**PRAKTIKUM KECERDASAN BUATAN**

MENCARI JALUR TERCEPAT MENGGUNAKAN ALGORITMA SEARCHING

“Disusun Untuk Memenuhi Tugas Mata Kuliah Praktikum Kecerdasan Buatan”

Dosen pengampu : Fitri Nuraeni, S.Kom., M.Kom.



Disusun Oleh :

|  |  |
| --- | --- |
| Yoga Agustiansyah | 2206050 |
| Ahmad Nur Sahid | 2206042 |
| Mochammad Dimas Nugraha | 2206043 |
| Ramada Gumelar | 2206044 |

Teknik Informatika B

**PROGRAM STUDI TEKNIK INFORMATIKA**

**INSTITUT TEKNOLOGI GARUT**

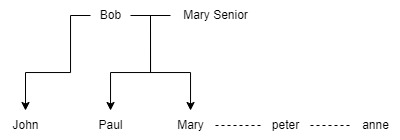
**2024**

**L5**

1. **Latihan FC 01**

|  |
| --- |
| saudara(john, mary).  saudara(john, paul).  saudara(mary, peter).  saudara(peter, anne).  orang\_tua(bob, john).  orang\_tua(bob, mary).  orang\_tua(bob, paul).  orang\_tua(mary\_senior, mary).  orang\_tua(mary\_senior, paul).  /\* aturan \*/  anak(X,Y):-  orang\_tua(Y,X).  keponakan(X,Y):-  anak(X,Z), saudara(Z,Y).  cucu(X,Y):-  anak(X,Z), orang\_tua(Z,Y).  paman(X,Y):-  saudara(X,Z), orang\_tua(Z,Y).  penalaran:-  write('Masukan nama orang: '),  read(Nama),  tulis\_hubungan(Nama), !.  tulis\_hubungan(Nama):-  anak(Nama, OrangTua),  write(Nama), write(' adalah anak dari '), write(OrangTua), nl,  fail.  tulis\_hubungan(Nama):-  keponakan(Nama, Paman),  write(Nama), write(' adalah keponakan dari '), write(Paman), nl,  fail.  tulis\_hubungan(Nama):-  cucu(Nama, Kakek),  write(Nama), write(' adalah cucu dari '), write(Kakek), nl,  fail.  tulis\_hubungan(Nama):-  paman(Nama, Keponakan),  write(Nama), write(' adalah paman dari '), write(Keponakan), nl,  fail.  tulis\_hubungan(\_).  :- penalaran. |

**Graph**



**Penjelasan syntax**

1. saudara(X, Y) dan orang\_tua(X, Y) adalah sebuah fakta yang di deklarasi
2. anak, keponakan, cucu, paman, adalah aturan
3. write untuk menulis / mencetak text ke terminal
4. read membaca fakta dari user
5. :- penalaran. Untuk malakukan pemanggilan terhadap predikat / aturan.

**Hasil**

Intruksi:

Panggil pengetahuan tersebut melalui perintah File<Consult pilih latihanFC01.pl

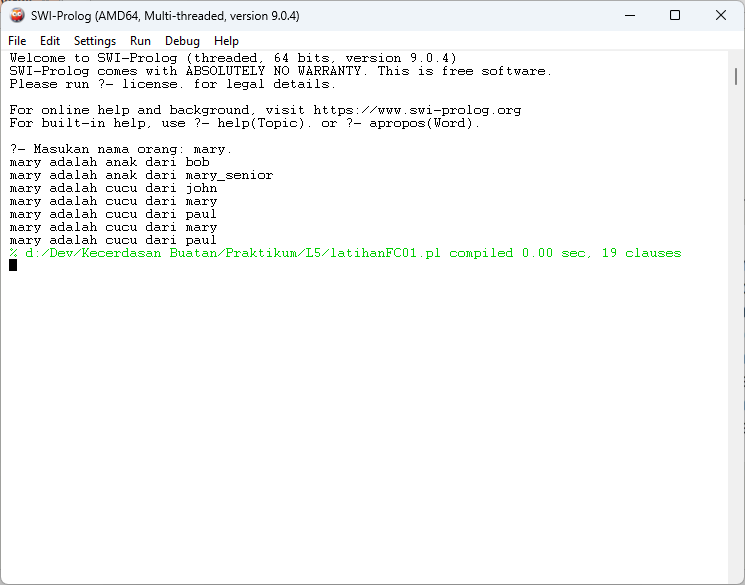
Jika sudah sukses terpanggil, ketikan perintah: penalaran.

