7. laboratorijas darbs

1. Uzrakstīt predikātu `concat_and_order(list, list, list)`. Iegūt no diviem sākotnējiem *kārtotiem* sarakstiem trešo, rezultējošu, sarakstu, kur visi elementi *kārtoti*. Kārtošana notiek augošajā secībā.

```
concat_and_order([1, 3], [2, 4], L) %L=[1,2,3,4]
concat_and_order([3, 4], [1, 2], L) %L=[1,2,3,4]
concat_and_order([], [1, 2], L)     %L=[1,2]
concat_and_order([1, 2], [], L)     %L=[1,2]
```

2. Pirmajā laboratorijas darba tika izmantoti fakti ar informāciju par vecākiem formā `parent(<kas>, <kam>)`. Uzrakstīt predikātu `unique_parents1(L)`, kurš atgriež sarakstu no *unikāliem* vecāku vārdiem. Lai ir fakti:

```
parent(aldis, vizma).
parent(aldis, uldis).
parent(inga, vizma).
parent(inga, uldis).
```

Predikāta rezultāts:

```
unique_parents1(L) %L=["inga","aldis"]
```

Piezīme: būs nepieciešami arī daži *papildus predikāti*. Kādā predikātā tiks pielietots predikāts `findall(...)`. Vārdu secība sarakstā var būt *patvaļīga*.

3. Uzrakstīt predikātu `unique_parents2(L)`, kura rezultāts sakrīt ar 2. uzdevuma rezultātu. Var atšķirties tikai vecāku vārdu kartība. Piemēram:

```
unique_parents2(L) %L=["aldis","inga"]
```

Piezīme: **nevienā** papildus predikātā **ne**izmanto predikātu `findall(...)`. Ieteicams iegūt līdzīgo efektu, *dinamiski* veidojot sarakstu `database` daļā. Piemēram, var deklarēt faktu `l(list)`.