Soal Pra Praktikum Logika Komputasional – IF2121

PETUNJUK PRA PRAKTIKUM:

1. Pra Praktikum IF2121 - Logika Komputasional adalah kegiatan yang bersifat **mandiri**. Semua bentuk kecurangan akan ditindaklanjuti sesuai dengan sanksi akademik yang ada.
2. Praktikum dikerjakan dengan menggunakan **GNU Prolog**. **Tidak diperbolehkan** menggunakan **SWI Prolog**.
3. Format file jawaban yang dikumpulkan adalah **PP01_[NIM].pl** untuk file prolog, dan **PP01_[NIM].txt** untuk query beserta hasilnya.
4. Sebelum menjawab tiap butir soal, **tandai** terlebih dahulu untuk **setiap poin**.

`/*Untuk File .pl*/`
 	
`/* Bagian <X> */`
`/* Deklarasi Fakta */`
`<fakta>`
 	
`/* Deklarasi Rules */`
`predikat1(X):- predikat2(X)`
	`/*Untuk File .txt*/`
	
	`Bagian <X>`
	
	`Query: <query>`
	`<hasil query>`

5. Jika terdapat pertanyaan, harap ditanyakan melalui **sheet QnA.** Pertanyaan yang diajukan secara personal ke asisten tidak akan dijawab untuk menghindari perbedaan informasi yang didapatkan oleh peserta praktikum.
6. Semua deliverable files jawaban praktikum **dikompres** ke dalam arsip dengan ekstensi **.zip**, lalu di-***upload*** ke Edunex pada bagian assignment. Format penamaan file arsip praktikum adalah **PP01_[NIM].zip**.
7. Disarankan untuk mengerjakan semua bagian dengan struktur yang rapi dan jelas dan disertai komentar, karena selain memudahkan penilaian oleh asisten, juga akan sangat membantu dalam proses *debugging*.
8. ***Deadline*** pengumpulan adalah **7 November 2022** pukul **23.59** waktu server.

Revisi: 6 November 2022, 12:13 WIB

BAGIAN I: Fakta, *Rule*, dan *Query*

Untuk menjawab soal-soal pada bagian ini, akses tautan berikut untuk melihat diagram keluarga yang digunakan adalah **berikut ini**

1. Buatlah fakta-fakta dari pohon keluarga di atas dengan menggunakan HANYA aturan fakta di bawah ini. Tulislah dalam bahasa pemrograman prolog lalu simpan dalam file .pl sesuai aturan.

    a. pria(X) : X adalah pria
    b. wanita(X) : X adalah wanita
    c. usia(X,Y) : X berusia Y
    d. menikah(X,Y) : X menikah dengan Y
    e. anak(X,Y) : X adalah anak Y
    f. saudara(X,Y) : X adalah saudara kandung Y

2. Buatlah rule/aturan di bawah ini **TANPA** membuat rule/fakta tambahan. Tulislah dalam bahasa pemrograman prolog lalu simpan dalam file **.pl** sesuai aturan! **(Boleh menggunakan rule yang sudah didefinisikan butir soal lain)**:

    a. kakak(X,Y) : X adalah kakak dari Y (baik perempuan maupun lelaki)
    b. keponakan(X,Y) : X adalah keponakan dari Y
    c. suami(X,Y) : X adalah suami dari Y
    d. sepupu(X,Y) : X adalah sepupu dari Y
    e. menantu(X,Y) : X adalah menantu dari Y
    f. orangtua(X,Y): X adalah orang tua dari Y
    g. bibi(X,Y) : X adalah bibi dari Y
    h. cucu(X,Y) : X adalah cucu dari Y
    i. keturunan(X,Y): X adalah keturunan dari Y (berlaku untuk anak, cucu, dan seterusnya)
    j. anaksulung(X) : X adalah anak paling tua
    k. anakbungsu(X) : X adalah anak paling muda

3. Implementasi kalimat di bawah ini ke bentuk query prolog, kemudian tulis query dan hasilnya dalam file **.txt** sesuai aturan! **(dilarang membuat rule tambahan selain dari soal)** Kerjakan soal menggunakan fakta, logika, dan fungsi dari 1.1 atau 1.2, dan tidak dengan langsung mencetak nama dari pohon keluarga. Dalam artian rule yang dibuat dalam variabel dahulu (misal X).

    - Untuk soal yang spesifik memberikan nama, cukup ganti X menjadi namanya dalam jawaban.

- Misalnya nama pada soal diganti, contoh 1.3.a menjadi Istri dari Ave, semua nama Vito di jawaban diganti menjadi Ave.
- Untuk soal g sampai l, jawaban harus mengembalikan semua nama orang yang memenuhi kriteria.

