ACARA PRAKTIKUM 4

SISTEM PAKAR 1

TUJUAN

- Memahami dan mengenal sistem pakar, logika predikat, bahasa deklaratif, dan bahasa pemrograman Prolog.
- Memahami langkah-langkah membuat sebuah sistem pakar sederhana menggunakan SWI-Prolog.
- Memahami langkah-langkah membuat aplikasi sistem pakar berbasis GUI sederhana menggunakan PySwip dan Tkinter.

DASAR TEORI

1. Sistem Pakar

Sistem pakar adalah sebuah sistem yang mengadopsi pengetahuan manusia ke dalam komputer, sehingga komputer dapat menangani masalah tertentu dengan meniru metode kerja ahli. Sistem pakar memungkinkan transfer keahlian dari seorang ahli (atau sumber lainnya) ke komputer. Informasi tersebut disimpan dalam komputer, dan pengguna dapat berkonsultasi untuk mendapatkan saran. Komputer kemudian dapat membuat kesimpulan, layaknya seorang ahli, serta memberikan penjelasan dan alasan ketika diperlukan.

2. Logika Predikat

Logika predikat (kalkulus predikat) merupakan bagian dari komputasi logika yang juga mencakup aljabar Boole (logika proporsional), di mana fakta dan aturan dinyatakan melalui predikat seperti berikut:

lelaki(Joko) // fakta

menikah(Joko, Tuti) // fakta

$\forall x \; \forall y$ [**menikah**(x,y) $\wedge$ **lelaki**(x)] $\rightarrow$ **~lelaki**(y) // aturan; $\forall$ berarti "untuk setiap"

$\forall y \; \exists x$ [**orang**(y) $\rightarrow$ **ibu**(x,y)] // aturan; $\exists$ berarti "ada"

Kalimat pertama menunjukkan adanya fakta bahwa Joko adalah seorang lelaki dan kalimat kedua menyatakan bahwa Joko menikah dengan Tuti. Kalimat ketiga dan keempat menunjukkan suatu aturan atau kaidah yang berlaku. Kalimat ketiga menyatakan aturan bahwa untuk setiap pasang orang x dan y, jika x menikah dengan y dan x adalah lelaki, dapat disimpulkan bahwa y adalah bukan seorang lelaki. Kalimat keempat menyatakan aturan bahwa untuk setiap y, jika y adalah orang, y mempunyai seorang ibu x (x adalah ibu dari y).

Simbol predikat yang digunakan dalam kalimat tersebut adalah **lelaki**, **menikah**, **orang**, dan **ibu** yang dapat disebut sebagai simbol relasi, serta Joko dan Tuti yang dapat disebut sebagai simbol konstanta.

3. Bahasa Deklaratif dan Prolog

Bahasa pemrograman biasanya bersifat prosedural atau imperatif, di mana program disusun melalui langkah-langkah dan prosedur yang terstruktur. Namun, ada juga bahasa pemrograman yang bersifat deskriptif atau deklaratif, seperti Prolog.

Prolog adalah singkatan dari Programming in Logic yang berarti pemrograman logika. Prolog menggunakan bahasa deklaratif, di mana pemrogram memberi fakta dan aturan untuk selanjutnya diselesaikan oleh Prolog secara deduktif sehingga menghasilkan suatu kesimpulan. Selain itu, berbeda dengan pemrograman fungsional, pemrograman logika menggunakan relasi, bukan fungsi, sehingga sangat sesuai untuk mengimplementasikan sebuah sistem pakar.

Sebagai contoh, dalam Prolog, bahasa deklaratif digunakan untuk menyatakan fakta dan aturan sebagai berikut:

- Jika ingin menyatakan bahwa Prawiro adalah bapak dari Joko, dalam Prolog dituliskan sebagai berikut:
    - **bapak**(prawiro, joko).
- Jika ingin menerangkan suatu kaidah bahwa A adalah kakek dari Z, dalam Prolog dituliskan sebagai berikut:
    - Aturan dalam Bahasa Indonesia:
        - A adalah kakek dari Z jika A adalah bapak dari X dan X adalah bapak dari Z.
        - A adalah kakek dari Z jika A adalah bapak dari X dan X adalah ibu dari Z.
    - Aturan dalam Bahasa Prolog:
        - **kakek(A,Z) :- bapak(A,X), bapak(X,Z).**
        - **kakek(A,Z) :- bapak(A,X), ibu(X,Z).**

