**FMI, MI, Anul III**
**Programare logică**

# Test laborator
# - Numărul I -

| P1 | P2 | P3 | Total |
|---|---|---|---|
| | | | |

*Pentru rezolvarea probemelor puteţi defini predicate ajutătoare, puteţi folosi predicate predefinite, dar* **nu** *puteţi folosi meta-predicate care colecteaza toate soluţiile (de tipul* findall*,* findnsol*,* bagof*,* setof*, etc.).*

*Nu trataţi cazuri de eroare în afara celor cerute explicit de problemă.*

**(P1)** [5 puncte]
Scrieţi un predicat `find_index(List, V,LI)` astfel încât:

- `List` este o lista cu perechi `iv(Index, Valoare)`
- pentru `V` dat să determine lista `LI` care conţine toate elementele `I` cu proprietatea că `iv(I,V)` este în `List`.

**Exemplu:**
```
?- find_index([iv(1,a), iv(2,b), iv(3,a)], a, LI)
LI=[1,3]
```

**(P2)** [10 puncte] Scrieţi un predicat `all_lists(N,LL)` care, având ca prim argument un număr natural `N`$\geq 0$ instanţiază `LL` cu lista tuturor listelor `[k,...,1]` cu $0 \leq$`k`$\leq$`N`, astfel încât elementele `[k,...,1]` sunt numere impare consecutive. În `LL` listele pot apărea în orice ordine, dar fiecare listă apare o singură dată.
**Exemplu:**
```
?- all_lists(6,LL).
LL = [[5, 3, 1], [3, 1], [1]].
```

**(P3)** [15 puncte] În această problemă vom lucra cu grafuri **neorientate**. Un graf va fi reprezentat prin lista vârfurilor `V` şi lista muchiilor `LE`. O muchie este reprezentată printr-o pereche `edge(a,b)` (muchia poate fi parcursă în ambele sensuri). De exemplu, lista vârfurilor este `[a,b,c,d,e,f]`şi lista muchiilor este `[edge(a,b), edge(b,c), edge(b,d), edge(d,e), edge(e,a)]`.

(a) [5 puncte] Scrieţi un predicat `valid(V,LE)` primeste ca argumente o listă de vârfuri şi o listă de muchii, şi întoarce `true` dacă capetele fiecărei muchii apar în lista vârfurilor.

**Exemplu:**

```
?- valid([a,b,c,d,e,f],[edge(a,b), edge(b,c), edge(b,d), edge(d,e), edge(e,a)]).
true .

?- valid([a,c,d,e,f],[edge(a,b), edge(b,c), edge(b,d), edge(d,e), edge(e,a)]).
false.
```

(b) [10 puncte] Scrieţi un predicat `connected(V,LE)` care întoarce `true` dacă graful reprezentat prin `V` şi `LE` este conex (există cel puţin un drum între oricare două vârfuri).

**Exemplu:**

```
?- connected([a,b,c,d,e],[edge(a,b), edge(b,c), edge(b,d), edge(d,e), edge(e,a)]).
true

?- connected([a,b,c,d,e,f],[edge(a,b), edge(b,c), edge(b,d), edge(d,e), edge(e,a)])
false.
```