Jadi jangan jawab langsung dengan langsung print nama jawaban ya :)

a. Istri dari Vito
b. Adik dari Elon Musk
c. Cicit dari Gojo
d. Paman atau bibi tertua dari Wesly (pilih yang tertua)
e. Pasangan dari ipar Aqua (X adalah ipar Y jika X adalah saudara dari suami/istri Y)
f. Cucu termuda dari Khelli
g. Orang yang menjadi anak kedua dari keluarganya*
h. Orang tua yang memiliki tepat dua anak
i. Orang yang punya mertua
j. Wanita yang punya lebih dari 1 anak
k. Wanita yang belum pernah melahirkan ketika berusia di atas 20 tahun
l. Keturunan dari Gede yang umurnya tidak lebih kecil dari 25 tahun

***) jika orang tuanya tidak terdefinisi, diasumsikan bukan anak kedua**

BAGIAN II: Rekursivitas

WARNING: Pada bagian ini Anda diharapkan untuk menggunakan pendekatan **rekursif** dalam implementasi solusi permasalahan. Jika **tidak** menggunakan rekursivitas maka akan terdapat **pengurangan nilai** meskipun jawaban yang diberikan benar. Pada bagian ini Anda dapat membuat fungsi lain untuk membantu pengerjaan.

1. combination(N,C,X)

N dalam hal ini adalah *total objects*, sedangkan C adalah *number of objects selected*. combination(N, C, X) mengeluarkan hasil kombinasi antara N dan C, yaitu X. Berikut contoh *query* dan hasil eksekusi *query*.

```
| ?- combination(5, 2, X).
X = 10.0

yes
```

2. change(X, Y)

Bayangkan kamu adalah seorang banker. Customer sering datang untuk menukarkan uang mereka. Pecahan uang yang kamu punya adalah 1, 2, dan 5. Y adalah nilai uang yang customer ingin tukar, sedangkan X adalah jumlah koin atau jumlah pecahan uang minimum yang bisa kamu berikan. Berikut adalah contoh hasil eksekusinya.

```
| ?- change(X, 14).
X = 4 ?

yes
```

Penjelasan: 14 = 5 + 5 + 2 + 2 (4 pecahan)

3. fpb(A, B, X)

X adalah faktor persekutuan terbesar dari A dan B. Berikut ini adalah contoh *query* dan hasil *query* untuk *rule* berikut.

```
| ?- fpb(14,18,X).
X = 2 ?
yes
```

4. mod(A,B,X)

X adalah sisa dari pembagian A terhadap B, dengan A >= 0 dan B > 0. Berikut ini adalah contoh *query* dan hasil *query* untuk *rule* berikut (**Solusi wajib menggunakan rekursif**).

```
| ?- mod(5,2,X).
X = 1 ?
yes
```

5. deret(X, Y)

Diberikan sebuah deret berikut: 1, 2, 4, 5, 10, …, dst.

X adalah suku ke-N dari deret berikut dan Y adalah nilai suku ke-N tersebut.

```
| ?- deret(5, Y).
Y = 10 ?
yes
```

Istilah deret boleh diganti barisan. Mohon maaf atas kekeliruan istilahnya.

BAGIAN III: *List*

WARNING: Untuk bagian ini Anda hanya diperbolehkan untuk menggunakan operator **pipe (|)**. Apabila Anda menggunakan *rule list processing* yang disediakan oleh gprolog, maka **nilai Anda untuk nomor tersebut adalah 0**. Anda **diperbolehkan untuk membuat *rule* tambahan** untuk mempermudah pengerjaan. Anda juga diperbolehkan untuk tidak mengikuti format query yang diberikan pada contoh query dan hasilnya selama di hasil pekerjaan anda **dijelaskan di komentar terkait struktur dan metode pemanggilan querynya! Asumsikan bahwa paling tidak terdapat 1 buah elemen di dalam list untuk semua pertanyaan di bawah ini!**

A. Statistik List

List merupakan sebuah tipe data yang digunakan untuk menyimpan sekumpulan data dengan tipe yang sama. Sering kali kita ingin menghitung statistik dari data yang kita miliki. Oleh karena itu, implementasikan lah operasi-operasi statistik dasar di bawah ini untuk membantu perhitungan statistik dari data yang dimiliki.