4. Dasar Pemrograman Prolog

Pada bagian ini akan diuraikan dasar-dasar pemrograman Prolog, aturan umum penulisan program, bagaimana melakukan dialog dengan Prolog, dan beberapa pengertian dasar yang berkaitan dengan program Prolog.

a. Klausa

Setiap program Prolog tersusun dari klausa-klausa. Klausa dapat berupa *fakta* atau *aturan*. Klausa terdiri dari *head* (kepala) dan *body* (tubuh) yang dipisahkan dengan tanda **:-** dan diakhiri dengan tanda titik. Namun, klausa fakta hanya terdiri dari *head* saja. Tubuh klausa dapat terdiri dari beberapa sub-klausa yang dihubungkan satu sama lain menggunakan tanda koma (**,**) yang menunjukkan hubungan **and** atau tanda titik koma (;) yang menunjukkan hubungan **or**. Penggabungan tubuh klausa yang dirangkai dengan *and* disebut sebagai **konjungsi**, sedangkan yang dirangkai dengan *or* disebut sebagai **disjungsi**.

b. Fakta

Fakta adalah suatu kenyataan atau kebenaran yang diketahui atau hubungan (relasi) antara dua atau lebih objek. Fakta dapat pula menunjukkan sifat suatu objek. Contoh sederhana adalah:

- **bapak**(prawiro, joko).
- **merah**(darah).
- **asin**(garam).

c. Aturan

Aturan merupakan logika yang dirumuskan dalam bentuk relasi sebab-akibat dan hubungan implikasi. Misalnya, dapat dibuat aturan bahwa jika A adalah bapak dari X dan X adalah bapak atau ibu dari Z, dapat dipastikan bahwa A adalah kakek dari Z. Contohnya adalah aturan **kakek** seperti pada halaman sebelumnya.

d. Predikat

Kumpulan klausa dengan struktur *head* yang sama disebut sebagai predikat. Suatu predikat memiliki nama dan nol atau lebih argumen. Nama predikat berupa atom, sedangkan argumen predikat dapat berupa atom atau variabel.

Pada aturan **kakek**, terdapat dua klausa yang menyatakan aturan bagaimana syarat A adalah kakek dari Z. Kakek(A,Z) tersebut adalah predikat yang menerima dua parameter, atau dapat ditulis sebagai kakek/2.

e. Atom dan Variabel

Argumen suatu predikat dapat berupa konstanta (atom) atau variabel. Atom disebut juga sebagai objek nyata, sedangkan variabel disebut sebagai objek umum. Suatu atom atau variabel dalam Prolog disebut sebagai **term**, sehingga argumen suatu predikat selalu berupa term.

Dalam Prolog, setiap term yang ditulis dengan awalan huruf kapital adalah *variabel bernama*, sedangkan awalan dengan huruf kecil adalah *atom* atau *konstanta*.

Selain itu, *variabel tak bernama* digunakan untuk mengabaikan nilai suatu variabel, yang berarti bisa bernilai apa saja.

f. Query

Query atau Pertanyaan digunakan untuk memperoleh jawaban dari suatu masalah secara deduktif berdasarkan fakta-fakta dan aturan-aturan yang telah ditentukan. Dalam Prolog, query dinyatakan dengan goal. Goal terdiri dari dua jenis, yaitu goal internal yang ditulis langsung di dalam program dan goal eksternal yang ditulis di luar program atau di console. Berikut adalah beberapa contoh goal:

