Logičko i funkcijsko programiranje

PROLOG

Kolokvijum, 2024/25

Na **Desktop**-u u direktorijumu **Rad** kreirati direktorijum **ImePrezime_BrIndeksa** i unutar njega sačuvati rešenja datih zadataka.

1. Za reprezentaciju celih brojeva biće korišćena lista koja sadrži odgovarajući broj simbola x. Na primer, broju nula lista [], broju dva odgovara lista [x, x], broju pet lista [x, x, x, x, x] i tako dalje. Naredne predikate definisati direktno, bez prevođenja na numeričke vrednosti.
   a. **[1 poen]** Napisati predikat **`successor/2`** koji za dati broj određuje njegovog naslednika

      ```
      ?- successor([x,x,x], L).
      L = [x, x, x, x].
      ```

      ```
      successor(L,[x|L]).
      ```

   b. **[2 poena]** Napisati predikat **`addition/3`** koji za dva data broja vreća njihov zbir

      ```
      ?- addition([x,x],[x,x,x,x], Z).
      Z = [x, x, x, x, x, x].

      ?- addition([x,x],Z,[x,x,x,x,x]).
      Z = [x, x, x].
      ```

      ```
      addition([],X,X).
      addition([x|X],Y,Z):-addition(X,Y,Z1),successor(Z1,Z).
      ```

   c. **[2 poena]** Napisati predikat **`multiplication/3`** koji za dva data broja određuje njihov proizvod.

      ```
      ?- multiplication([x,x],[x,x,x],M).
      M = [x, x, x, x, x, x] .

      ?- multiplication(M,[x,x],[x,x,x,x]).
      M = [x, x] .
      ```

      ```
      multiplication([],_,[]).
      ```

```
multiplication(_,[],[]).
multiplication([x|X],Y,Z):-multiplication(X,Y,Z1),
                           addition(Z1,Y,Z).
```

2. Konstruisan je robot koji može da izvršava tri komande: **right** – okreni se desno za 90 stepeni, **left** – okreni se levo za 90 stepeni, **move** – pomeri se napred za 1 metar. Početno stanje robota je na poziciji (0,0) i orijentacija ka severu (**north**).
   a. **[3 poena]** Definisati predikat **`execute/5`** koji za datu poziciju, orijentaciju i komandu određuje novu poziciju i orijentaciju.

```
execute((2,3),north,right, Position, Orintation).
Position = (2, 3),
Orintation = east .

?- execute((2,3), north, move, Position, Orintation).
Position = (2, 4),
Orintation = north.

?- execute((2,3), south, move, Position, Orintation).
Position = (2, 2),
Orintation = south.
```

```
execute((X,Y), north, right, (X,Y), east).
execute((X,Y), east, right, (X,Y), south).
execute((X,Y), south, right, (X,Y), west).
execute((X,Y), west, right, (X,Y), north).

execute((X,Y), north, left, (X,Y), west).
execute((X,Y), east, left, (X,Y), north).
execute((X,Y), south, left, (X,Y), east).
execute((X,Y), west, left, (X,Y), south).

execute((X,Y), north, move, (X,Y1), north) :- Y1 is Y+1.
execute((X,Y), south, move, (X,Y1), south) :- Y1 is Y-1.
execute((X,Y), east, move, (X1,Y), east) :- X1 is X+1.
execute((X,Y), west, move, (X1,Y), west) :- X1 is X-1.
```

   b. **[3 poena]** Definisati predikat **`status/3`** koji za listu komandi određuje na kojoj pozicije će robot biti ako polazeći od početnog stanja izvrši dati spiska komandi.

```
?- status([move,move,right,move,move,move],Pos,Orient).
```

```
Pos = (3, 2),
Orient = east .

?- status([],Position,Orientation).
Position = (0, 0),
Orientation = north.

?- status([left,left,move],Position,Orientation).
Position = (0, -1),
Orientation = south .
```

```
status([], Position, Orientation,Position,Orientation).
status([X|L],Position,Orientation,Position1,Orientation1):-
                    execute(Position,Orientation,X,P,O),
                    status(L,P,O,Position1,Orientation1).
status(L,Position,Orientation):-
                    status(L,(0,0),north,Position,Orientation).
```

