

Soal Pra Praktikum Logika Komputasional – IF2121

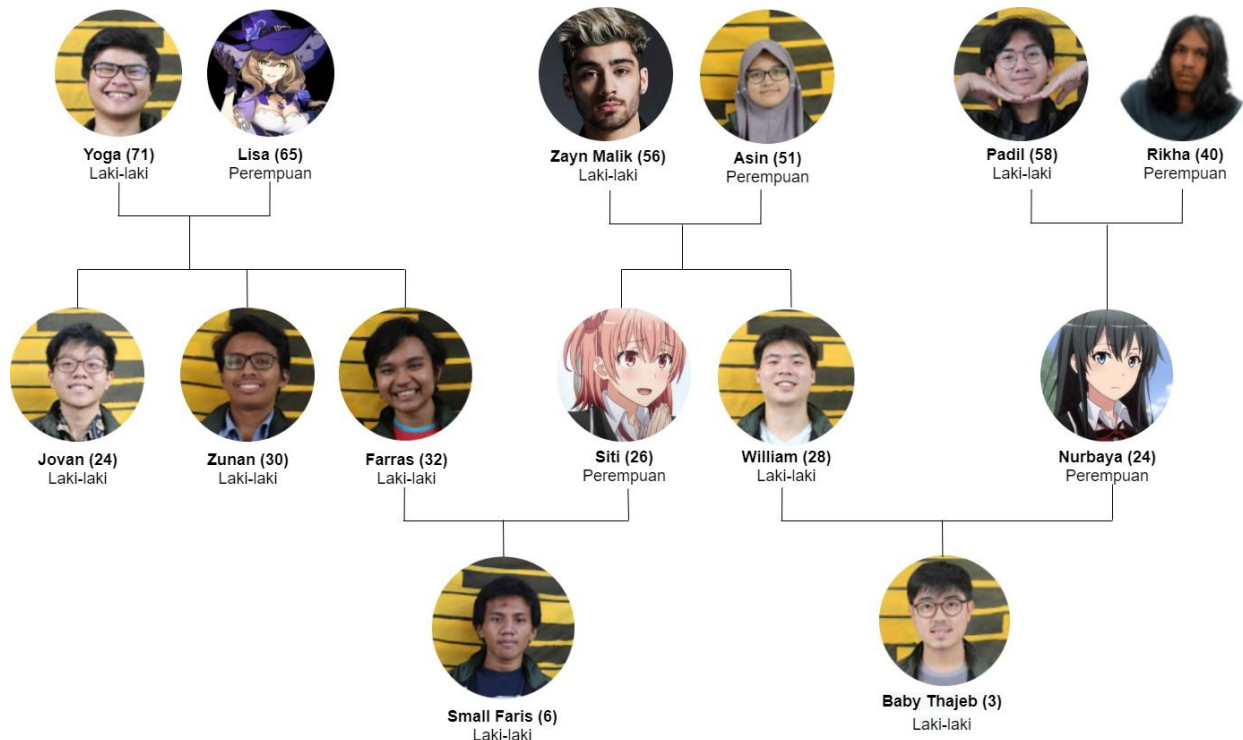
PETUNJUK PRA PRAKTIKUM:

1. Praktikum IF2121 - Logika Komputasional adalah praktikum yang bersifat **mandiri**.
2. Praktikum dikerjakan dengan menggunakan **GNU 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**.

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

5. Semua deliverable files jawaban praktikum **dikompres** ke dalam arsip dengan ekstensi **.zip**, lalu di-**upload** ke link [berikut](#). Format penamaan file arsip praktikum adalah **PP01_[NIM].zip**.
 6. **Deadline** pengumpulan adalah **8 November 2020** pukul **23.59** waktu server.
-

BAGIAN I : Fakta, Rule, dan Query



- 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.
 - pria(X) : X adalah pria
 - wanita(X) : X adalah wanita
 - usia(X,Y) : X berusia Y
 - menikah(X,Y) : X menikah dengan Y
 - anak(X,Y) : X adalah anak Y
 - saudara(X,Y) : X adalah saudara kandung Y
- 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**):
 - kakak(X,Y) : X adalah kakak dari Y (baik perempuan maupun lelaki)
 - keponakan(X,Y) : X adalah keponakan dari Y
 - suami(X,Y) : X adalah suami dari Y
 - sepupu(X,Y) : X adalah sepupu dari Y
 - mertua(X,Y) : X adalah mertua dari Y
 - bibi(X,Y) : X adalah bibi dari Y
 - cucu(X,Y) : X adalah cucu dari Y
 - anaksulung(X) : X adalah anak paling tua