- Goal Internal
  - **orangtua**(P,Q) :- **bapak**(P,Q); **ibu**(P,Q).
    - Pada klausa tersebut, terdapat dua goal internal, yaitu **bapak**(P,Q) dan **ibu**(P,Q), di mana goal pertama menanyakan program apakah P adalah bapak dari Q dan goal kedua menanyakan program apakah P adalah ibu dari Q.
- Goal Eksternal
  - **write**("Hello World"), **nl**.
    - Pada klausa tersebut, terdapat dua goal eksternal, yaitu **write** dan **nl**. Memasukkan query tersebut di console akan menampilkan tulisan "Hello World" dan membuat satu baris baru.
  - **bapak**(prawiro, joko).
    - Menanyakan program apakah Prawiro adalah bapak dari Joko.
  - **bapak**(X, joko).
    - Menanyakan program siapa bapak dari Joko dan menampilkan semua jawabannya.
  - **bapak**(prawiro, X).
    - Menanyakan program siapa anak dari Prawiro dan menampilkan semua jawabannya.

ALAT DAN BAHAN

1. PC/Laptop
2. SWI-Prolog
3. Code Editor
4. Python

PERCOBAAN

Percobaan 1 (Silsilah Keluarga)

Langkah-langkah:

1) Buka Code Editor.
2) Buat sebuah file baru bernama *silsilah.pl*
3) Pertama, definisikan beberapa fakta seperti berikut.

```
% SILSILAH KELUARGA

% FAKTA
% -- Hubungan orang tua.
% -- Predikat parent/2 (menerima dua parameter X dan Y)
% -- berarti X adalah orang tua dari Y.
parent(alya, bima).
parent(alya, satria).
parent(bima, david).
parent(bima, emma).
parent(satria, yunita).
parent(satria, grace).
```

4) Kemudian, definisikan aturan-aturan yang berlaku seperti berikut.

```
% ATURAN
% -- Hubungan saudara kandung.
% -- Predikat sibling/2 (menerima dua parameter X dan Y)
% -- berarti X adalah saudara kandung Y apabila fakta-fakta sesuai dengan
aturan.
sibling(X, Y) :-
      parent(Z, X),
      parent(Z, Y),
      X \= Y.      % X tidak sama dengan Y (Garis miring adalah negasi)

% -- Hubungan kakek-nenek.
% -- Predikat grandparent/2 (menerima dua parameter X dan Y)
% -- berarti X adalah kakek/nenek dari Y apabila fakta-fakta sesuai
dengan aturan.
grandparent(X, Y) :-
      parent(X, Z),
      parent(Z, Y).

% -- Hubungan turun-temurun.
% -- Predikat ancestor/2 (menerima dua parameter X dan Y)
% -- berarti X adalah pendahulu Y apabila fakta-fakta sesuai dengan
aturan.
ancestor(X, Y) :-
      parent(X, Y).

ancestor(X, Y) :-
      parent(X, Z),
      ancestor(Z, Y).
```

5) Buka SWI-Prolog. Kemudian, pilih File > Consult. Pilih file *silsilah.pl* yang sudah dibuat.
6) Masukkan beberapa query berikut.
    - **parent**(bima, Anak). // Siapa anaknya Bima?
    - **grandparent**(alya, Cucu). // Siapa cucunya Alya?
    - **sibling**(bima, satria). // Apakah Bima dan Satria saudara kandung?
    - **ancestor**(Leluhur, emma). // Siapa leluhurnya Emma?

Percobaan 2 (Gejala Penyakit Malaria)

Diketahui gejala dan pola penyakit sebagai berikut.

Kode Gejala	Keterangan
G1	Demam
G2	Menggigil
G3	Rasa tidak enak badan
G4	Nyeri otot
G5	Keringat dingin
G6	Sakit kepala
G7	Mimisan
G8	Mual
G9	Muntah
G10	Kejang-kejang

No	Penyakit	Gejala
1	Malaria Tertiana	G4, G9, G10
2	Malaria Quartana	G2, G3, G4
3	Malaria Tropika	G5, G6, G7, G8
4	Malaria Pernisiosa	G2, G3, G1, G7, G8

Langkah-langkah:

1) Buka Code Editor.
2) Buat file baru bernama *malaria.pl*
3) Ketik code berikut.

```
% GEJALA PENYAKIT MALARIA

% DATABASE
% -- Menyimpan data apakah X memiliki gejala Y.
:- dynamic gejala/2.

% ATURAN
% -- Mengandung daftar gejala untuk beberapa jenis penyakit malaria,
% -- dengan predikat penyakit/2 (menerima dua parameter)
% -- yang berarti X mengidap penyakit Y.
% -- Predikat gejala/2 berarti X memiliki gejala Y.
penyakit(X, tertiana) :-
      gejala(X, nyeri_otot),
      gejala(X, muntah),
      gejala(X, kejang).

penyakit(X, quartana) :-
      gejala(X, nyeri_otot),
      gejala(X, menggigil),
      gejala(X, tidak_enak_badan).

penyakit(X, tropika) :-
      gejala(X, keringat_dingin),
      gejala(X, sakit_kepala),
      gejala(X, mimisan),
      gejala(X, mual).

penyakit(X, pernisiosa) :-
      gejala(X, menggigil),
      gejala(X, tidak_enak_badan),
      gejala(X, demam),
      gejala(X, mimisan),
      gejala(X, mual).
```

4) Buka SWI-Prolog. Pilih menu File > Consult dan pilih file *malaria.pl* yang sudah dibuat.
5) Masukkan query berikut secara berurutan.
    - **penyakit**(steph, Penyakit). // Apa penyakit yang dimiliki Steph?
    - **assertz**(gejala(steph, nyeri_otot)). // Menyimpan fakta X memiliki gejala Y
    - **assertz**(gejala(steph, menggigil)).
    - **assertz**(gejala(steph, tidak_enak_badan)).
    - **penyakit(**steph, Penyakit).

- **retract**(gejala(steph, nyeri_otot)). // Menghapus fakta X memiliki gejala Y
- **assertz**(gejala(steph, demam)).
- **assertz**(gejala(steph, mimisan)).
- **assertz**(gejala(steph, mual)).
- **penyakit**(steph, Penyakit).

Query seperti **penyakit**(steph, Penyakit) dapat menghasilkan lebih dari satu jawaban. Pada SWI-Prolog, tekan SPASI untuk lanjut ke jawaban selanjutnya hingga semua jawaban ditampilkan atau tekan ENTER untuk berhenti.

Percobaan 3 (Sistem Pakar Malaria)

Langkah-langkah:

1) Buka Code Editor.
2) Buat file baru bernama *pakar_malaria.pl*
3) Ketik code berikut.

```
% GEJALA PENYAKIT MALARIA

% DATABASE
% -- Menyimpan data apakah gejala X positif atau negatif.
:- dynamic gejala_pos/1.
:- dynamic gejala_neg/1.

% ATURAN
% -- Predikat pertanyaan/1 untuk menanyakan pertanyaan terkait gejala X.
pertanyaan(nyeri_otot) :-
	write("Apakah Anda merasa nyeri otot?").

pertanyaan(muntah) :-
	write("Apakah Anda muntah-muntah?").

pertanyaan(kejang) :-
	write("Apakah Anda mengalami kejang-kejang?").

pertanyaan(menggigil) :-
	write("Apakah Anda sering menggigil?").

pertanyaan(tidak_enak_badan) :-
	write("Apakah Anda merasa tidak enak badan?").

pertanyaan(keringat_dingin) :-
	write("Apakah Anda mengalami keringat dingin?").

pertanyaan(sakit_kepala) :-
	write("Apakah Anda sering sakit kepala?").

pertanyaan(mimisan) :-
```

```
      write("Apakah Anda sering mimisan?").

pertanyaan(mual) :-
      write("Apakah Anda merasa mual?").

pertanyaan(demam) :-
      write("Apakah Anda demam?").

% -- Predikat diagnosa/1 digunakan untuk menanyakan dan menyimpan status
gejala X.
diagnosa(G) :-
      pertanyaan(G),
      writeln(" (y/t)"),
      read(Jawaban),
      Jawaban == y,
      assertz(gejala_pos(G)).

diagnosa(G) :-
      assertz(gejala_neg(G)),
      fail.

% -- Predikat gejala/1 dipanggil untuk memerika status gejala dari
database,
% -- atau menanyakan user terkait gejala tersebut.
gejala(G) :-
      gejala_pos(G), !.

gejala(G) :-
      gejala_neg(G), !,
      fail.

gejala(G) :-
      diagnosa(G).

% -- Mengandung daftar gejala untuk beberapa jenis penyakit malaria,
% -- dengan predikat penyakit/1 (menerima satu parameter)
% -- yang berarti mengidap penyakit X.
penyakit(tertiana) :-
      gejala(nyeri_otot),
      gejala(muntah),
      gejala(kejang),
      terdeteksi("Malaria Tertiana").

penyakit(quartana) :-
      gejala(nyeri_otot),
      gejala(menggigil),
      gejala(tidak_enak_badan),
      terdeteksi("Malaria Quartana").

penyakit(tropika) :-
      gejala(keringat_dingin),
      gejala(sakit_kepala),
      gejala(mimisan),
      gejala(mual),
      terdeteksi("Malaria Tropika").

penyakit(pernisiosa) :-
      gejala(menggigil),
```

```
      gejala(tidak_enak_badan),
      gejala(demam),
      gejala(mimisan),
      gejala(mual),
      terdeteksi("Malaria Pernisiosa").

penyakit(_) :-
      writeln("Tidak terdeteksi penyakit.").

% -- Predikat terdeteksi/1 dipanggil untuk mencetak penyakit.
terdeteksi(P) :-
      write("Anda terdeteksi penyakit "),
      writeln(P).

% -- Predikat clear_db/0 untuk membersihkan database gejala.
clear_db :-
      retractall(gejala_pos(_)),
      retractall(gejala_neg(_)).

% -- Predikat main/0 sebagai main loop sistem pakar.
main :-
      write('\33\[2J'),   % Clear window
      writeln("DIAGNOSA PENYAKIT THT"),
      penyakit(_),
      clear_db,
      writeln("INGIN MENGULANG?"),
      read(Jawaban), !,
      Jawaban == y,
      main.
```