3. Definisati predikat **`roman2arabic/2`** koji string koji predstavlja broj zapisan rimskim ciframa prevodi u arapski broj.
   Za rešavanje problema definisati sledeće predikate:
   a. **[1 poen]** Definisati predikat **`symbol/2`** koji određuje numeričku vrednost za rimsku cifru. Obuhvatiti cifre I, V, X, L, C, D, M.

```
?- symbol('X', Value).
Value = 10.
```

```
symbol('I', 1).
symbol('V', 5).
symbol('X', 10).
symbol('L', 50).
symbol('C', 100).
symbol('D', 500).
symbol('M', 1000).
```

   b. **[2 poena]** Definisati predikat **`symbol2numbers/2`** koji za listu rimskih cifara određuje listu arapskih vrednosti.

```
?- symbol2numbers(['M','M','X','V'], Value).
Value = [1000, 1000, 10, 5].

?- symbol2numbers(['X','L','I','I'], Value).
Value = [10, 50, 1, 1].
```

```
symbol2numbers([],[]).
symbol2numbers([R|L],[V|L1]):-symbol(R,V), symbol2numbers(L,L1).
```

c. **[1 poen]** Unaprediti predikat **`symbol2numbers`** da vraća korektnu listu vrednosti s obzirom na redosled rimskih cifara

```
?- symbol2numbers2(['X','L','I','I'], Value).
Value = [40, 1, 1].
```

```
symbol2numbers2([],[]).
symbol2numbers2([R1,R2|L],[V|L1]):-symbol(R1,V1),symbol(R2,V2),
          V1 < V2, V is V2 - V1, symbol2numbers2(L,L1),!.
symbol2numbers2([R|L],[V|L1]):-symbol(R,V),symbol2numbers2(L,L1).
```

d. **[3 poena]** Rešiti početni problem.

```
?- roman2arabic('XLII', Arabic).
Arabic = 42.
```

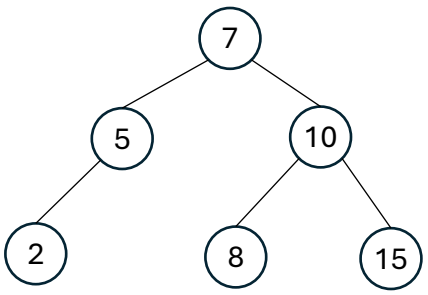
Dozvoljeno je korišćenje ugrađenog predikata **`string_chars/2`** koji od stringa formira listu karaktera

```
?- string_chars('ABC',L).
L = ['A', 'B', 'C'].
```

```
sum([],0).
sum([X|L], R):- sum(L,R1), R is R1 + X.

roman2arabic(Roman, Arabic):-string_chars(Roman, List),
          symbol2numbers2(List,List1), sum(List1, Arabic).
```

4. **[5 poena]** Binarno stablo brojeva je predstavljeno predikatom **`t/3`**, gde prvi argument predstavlja sadržaj čvora, drugi predikat levo podstablo, a treći preditak desno podstablo, pri čemu konstanta **nil** označava da podstablo ne postoji. Stablu na slici odgovara predikat



t(7, t(5, t(2, nil, nil), nil), t(10, t(8, nil, nil), t(15, nil, nil)))

Ako je list čvor sa dva nil podstabla, napisati predikat **`count_leaves/2`** koji broji listove u datom stablu.

```
?- count_leaves(t(7, t(5, t(2, nil, nil), nil), t(10, t(8, nil,
nil), t(15, nil, nil))), N).
N = 3.

?- count_leaves(t(7, t(5, t(2, nil, nil), nil), t(10, t(8, nil,
nil), t(15, nil, nil))), N), write(N), fail.
3
false.
```

```
count_leaves(nil,0).
count_leaves(t(_,nil,nil),1) :- !.
count_leaves(t(_,L,R),N) :- count_leaves(L,NL),
                    count_leaves(R,NR), N is NL+NR.
```