- i. 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 2**)
- a. Suami dari Nurbaya
 - b. Paman dari Small Faris
 - c. Menantu dari Yoga
 - d. Nenek dari Small Faris
 - e. Cucu dari Padil
 - f. Ipar dari Siti
 - g. Sepupu dari Baby Thajeb
 - h. Wanita yang merupakan anak tunggal
 - i. Pria yang belum menikah

BAGIAN II : Rekurens

Perhatian: Sangat dianjurkan untuk mengerjakan dengan cara rekursif! Jika tidak menggunakan rekursif akan terdapat pengurangan nilai! Diperbolehkan juga membuat fungsi selain fungsi yang dikerjakan untuk membantu pengerjaan

1. **faktorial(N,X)** : X merupakan hasil perkalian bilangan bulat dari 1 hingga N dengan ketentuan $N \geq 1$.

Berikut ini adalah contoh *query* dan hasil *query* untuk *rule* berikut

```
| ?- faktorial(3,X) .  
  
X = 6 ?  
  
yes
```

2. **gcd(A,B,X)** : X adalah *greatest common divisor* dari A dan B.

Berikut ini adalah contoh *query* dan hasil *query* untuk *rule* berikut

```
| ?- gcd(8,20,X) .  
  
X = 4 ?  
  
yes
```

3. **power(A,B,X)** : X adalah hasil A dipangkatkan dengan B

Berikut ini adalah contoh *query* dan hasil *query* untuk *rule* berikut

```
| ?- power(2,5,X) .  
  
X = 32 ?  
  
yes
```

4. **countDigit(A,X)** : X merupakan banyak digit dari A

Berikut ini adalah contoh *query* dan hasil *query* untuk *rule* berikut

```
| ?- countDigit(13519000,X) .  
  
X = 8 ?  
  
yes
```

5. **createTriangle(X)** : Menuliskan sebuah pola segitiga siku-siku seperti pada contoh.

Berikut ini adalah contoh *query* dan hasil *query* untuk *rule* tersebut

```
| ?- createTriangle(5) .  
*****  
****  
***  
**  
*  
  
true ?
```

BAGIAN III : List

Perhatian: Diperbolehkan membuat fungsi selain fungsi yang dikerjakan untuk membantu pengerjaan

1. Queue merupakan struktur data dalam ilmu komputer yang memiliki prinsip **FIFO (First In First Out)**, bacaan lebih lanjut <https://www.geeksforgeeks.org/fifo-vs-lifo-approach-in-programming/>)
Operasi-operasi dasar queue adalah
 - Push: menambahkan elemen ke belakang Queue sesuai aturan FIFO,
 - Pop: menghapus elemen terdepan dari Queue sesuai aturan FIFO,

- Front: mengembalikan elemen terdepan dari Queue, dan
- Back: mengembalikan element terbelakang dari Queue.

Tugas kalian adalah mengimplementasikan fungsi dari push, pop, front, dan back.

push(Element, Queue, Result) memasukkan **element** ke dalam **queue**, kemudian menghasilkan **result**.

```
| ?- push(4, [1, 2, 3], Result).  
  
Result = [1, 2, 3, 4] ?  
  
yes
```

pop(Queue, Result) menghapus elemen terdepan dari **queue**, kemudian menghasilkan **result**

```
| ?- pop([1, 2, 3], Result).  
  
Result = [2, 3] ?  
  
yes
```

front(Queue, Result) mengembalikan elemen terdepan dari **queue** sebagai **result**

```
| ?- front([1, 2, 3], Result).  
  
Result = 1 ?  
  
yes
```

back(Queue, Result) mengembalikan elemen terbelakang dari **queue** sebagai **result**

```
| ?- back([1, 2, 3], Result).  
  
Result = 3 ?  
  
yes
```

2. **concatenate(Flist, Slist, X, Y, Result)**: terdapat 2 buah *list* dan 2 buah angka, **Flist** akan diambil elemen sepanjang X, **Slist** akan diambil elemen sepanjang Y, kemudian keduanya akan digabung menjadi **Result**.