4) Buka SWI-Prolog. Pilih menu File > Consult dan pilih file *pakar_malaria.pl* yang sudah dibuat.
5) Masukkan query **main.** untuk memulai program sistem pakar dan lakukan uji coba dengan mensimulasikan gejala-gejala salah satu penyakit.

Percobaan 4 (Sistem Pakar Malaria berbasis GUI)

Langkah-langkah:

1) Install library PySwip menggunakan perintah berikut.
2) Buka Code Editor.
3) Buat file baru bernama *pakar_malaria_gui.pl*
4) Ketik code berikut.

```
% GEJALA PENYAKIT MALARIA

% DATABASE
:- dynamic gejala_pos/1.
:- dynamic gejala_neg/1.
```

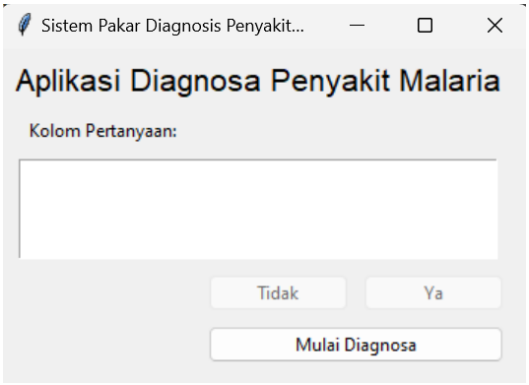
```
% FAKTA & ATURAN
penyakit("Malaria Tertiana").
penyakit("Malaria Quartana").
penyakit("Malaria Tropika").
penyakit("Malaria Pernisiosa").

gejala(nyeri_otot, "Malaria Tertiana").
gejala(muntah, "Malaria Tertiana").
gejala(kejang, "Malaria Tertiana").
gejala(nyeri_otot, "Malaria Quartana").
gejala(menggigil, "Malaria Quartana").
gejala(tidak_enak_badan, "Malaria Quartana").
gejala(keringat_dingin, "Malaria Tropika").
gejala(sakit_kepala, "Malaria Tropika").
gejala(mimisan, "Malaria Tropika").
gejala(mual, "Malaria Tropika").
gejala(menggigil, "Malaria Pernisiosa").
gejala(tidak_enak_badan, "Malaria Pernisiosa").
gejala(demam, "Malaria Pernisiosa").
gejala(mimisan, "Malaria Pernisiosa").
gejala(mual, "Malaria Pernisiosa").

pertanyaan(nyeri_otot, Y) :-
      Y = "Apakah Anda merasa nyeri otot?".

pertanyaan(muntah, Y) :-
      Y = "Apakah Anda muntah-muntah?".

pertanyaan(kejang, Y) :-
      Y = "Apakah Anda mengalami kejang-kejang?".

pertanyaan(menggigil, Y) :-
      Y = "Apakah Anda sering menggigil?".

pertanyaan(tidak_enak_badan, Y) :-
      Y = "Apakah Anda merasa tidak enak badan?".

pertanyaan(keringat_dingin, Y) :-
      Y = "Apakah Anda mengalami keringat dingin?".

pertanyaan(sakit_kepala, Y) :-
      Y = "Apakah Anda sering sakit kepala?".

pertanyaan(mimisan, Y) :-
      Y = "Apakah Anda sering mimisan?".

pertanyaan(mual, Y) :-
      Y = "Apakah Anda merasa mual?".

pertanyaan(demam, Y) :-
      Y = "Apakah Anda demam?".
```

