Primene PROLOG-a u različitim oblastima

1. Geografija – interakcija sa korisnikom

```
glavni(srbija, beograd).
glavni(rumunija, bukurest).
glavni(bugarska, sofija).
glavni(madjarska, budimpesta).
glavni(hrvatska, zagreb).

glavnigrad:- nl, nl,
      write('Unesite ime drzave i zavrsite tackom '),
            read(D),
      glavni(D,G), nl,
      write('Glavni grad drzave je:  '),
            write(G).
```

PROGRAM SE poziva sa **glavnigrad.**

Objašnjenje: Da bi predikat **glavnigrad** uspeo, neophodno je:
Da se učita naziv države (biće instanciran promenljivom D),
Da se izvrši unifikacija po činjenicima glavni(drzava, grad)
Da se štampaju komentari *Unesite ime drzave i zavrsite tackom Glavni grad drzave je*
Da se štampa naziv glavnog grada koji se nalazi u promenljivoj G

2. Realizujte jednostavni PROLOG kviz u omiljenoj oblasti i zavrsite tackom.

3. Realizujte jednostavni kviz u omiljenoj oblasti u programskim jezicima C, C++, Python, Java.

4. **Dešifrovanje:**
PROLOG se može uspešno primeniti u rešavanju raznih kriptografskih zadataka.
Tada važnu ulogu ima način rešavanja problema u PROLOG'u, tj. mogućnost nalaženja svih rešenja jednog zadatka.

Zadatak: U sledećim rečima

```
    SNEG
  +SANKE
   -----
   SPORT
```

umesto slova staviti odgovarajuće cifre tako da različita slova označavaju različite cifre, a ista slova iste cifre.

REŠENJE:
1. Rešavamo zadatak tako da se rešenje može iskoristiti i za druge slične kripto zagonetke.
2. Tada je u rešenju potrebno izmeniti samo slova koja se koriste u sabiranju, kao i predikat za štampanje.
3. Ključni deo rešenja je pravilo suma u kom se izračunava zbir dva slova iz iste kolone sa uračunavanjem prenosa iz sabiranja tri slova, ali i izračunavanjem prenosa za naredna dva slova.
Na početku, prenos je 0.
4. Važno je i pravilo koje proverava da li su svi elementi jedne liste međusobno različiti. Na taj način obezbeđujemo da različite promenljive sadrže različite cifre.

```
/*zamena slova ciframa */
cifra(0).
cifra(1).
```

```
cifra(2).
cifra(3).
cifra(4).
cifra(5).
cifra(6).
cifra(7).
cifra(8).
cifra(9).

/*  ovaj deo programa treba izmeniti u slucaju da se resava zadatak istog tipa */
/*  u ovom delu vrsi se sabiranje po kolonama */

/* suma(P,X,Y,Z,P1):  P je prenos, cifre su X, Y, W je X+Y+P,
                P1 je novi prenos W//10, Z je rezultat koji pisemo W mod 10. */

/* sabiramo unazad E+G=pisemo T prenosimo P4,
                P4+E+K=pisemo R prenosimo P3
                P3+N+N=pisemo O prenosimo P2
                P2+S+A=pisemo P prenosimo P1
                P1+0+S=pisemo S prenosimo 0
 */
resenje([S,N,E,G,A,K,P,O,R,T]) :-
    suma(P1,0,S,S,0),
    suma(P2,S,A,P,P1),
          suma(P3,N,N,O,P2),
    suma(P4,E,K,R,P3),
    suma(0,G,E,T,P4),  S \= 0,
    razliciti([S,N,E,G,A,K,P,O,R,T]).

/* Pravilu    suma(0,G,E,T,P4) dodajemo uslov S \= 0
   da bi se izbeglo stampanje brojeva koji imaju vodece nule */


/* sabiranje trojke slova */
  suma(P,X,Y,Z,P1) :- prenos(P), cifra(X), cifra(Y), W is X+Y+P,
                P1 is W//10, Z is W mod 10.

/* vrste prenosa */
prenos(1).
prenos(0).

/* provera da li su elementi liste razliciti */
razliciti([]).
razliciti([X|Y]) :- van(X,Y), razliciti(Y).

/* provera da li element pripada listi */
van(X,[]).
van(X,[Y|Z]) :- X \= Y, van(X,Z).

/* i ovaj deo treba promeniti ako se resava zadatak sa drukcijim sabircima
  zbog odgovarajuceg ispisa resenja */
  izlaz:- resenje(L), pravi_izlaz(L).
  pravi_izlaz([S,N,E,G,A,K,P,O,R,T]) :-
    write('           '), s1([S,N,E,G]),
          write('         +'), nl,
          write('          '), s1([S,A,N,K,E]),
```

```
   write('          ----------'),nl,
        write('          '), s1([S,P,O,R,T]).

/* stampanje liste */
s1([]):-nl.
s1([H|T]) :- write(H), tab(1), s1(T).
```