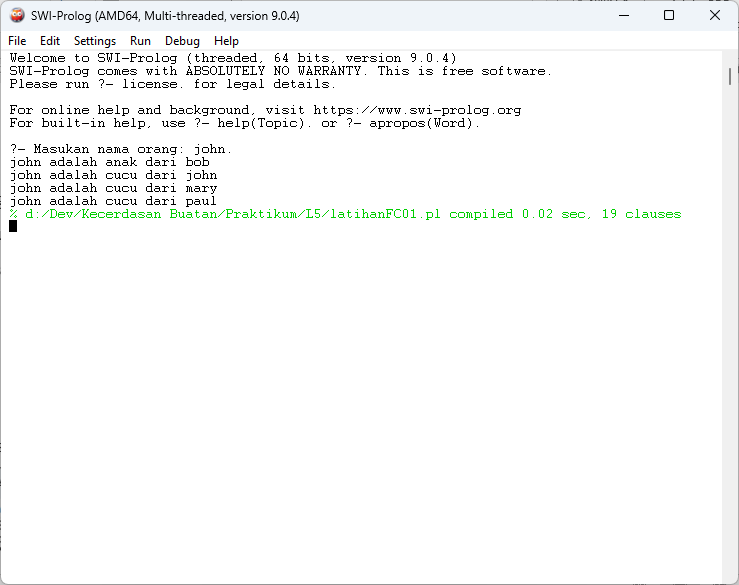
Coba masukan nama:

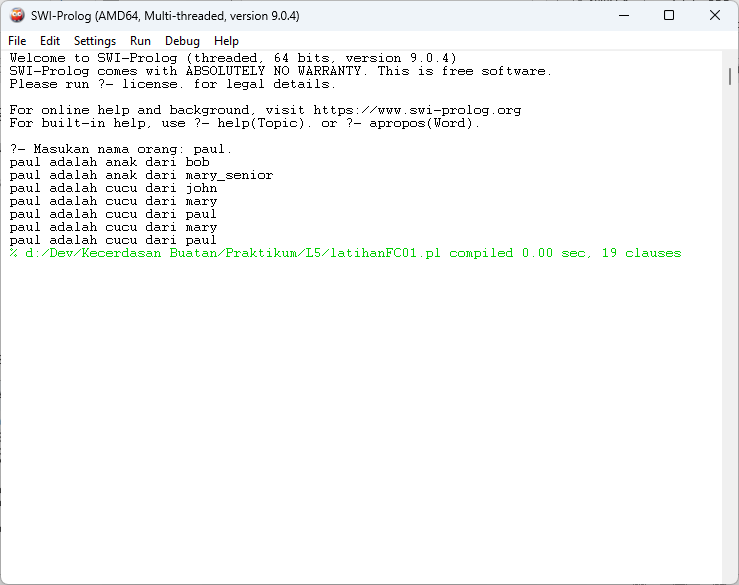
1. mary
2. john
3. paul

Screenshoot hasil inferensinya seperti gambar disamping

Hasil Intruksi:

****

****

****

Dari semua fakta yang telah diinputkan akan dicek satu persatu aturan-aturan yang ada dan kemudian dicetak ke terminal.