5) Kemudian, buat file baru bernama *pakar_malaria_gui.py*

6) Ketik code berikut untuk membuat tampilan GUI dari aplikasi.

```
import tkinter as tk
from tkinter import ttk

# Inisialisasi window utama
root = tk.Tk()
root.title("Sistem Pakar Diagnosis Penyakit Malaria")

# Inisialisasi frame utama
mainframe = ttk.Frame(root, padding="3 3 12 12")
mainframe.grid(column=0, row=0, sticky=(tk.N, tk.W, tk.E, tk.S))
root.columnconfigure(0, weight=1)
root.rowconfigure(0, weight=1)

# Membuat widget yang diperlukan
ttk.Label(mainframe, text="Aplikasi Diagnosa Penyakit Malaria",
font=("Arial", 16)).grid(column=0, row=0, columnspan=3)

ttk.Label(mainframe, text="Kolom Pertanyaan:").grid(column=0, row=1)

kotak_pertanyaan = tk.Text(mainframe, height=4, width=40,
state=tk.DISABLED)

kotak_pertanyaan.grid(column=0, row=2, columnspan=3)

no_btn = ttk.Button(mainframe, text="Tidak", state=tk.DISABLED,
command=lambda: jawaban(False))

no_btn.grid(column=1, row=3, sticky=(tk.W, tk.E))

yes_btn = ttk.Button(mainframe, text="Ya", state=tk.DISABLED,
command=lambda: jawaban(True))

yes_btn.grid(column=2, row=3, sticky=(tk.W, tk.E))

start_btn = ttk.Button(mainframe, text="Mulai Diagnosa",
command=mulai_diagnosa)

start_btn.grid(column=1, row=4, columnspan=2, sticky=(tk.W, tk.E))

# Tambah padding ke setiap widget
for widget in mainframe.winfo_children():
       widget.grid_configure(padx=5, pady=5)

# Menjalankan GUI
root.mainloop()
```

7) Jalankan program. Akan muncul tampilan aplikasi seperti gambar berikut.



8) Kemudian, tambahkan library yang diperlukan selanjutnya. Ketik code berikut di bagian atas program bersama dengan statement import lainnya.

```
from pyswip import Prolog
from tkinter import messagebox
```