POZIV PROGRAMA: izlaz.

5. Pronađite tekst nekog od takmičarskih zadataka iz matematike koji predstavlja sličnu mozgalicu u kojoj se traži dešifrovanje. Rešite sličnu mozgalicu uz pomoć PROLOG programa.

6. Pronađite tekst nekog od takmičarskih zadataka iz matematike koji predstavlja sličnu mozgalicu u kojoj se traži dešifrovanje. Rešite sličnu mozgalicu uz pomoć programskog jezika C, C++, Python, Java.

7. Biohemija

```
/* RH - pozitivnost */

/* odredjivanje verovatnoce da dete bude RH-pozitivno*/
/* kada su poznati Rh-antigeni roditelja */

rh :- start(R1,prvog), start(R2,drugog),
    (
          (sigurno_plus(R1,R2),
           write('Dete je 100% Rh-pozitivno'));
                (sigurno_minus(R1,R2),
                write('Verovatnoca da dete bude Rh-pozitivno je 0'));
                (dva_tri(R1,R2),
                write('Verovatnoca da dete bude Rh-pozitivno je 2/3'));
                (cetiri_sedam(R1,R2),
                write('Verovatnoca da dete bude Rh-pozitivno je 4/7'));
                (jedan_dva(R1,R2),
                write('Verovatnoca da dete bude Rh-pozitivno je 1/2'))
          ).

start(M,S) :- nl,
     write('Unesite kombinaciju Rh-antigena '),
     write(S), write(' roditelja.'), nl,
     write('Kombinaciju cine 3 slova izmedju navodnika, a'),
     nl, write('slova mogu biti iz skupa {C,D,E,c,d,e}'),
     nl, write('u abecednom poretku i bez ponavljanja.'),
     nl, write('Unos zavrsiti tackom.'),nl,
     read(M),
     provera(M),nl.

provera([A,B,C]) :- A=99, B=100, C=69,
     write('NEKOREKTNO!'),nl,
     write('Kombinacija "cdE" nije moguca')
          !, fail.

provera([A,B,C]) :- ((A=67;A=99),
                (B=68;B=100),
                (C=69;C=101));
                (write('NEKOREKTNO!'),nl,
```

```
                write('Dozvoljena su samo slova iz skupa {C,D,E,c,d,e}'), fail).

sigurno_plus([A1,B1,C1],[A2,B2,C2]) :- B1=68, B2=68.
sigurno_minus([A1,B1,C1],[A2,B2,C2]) :- B1=100, B2=100.

dva_tri([A1,B1,C1],[A2,B2,C2]) :- A1=99, A2=99, C1 \= C2.
dva_tri([A1,B1,C1],[A2,B2,C2]) :- A1 \= A2, C1=69, C2=69.

cetiri_sedam([A1,B1,C1],[A2,B2,C2]) :- A1 \= A2, C1 \= C2.

jedan_dva([A1,B1,C1],[A2,B2,C2]) :- A1=67, A2=67.
jedan_dva([A1,B1,C1],[A2,B2,C2]) :- C1=101, C2=101.
```