1. **Latihan FC02**

|  |
| --- |
| :- dynamic user\_fact/1.  :- dynamic infered\_fact/1.  % Predikat untuk menghapus semua fakta sebelum program dijalankan  clear\_facts :-  retractall(user\_fact(\_)),  retractall(infered\_fact(\_)).  % predikat untuk meminta input fakta dari pengguna  ask\_for\_facts :-  write('Masukan fakta-fakta awal (ketik "Sunny."/"Dry."/"Wet."/"Rainy."/"Selesai." untuk berhenti): '), nl,  read(Fact),  (  Fact \= selesai ->  (  assert(user\_fact(Fact)),  ask\_for\_facts  );  true  ).  % rules  rule(play) :- user\_fact(sunny), user\_fact(dry).  rule(not\_play) :- user\_fact(sunny), user\_fact(wet).  rule(not\_play) :- user\_fact(rainy).  % inference engine  infer :-  rule(X),  not(infered\_fact(X)),  assert(infered\_fact(X)),  fail.  infer.  % print  print\_facts :-  write('Facts: '), nl,  user\_fact(X),  write('- '), write(X), nl,  fail.  print\_facts :-  write('Infered Facts: '), nl,  infered\_fact(X),  write('- '), write(X), nl,  fail.  print\_facts.  % start  start :-  clear\_facts, %clear fact  ask\_for\_facts,  infer,  print\_facts. |

**Penjelesan Syntax**

1. :- dynamic user\_fact/1. dan :- dynamic infered\_fact/1.: Mendeklarasikan bahwa user\_fact/1 dan infered\_fact/1 adalah fakta yang bersifat dinamis, yang artinya fakta-fakta ini dapat diubah atau dihapus selama runtime program.
2. clear\_facts/0: Ini adalah sebuah predikat yang bertugas untuk menghapus semua fakta sebelum program dijalankan. Ini dilakukan dengan menggunakan retractall, yang menghapus semua fakta yang cocok dengan pola yang diberikan.
3. ask\_for\_facts/0: Predikat ini bertugas untuk meminta input fakta-fakta awal dari pengguna. Pengguna diminta untuk memasukkan fakta-fakta cuaca seperti Sunny, Dry, Wet, Rainy, atau mengetikkan Selesai untuk menghentikan input.
4. rule/1: Ini adalah aturan dalam bentuk predikat yang menentukan kapan suatu fakta dapat disimpulkan berdasarkan fakta-fakta yang telah dimasukkan oleh pengguna. Misalnya, aturan rule(play) menyatakan bahwa seseorang akan bermain jika cuaca Sunny dan Dry.
5. infer/0: Ini adalah mesin inferensi yang bertugas untuk mencoba mencari fakta-fakta yang dapat disimpulkan berdasarkan aturan-aturan yang telah didefinisikan. Proses inferensi dilakukan dengan memeriksa setiap aturan yang ada, kemudian mencoba mencocokkan fakta-fakta yang sudah diketahui dengan aturan tersebut. Jika suatu aturan berhasil dipenuhi, maka fakta tersebut akan disimpulkan dan dimasukkan ke dalam fakta yang telah disimpulkan (infered\_fact).
6. print\_facts/0: Predikat ini bertugas untuk mencetak semua fakta yang telah dimasukkan oleh pengguna (user\_fact) dan fakta yang telah disimpulkan (infered\_fact).
7. start/0: Ini adalah predikat utama yang memulai jalannya program. Pertama, semua fakta yang ada dihapus menggunakan clear\_facts, kemudian pengguna diminta untuk memasukkan fakta-fakta awal menggunakan ask\_for\_facts, selanjutnya dilakukan proses inferensi menggunakan infer, dan terakhir mencetak semua fakta yang ada menggunakan print\_facts.