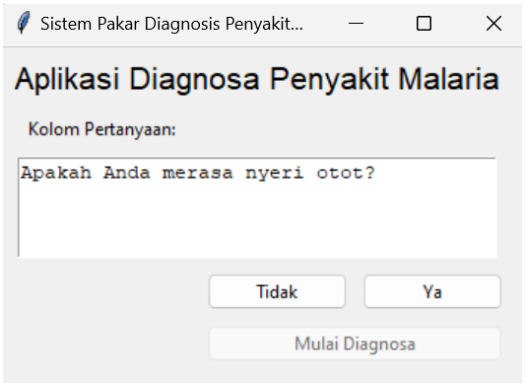
9) Kemudian, ketik code berikut untuk menghubungkan Python dengan Prolog dan mengimplementasikan fitur diagnosa. Ketik code-nya di bagian setelah import library dan sebelum implementasi GUI.

```
# Inisialisasi prolog
prolog = Prolog()
prolog.consult("pakar_malaria_gui.pl")

penyakit = list()
gejala = dict()
index_penyakit = 0
index_gejala = 0
current_penyakit = ""
current_gejala = ""

def mulai_diagnosa():
	global penyakit, gejala, index_penyakit, index_gejala

	# Bersihkan database prolog
	prolog.retractall("gejala_pos(_)")
	prolog.retractall("gejala_neg(_)")

	start_btn.configure(state=tk.DISABLED)
	yes_btn.configure(state=tk.NORMAL)
	no_btn.configure(state=tk.NORMAL)

	# Mendapatkan daftar penyakit dan gejala
	penyakit = [p["X"].decode() for p in
list(prolog.query("penyakit(X)"))]
	for p in penyakit:
		gejala[p] = [g["X"] for g in list(prolog.query(f"gejala(X,
\"{p}\")"))]

	index_penyakit = 0
	index_gejala = -1
```

```
	pertanyaan_selanjutnya()

def pertanyaan_selanjutnya(ganti_penyakit = False):
	global current_penyakit, current_gejala, index_penyakit,
index_gejala

	# Atur index penyakit
	if ganti_penyakit:
		# Ganti ke penyakit selanjutnya
		index_penyakit += 1
		index_gejala = -1

	# Apabila daftar penyakit sudah habis berarti tidak terdeteksi
penyakit
	if index_penyakit >= len(penyakit):
		hasil_diagnosa()
		return
	current_penyakit = penyakit[index_penyakit]

	# Atur index gejala
	index_gejala += 1

	# Apabila semua gejala dari penyakit habis, berarti terdeteksi
penyakit tsb
	if index_gejala >= len(gejala[current_penyakit]):
		hasil_diagnosa(current_penyakit)
		return
	current_gejala = gejala[current_penyakit][index_gejala]

	# Cek status gejala di database prolog
	if list(prolog.query(f"gejala_pos({current_gejala})")):
		pertanyaan_selanjutnya()
		return
	elif list(prolog.query(f"gejala_neg({current_gejala})")):
		pertanyaan_selanjutnya(ganti_penyakit=True)
		return

	# Mendapatkan pertanyaan baru
	pertanyaan = list(prolog.query(f"pertanyaan({current_gejala},
Y)"))[0]["Y"].decode()

	# Set pertanyaan ke kotak pertanyaan
	tampilkan_pertanyaan(pertanyaan)

def tampilkan_pertanyaan(pertanyaan):
	kotak_pertanyaan.configure(state=tk.NORMAL)
	kotak_pertanyaan.delete(1.0, tk.END)
	kotak_pertanyaan.insert(tk.END, pertanyaan)
	kotak_pertanyaan.configure(state=tk.DISABLED)

def jawaban(jwb):
	if jwb:
		prolog.assertz(f"gejala_pos({current_gejala})")
		pertanyaan_selanjutnya()
	else:
		prolog.assertz(f"gejala_neg({current_gejala})")
		pertanyaan_selanjutnya(ganti_penyakit=True)
```

```
def hasil_diagnosa(penyakit = ""):
	if penyakit:
		messagebox.showinfo("Hasil Diagnosa", f"Anda terdeteksi
{penyakit}.")
	else:
		messagebox.showinfo("Hasil Diagnosa", "Tidak terdeteksi
penyakit.")

	yes_btn.configure(state=tk.DISABLED)
	no_btn.configure(state=tk.DISABLED)
	start_btn.configure(state=tk.NORMAL)
```

10) Jalankan program. Klik tombol “Mulai Diagnosa” untuk memulai.



11) Lakukan uji coba dengan mensimulasikan gejala-gejala dari salah satu penyakit.

TUGAS

Buat sebuah program sistem pakar (berbasis console atau GUI) dengan topik yang berbeda! Upload source code keempat percobaan di atas dan tugas ke GitHub dengan nama repositori sebagai berikut,

NIM-PraktikumKB-Pertemuan4

Selamat mengerjakan!