- min: mencari elemen dengan nilai minimum
- max: mencari elemen dengan nilai maksimum
- range: mencari selisih antara elemen terbesar dan elemen terkecil
- count: mencari jumlah elemen pada list
- sum: mencari jumlah total elemen pada list

Berikut ini adalah contoh *query* dan hasil *query* untuk *rule-rule* di atas

min(List, Min)

`| ?- min([9, -8, -7, 11, -10], Min).`
 	
`Min = -10?`
 	
`yes`

max(List, Max)

`| ?- max([9, -8, -7, 11, -10], Max).`
 	
`Max = 11?`
 	
`yes`

range(List, Range)
`| ?- range([9, -8, -7, 11, -10], Range).`

```
Range = 21?

yes
```

count(List, Count)

```
| ?- count([9, -8, -7, 11, -10], Count).

Count = 5?

yes
```

sum(List, Sum)

```
| ?- sum([9, -8, -7, 11, -10], Sum).

Sum = -5?

yes
```

B. List and Queue Manipulation

1. Get Index

Mengembalikan indeks sebuah elemen pada list. Jika ada **dua elemen yang bernilai sama** pada list, maka **yang dikembalikan adalah yang pertama.** Jika **elemen tidak ditemukan** pada list, **kembalikan 'no'. Index dimulai dari 1.**

getIndex(List, SearchedElement, Result)

```
| ?- getIndex([1,2,5,-2,1,-4,-7], -2, Index).

Index = 4

yes
| ?- getIndex([1,2,5,-2,1,-2,-7], -2, Index).

Index = 4

yes

| ?- getIndex([1,2,5,-2,0,-2,-7], -11, Index).

no
```

2. **Swap**

Menukar dua buah elemen pada indeks tertentu dan mengembalikan List yang sudah diperbarui nilainya. **Nilai indeks masukan diasumsikan selalu positif**. **Index dimulai dari 1.**

swap(List, Index1, Index2, Result)

```
| ?- swap([5,6,7,8], 1, 1, Result).

Result = [5,6,7,8]

yes

| ?- swap([5,6,7,8], 4, 2, Result).

Result = [5,8,7,6]

yes

| ?- swap([5,6,7,8,9], 4, 6, Result).

no
```

3. **Split List**

Memisahkan List yang di-*input* menjadi dua buah List berdasarkan nilai elemennya dengan aturan ganjil-genap. **Asumsikan terdapat paling tidak ada satu elemen pada List**.

splitList(ListAwal, ListGanjil, ListGenap)

```
| ?- splitList([1,2,3,4,5,6], Ganjil, Genap).

Ganjil = [1,3,5]
Genap = [2,4,6] ?

yes

| ?- splitList([1,2,3,4,5,6,7], Ganjil, Genap).

Ganjil = [1,3,5,7]
Genap = [2,4,6] ?

yes
```

BONUS

Buatlah program kalkulator sederhana dengan mendefinisikan perintah* dan *query* berikut:

- startCalculator: menandakan kalkulator dimulai dan menyimpan nilai 0.
- add(X) : menambahkan X ke dalam nilai yang disimpan oleh kalkulator dan menyimpannya
- subtract(X): mengurangi X dari nilai yang disimpan oleh kalkulator dan menyimpannya
- multiply(X): mengalikan X dengan nilai yang disimpan oleh kalkulator dan menyimpannya
- divide(X): membagi nilai yang disimpan oleh kalkulator dengan X dan menyimpannya
- clearCalculator: mereset nilai yang disimpan di kalkulator menjadi 0
- getValue: mengembalikan nilai yang saat ini disimpan di kalkulator
- exitCalculator(X): keluar dari aplikasi kalkulator

Catatan:

- Jika permainan belum berlangsung atau sudah selesai, kemudian perintah selain start dimasukkan, kembalikan false (atau no).
- Program ini memerlukan fungsi bawaan print dan teknik *dynamic predicate*. Selamat melakukan eksplorasi. :D
- Diperbolehkan menggunakan fungsi I/O untuk merapikan dan memperindah *output*.
- Perhatikan bahwa setiap aksi pada kalkulator, kecuali *startCalculator* dan *getValue*, mengeluarkan nilai lama yang disimpan kalkulator dan nilai yang baru.

*) Perintah adalah fungsi yang tidak memiliki parameter apapun.

Berikut adalah contoh jalannya permainan:

```
| ?- startCalculator.
Selamat Datang di Kalkulator Prolog!
Currently Saved Value is 0

yes
| ?- add(5).
Old Saved Value is 0
New Saved Value is 5

yes
| ?- subtract(7).
Old Saved Value is 5
New Saved Value is -2

yes
| ?- divide(-2).
```

```
Old Saved Value is -2
New Saved Value is 1

yes
| ?- multiply(0.5).
Old Saved Value is 1
New Saved Value is 0.5

yes
| ?- getValue.
Currently Saved Value is 0.5

yes
| ?- clearCalculator.
Old Saved Value is 0.5
New Saved Value is 0

yes
| ?- exitCalculator.
Terima Kasih telah Menggunakan Kalkulator Prolog

yes
| ?- add(6).

no
| ?- subtract(8).

no
```