**Hasil**

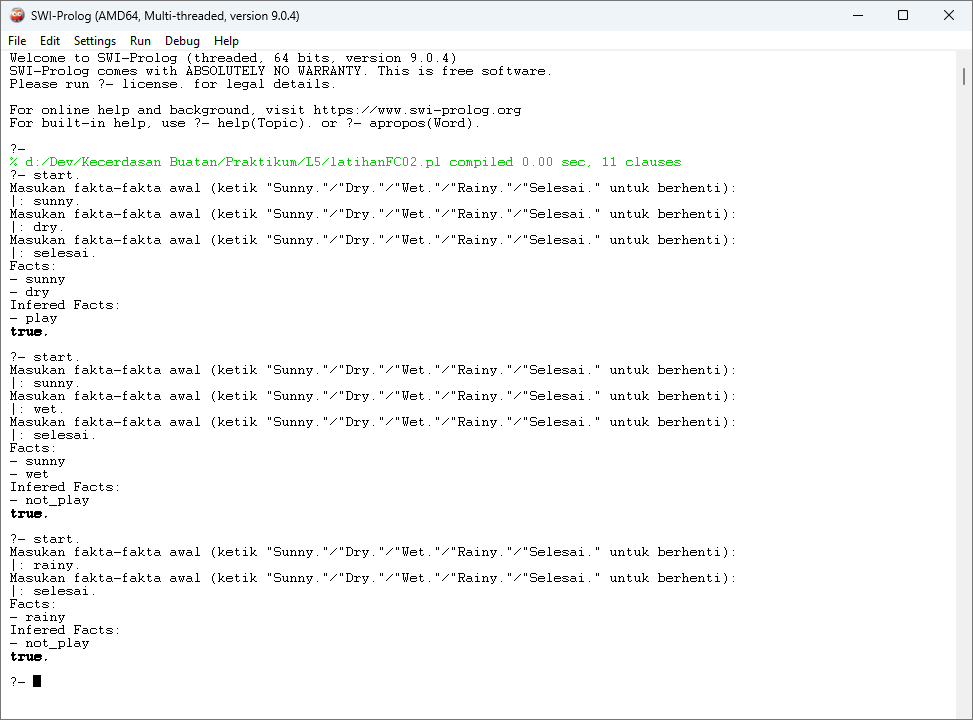
Panggil pengetahuan tersebut melalui perintah File<Consult pilih latihanFC02.pl

Jika sudah sukses terpanggil, ketikan perintah: start.

Coba masukan fact:

1. sunny & dry
2. Sunny & wet
3. rainy

Screenshoot hasil inferensinya seperti gambar disamping



Penjelasan Hasil

1. memasukkan sunny dan dry akan bernilai true karena fakta yang di inputkan sesuai dengan aturan rule(play) :- user\_fact(sunny), user\_fact(dry). . dengan metode forward chaining semua aturan yang ada akan dicoba dan menerapkan aturan-aturan tersebut pada fakta-fakta yang telah diketahui.
2. memasukkan sunny dan wet akan bernilai true dan infered not\_play. Dikarena pada Forward Chaining sistem akan memeriksa aturan-aturan yang ada dan mencoba menerapkan aturan-aturan tersebut pada fakta-fakta yang telah diketahui.
3. memasuukan rainy akan bernilai true dan infered not\_play. Sama seperti no 1 dan 2, Forward Chaining sistem akan memeriksa aturan-aturan yang ada dan mencoba menerapkan aturan-aturan tersebut pada fakta-fakta yang telah diketahui.
4. **Latihan BC 01**

|  |
| --- |
| % fakta  orangtua(ahmad, budi).  orangtua(ahmad, candra).  orangtua(budi, cindy).  orangtua(budi, david).  orangtua(candra, emily).  orangtua(candra, fahri).  orangtua(david, grace).  % Aturan  kakek(Kakek, Cucu) :-  orangtua(Kakek, Anak),  orangtua(Anak, Cucu).  saudara(Saudara1, Saudara2) :-  orangtua(Orangtua, Saudara1),  orangtua(Orangtua, Saudara2),  Saudara1 \== Saudara2.  paman(Paman, Keponakan) :-  saudara(Paman, Orangtua),  Orangtua(Orangtua, Keponakan).  cucu(Cucu, Kakek) :-  kakek(Kakek, Cucu).  cucu(Cucu, Kakek) :-  orangtua(Orangtua, Cucu),  cucu(Orangtua, Kakek). |

**Penjelesan Syntax**

Fakta

orangtua(ahmad, budi): Ahmad adalah orang tua dari Budi.

orangtua(ahmad, candra): Ahmad adalah orang tua dari Candra.

orangtua(budi, cindy): Budi adalah orang tua dari Cindy.

orangtua(budi, david): Budi adalah orang tua dari David.

orangtua(candra, emily): Candra adalah orang tua dari Emily.

orangtua(candra, fahri): Candra adalah orang tua dari Fahri.

orangtua(david, grace): David adalah orang tua dari Grace.