Berikut adalah contoh query dan hasil querynya

```
| ?- concatenate([1, 2, 3], [4, 5, 6], 2, 2, Result).  
Result = [1, 2, 4, 5] ?  
yes
```

3. **palindrom(List)** : *Rule* digunakan untuk mengecek apakah sebuah **list** yaitu **list** palindrom atau tidak.

Berikut ini adalah contoh *query* dan hasil *query* untuk *rule* berikut

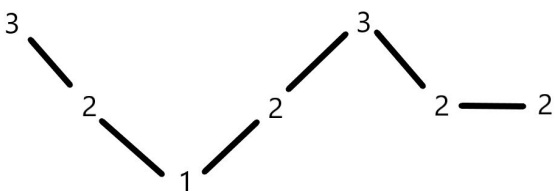
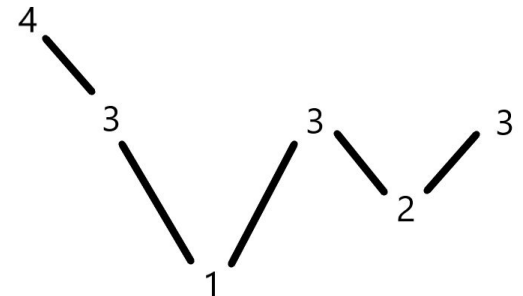
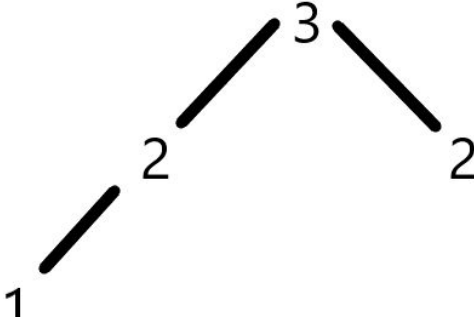
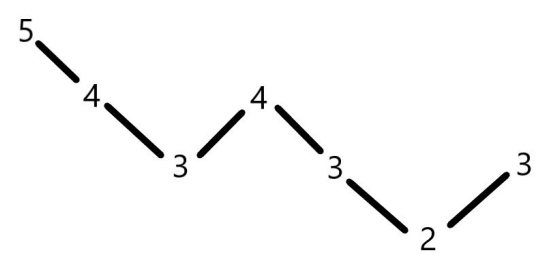
```
| ?- palindrom([1,2,3,2,1]).  
yes  
  
| ?- palindrom([]).  
yes  
  
| ?- palindrom([1,2,2,3,2,1]).  
no
```

BONUS

1. **ridge(List)**: ridge adalah list yang berisi lembah. Jika **list** tidak terdapat lembah, akan mengembalikan *no*, jika terdapat lembah, akan mengembalikan banyaknya lembah yang terbentuk. Lembah diawali dengan barisan menurun sampai dasar, kemudian setelah sampai dasar, akan dilanjutkan barisan menaik. Titik dasar sebelum menaik disebut **titik balik lembah**.

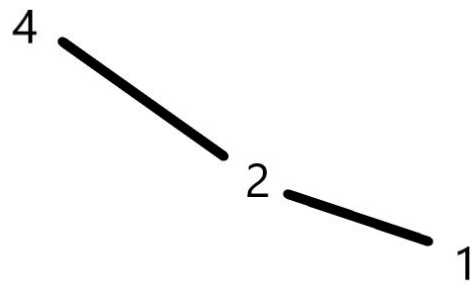
Disarankan menggunakan **dynamic predicate** yang disediakan pada prolog.

Karakter yang dibold di bawah adalah **titik balik lembah**.

<pre> ?- ridge([3, 2, 1, 2, 3, 2, 2]).</pre> <p>1</p>	
<pre> ?- ridge([4, 3, 1, 3, 2, 3])</pre> <p>2</p>	
<pre> ?- ridge([1, 2, 3, 2]).</pre> <p>no</p>	 <p>Tidak ada lembah</p>
<pre> ?- ridge([5, 4, 3, 4, 3, 2, 3])</pre> <p>2</p>	

```
| ?- ridge([4, 2, 1])
```

no



Tidak ada lembah