“ poziva se funkcija *novi\_primjer* i potom *key\_search*. Osim za izlazak iz programa uvijek se ponovno poziva *izbor\_primjera*. U slučaju unosa nečeg što nije navedeno funkcija se ponovno poziva, traži se ponovni unos.Funkcija prima bazu kao parametar.

**Funkcija *split***

Kreira prazanu listu. Funkcija dijeli funkcijonalne ovisnosti po strelicama i u listu dodajemo lijevu pa desnu stranu pojedine f.o.(Na neparnim indeksima su desne strane, a na parnim indeksima lijeve strane f.o.). Funkcija prima funkcijonalne ovisnosti primjera.

**Funkcija *containeri***

Funkcija kreira spremnike *temp\_left, left, mid, rifgt* i *temp*. Poziva funkciju *split* nad f.o. primljenog primjera. U prvoj for petlji prolazimo kroz cijelu relaciju, u drugoj for petlji prolazimo kroz listu napravljenu funkcijom *split* od indeksa 0 s korakom 2 i time provjeravamo lijeve strane f.o. Svaki element u relaciji uspoređujemo sa elemtima iz druge for petlje i ako se taj element već ne nalazi u *temp\_left* pridodajemo ga. Isti postupak ponovimo u trećoj for petlji u kojoj provjeravamo elemente s desne strane f.o. i ako se ne nalaze u *right* spremniku dodamo ih. Potom provjerimo sve elemente u *temp\_left* i ako se ikoji nalazi i u *right* spremniku pridodat ćemo ga *mid* spremniku jer se nalazi i s lijeve i s desne strane, a ako se element is *temp\_left* ne nalazi u *mid* spremniku pridodat ćemo ga *left* spremniku. Na kraju ako postoji element koji se ne nalazi ni u *left* ni u *right* spremniku pridodajemo ga u *temp* spremnik. Vraćamo vrijednosti *left,* *mid* i *temp* spremnika. Funkcija prima cijeli primjer odnosno relaciju i funkcijonalne ovisnosti.

**Funkcija *ograda***

**Funkcija *key\_search***

Poziva funkciju *containeri*. Kreira novu listu u koju sprema relaciju bez atributa koji se nalaze u *temp\_cont* spremniku i sortira je. Kreira prazanu listu *keys* za spremanje ključeva. Koristi for petlju koja ima duljinu za 1 više nego što ima atributa u *mid\_cont* spremniku (jedan više kako bi obuhvatili kombinaciju svih atributa zajedno) i poziva ugrađenu funkciju *itertools.combinations* koja kreira kombinacije od prvog primljenog parametra, veličine drugog primljenog parametra. Na kraju iz liste izbacimo kombinacije koje se ponavljaju. Ako je lijevi spremnik prazan u for petlji provjeravamo sve opcije koje smo dobili preko ugrađene funkcije kao potencijalne ključeve i šaljemo ih u funkciju *ograda*. Ako je rezltat jednak relaciji, zadani skup atributa predtavlja jedan od ključeva. Dodajemo mu atribute iz *temp\_cont* spremnika i dodajemo cijeli skup atributa u listu *keys*. U slučaju da lijevi spremnik nije prazan radimo isti postupak, ali prije slanja skupa atributa u funkciju *ograda* pridodamo im sadržaj lijevog spremnika. Na kraju provjerimo ako je duljina liste ključeva veća od nule i stanemo. Printamo relaciju primjera, funkcijonalne ovisnosti, ključeve, te *left\_cont*, *mid\_cont* i *temp\_cont* spremnike kao vizualnu potvrdu ključa. Funkcija prima cijeli primjer kao parametar.