Aturan

kakek(Kakek, Cucu): Kakek adalah kakek dari Cucu jika Kakek adalah orang tua dari Anak dan Anak adalah orang tua dari Cucu.

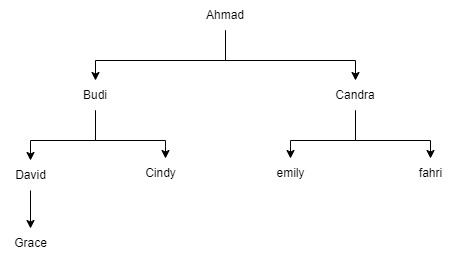
saudara(Saudara1, Saudara2): Saudara1 adalah saudara dari Saudara2 jika mereka memiliki orang tua yang sama dan mereka bukan orang yang sama.

paman(Paman, Keponakan): Paman adalah paman dari Keponakan jika Paman adalah saudara dari Orangtua dan Orangtua adalah orang tua dari Keponakan.

cucu(Cucu, Kakek): Cucu adalah cucu dari Kakek jika Kakek adalah kakek dari Cucu.

cucu(Cucu, Kakek): Cucu adalah cucu dari Kakek jika Orangtua adalah orang tua dari Cucu dan Orangtua adalah cucu dari Kake

**Graph**



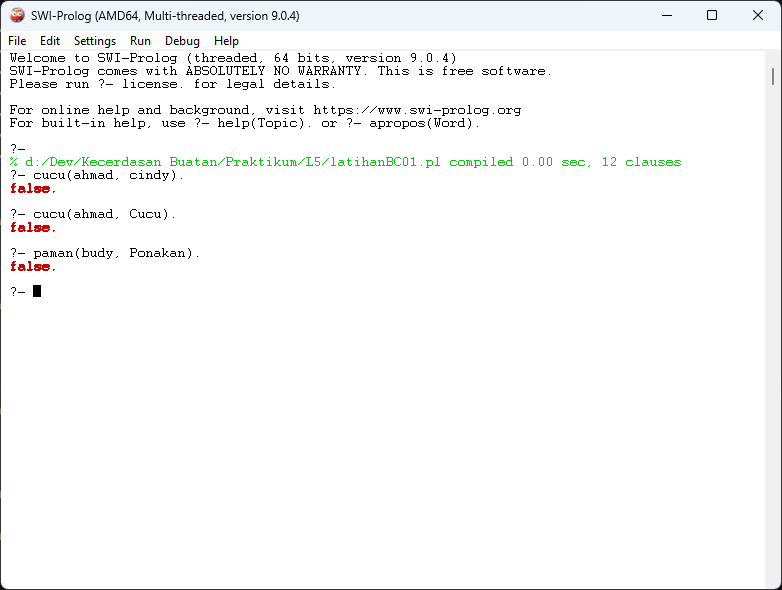
**Hasil**

Hasil instruksi:

Panggil pengetahuan tersebut melalui perintah **File<Consult** pilih **latihanBC01.pl**

Jika sudah sukses terpanggil, ketikan query:

1. cucu(ahmad, cindy).
2. cucu(ahmad, Cucu).
3. paman(budy, Ponakan).



Penjelasan Output

?- cucu(ahmad, cindy).

Akan false karena cindy bukan lah kakek dari ahmad. Bisa dilihat bahwa cindy lah cucu dari ahmad.

?- cucu(ahmad, Cucu).

Akan tetap false karena ahmad bukanlah cucu dari Cucu.

?- paman(budy, Ponakan).

Akan false karena budy tidak terdaftar di fakta. Seharusnya budi pake i bukan y.

1. **Latihan BC 02**

|  |
| --- |
| :- dynamic user\_fact/1.  :- dynamic inferred\_fact/1.  clear\_facts :-  retractall(user\_fact(\_)),  retractall(inferred\_fact(\_)).  % Predikat untuk meminta input fakta dari pengguna  ask\_for\_facts :-  write('Masukan fakta fakta awal (ketik "sunny."/"dry."/"wet."/"rainy."/"selesai." untuk berhenti): '), nl,  read(Fact),  (  Fact \= selesai ->  (  assert(user\_fact(Fact)),  ask\_for\_facts  );  true  ).  % RUles  % fakta play dapat dibuktikan jika sunny dan dry terpenuhi  can\_prove(play) :- user\_fact(sunny), user\_fact(dry).  % fakta no\_play dapat dibuktikan jika sunny dan wet terpenuhi  can\_prove(not\_play) :- user\_fact(sunny), user\_fact(wet).  % fakta no\_play dapat dibuktikan jika rainy terpenuhi  can\_prove(not\_play) :- user\_fact(rainy).  infer(Fact) :-  can\_prove(Fact),  assert(inferred\_fact(Fact)),!.  infer(\_).  print\_facts :-  write('Facts:'), nl,  user\_fact(X),  write('- '),  write(X), nl,  fail.  print\_facts :-  write('Inferred Facts:'), nl,  inferred\_fact(X),  write('- '),  write(X), nl,  fail.  print\_facts.  start :-  clear\_facts,  ask\_for\_facts,  infer(play),  infer(not\_play),  print\_facts. |

**Penjelasan Syntax**

1. :- dynamic user\_fact/1. dan :- dynamic inferred\_fact/1.: Ini adalah directive Prolog yang mendeklarasikan bahwa user\_fact/1 dan inferred\_fact/1 adalah fakta dinamis yang bisa dimodifikasi selama runtime.
2. clear\_facts/0: Ini adalah sebuah predikat yang menghapus semua fakta yang telah didefinisikan sebelumnya baik yang dimasukkan oleh pengguna (user\_fact) maupun yang disimpulkan (inferred\_fact).
3. ask\_for\_facts/0: Ini adalah sebuah predikat yang meminta pengguna untuk memasukkan fakta-fakta awal, seperti cuaca (sunny, dry, wet, rainy). Pengguna diminta untuk memasukkan fakta dengan mengetikkan salah satu dari opsi tersebut atau mengetikkan "selesai." untuk menghentikan input.
4. can\_prove/1: Ini adalah sebuah predikat yang menentukan apakah suatu fakta dapat dibuktikan berdasarkan fakta-fakta yang telah dimasukkan oleh pengguna. Misalnya, fakta play dapat dibuktikan jika cuaca sunny dan dry, sedangkan fakta not\_play dapat dibuktikan jika cuaca sunny dan wet atau jika cuaca rainy.
5. infer/1: Ini adalah predikat yang digunakan untuk menginferensi fakta berdasarkan aturan yang telah didefinisikan dalam can\_prove/1. Jika suatu fakta dapat dibuktikan, fakta tersebut akan disimpulkan dan dimasukkan ke dalam fakta yang disimpulkan (inferred\_fact).
6. print\_facts/0: Ini adalah sebuah predikat yang digunakan untuk mencetak semua fakta yang telah dimasukkan oleh pengguna (user\_fact) dan fakta yang disimpulkan (inferred\_fact).

start/0: Ini adalah predikat utama yang memulai proses inferensi. Pertama, semua fakta yang ada dihapus menggunakan clear\_facts, kemudian pengguna diminta untuk memasukkan fakta-fakta awal menggunakan ask\_for\_facts, setelah itu dilakukan inferensi untuk fakta play dan not\_play, dan akhirnya semua fakta yang ada dicetak menggunakan print\_facts.

**Hasil**

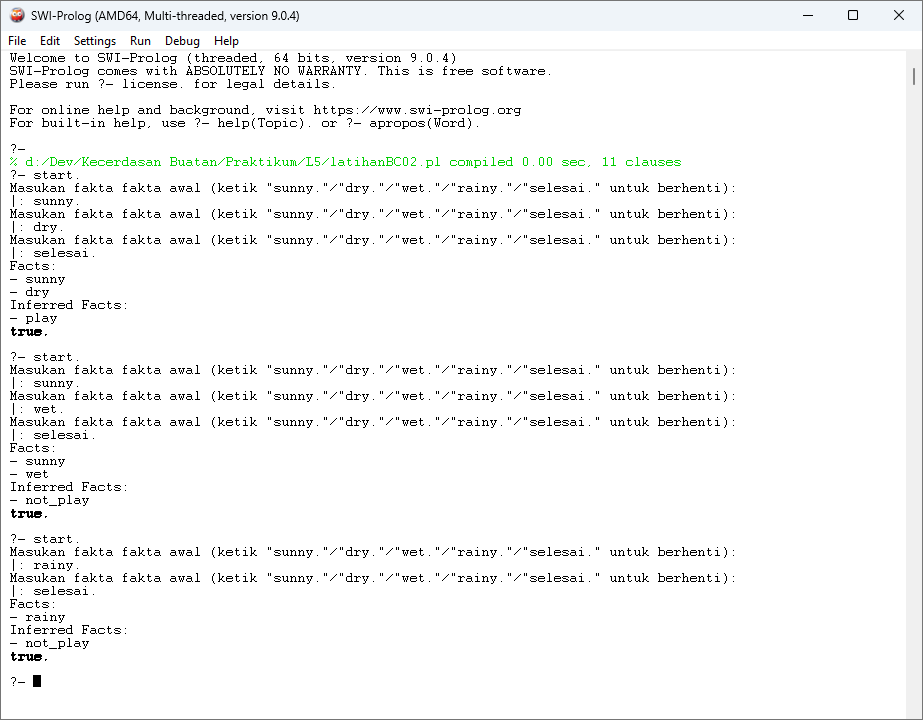
Intruksi Hasil

Panggil pengetahuan tersebut melalui perintah **File<Consult** pilih **latihanBC02.pl**

Jika sudah sukses terpanggil, ketikan perintah: **start.**

Coba masukan fact:

1. sunny & dry
2. Sunny & wet
3. Rainy



**Penjelesan Output**

1. memasukkan fakta sunny & dry akan menghasilkan true dan inferred facts play. Hal ini bisa didapat kerena pada backward chaining semua kesimpulan yang ada di coba samapi menemukan fakta-fakta sunny & dry adalah inferred facts play
2. memasukkan fakta sunny & wet akan menghasilkan true dan inferred facts not play. Kerena menggunakan metode Backward Chaining kesimpukan atau aturan dari can\_prove di coba satu satu sampai selesai.
3. memasukkan fakta rainy akan akan menghasilkan true dan inferred facts not play. Sama seperti no 1 dan 2 semua kesimpualan atau aturan terlebih dahulu di coba satu-satu sampai selesai.

**Kesimpulan**

1. **Forward Chaining (Pencarian Maju)**

Dalam forward chaining, sistem memeriksa aturan-aturan yang ada dan mencoba menerapkan aturan-aturan tersebut pada fakta-fakta yang telah diketahui. Jika suatu aturan dapat diterapkan pada fakta yang ada, maka fakta baru akan dihasilkan dan dimasukkan ke dalam basis pengetahuan.

1. **Backward Chaining (Pencarian Mundur)**

Dalam backward chaining, sistem mulai dengan tujuan yang ingin dicapai. Kemudian, sistem mencari aturan-aturan yang dapat mendukung tujuan tersebut, dan terus mundur ke belakang sampai menemukan fakta-fakta dasar yang diperlukan untuk mendukung tujuan tersebut.