Fakulti Teknologi Maklumat & Komunikasi
Universiti Teknikal Kebangsaan Malaysia Melaka (UTeM)

BITI2113
Logic Programming

Laboratory 8 - Built-in operators and Database Manipulation

An Investment Advisory System

You are required to develop a simple investment advisory system which your system will be able to advise potential investors which stock to buy. There are three type of stocks: **oil, computers or gold.** The advice is based on the type of portfolio the investor is to maintain. The investors can be classified into one of the following categories, based on the type of portfolio: **high risk, moderate risk, or stable risk portfolios**. Table 1 below shows the list of portfolio and which stock the investors may buy.

Table 1: Portfolio and its respective stock

Portfolio	Stocks to Invest
Stable Risk	Oil
Moderate Risk	Gold
High Risk	Computers

The investor is on **stable risk** portfolio if the investor is married, his/her income is less or equal to RM50,000.00 and has mortgage more than RM 50,000.00.
If the investor is single, his/her age is less than 50, and his/her salary is greater than RM35,000.00 then the portfolio is **high risk**.
To be categorized into the **moderate risk** portfolio, the investor should be

i) married, his/her income is less or equal to RM50,000.00 and has mortgage less than RM 50,000.00, or
ii) married and his/her income is greater than RM50,000.00, or
iii) single and his/her income is less or equal to RM35,000.00, or
iv) single, his/her income is greater than RM35,000.00 and his/her age is more than 50.

This model program shows how a decision tree knowledge structure can be implemented in Prolog.

A sample classification tree incorporating the knowledge used for determining the portfolio type is shown in Figure 1.

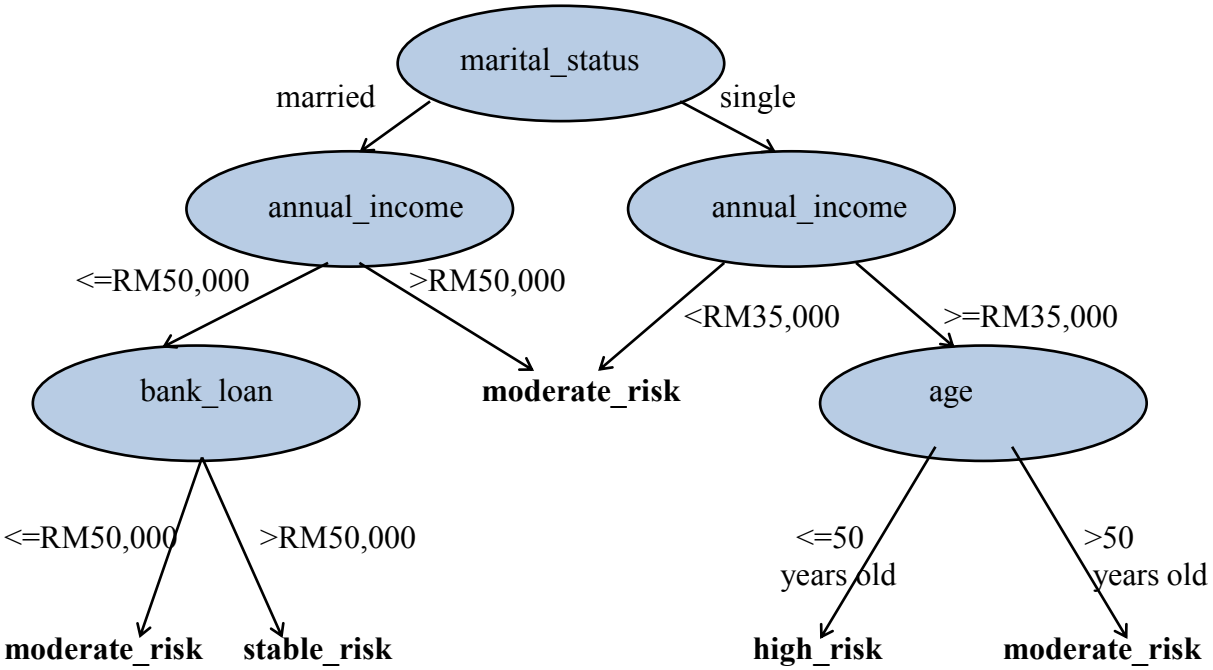

Figure 1: Portfolio Classification

Below is the sample input/output sessions run from a complete Prolog program.

Sample Sessions

```
| ?- main.
What is your name? |: ali.
What is your marital status: married or single? |: married.
What is your annual income? (RM) |: 50000.
What is your mortgage? (RM) |: 50000.
Name: ali
You are advised to invest in: gold

What is your name? |: tan.
What is your marital status: married or single? |: married.
What is your annual income? (RM) |: 55000.
Name: tan
You are advised to invest in: gold

What is your name? |: rama.
What is your marital status: married or single? |: married.
What is your annual income? (RM) |: 40000.
What is your mortgage? (RM) |: 60000.
Name: rama
You are advised to invest in: oil

What is your name? |: ahmad.
What is your marital status: married or single? |: single.
What is your annual income? (RM) |: 30000.
Name: ahmad
```

You are advised to invest in: gold

What is your name? |: chong.
What is your marital status: married or single? |: single.
What is your annual income? (RM) |: 35000.
What is your age? |: 30.
Name: chong
You are advised to invest in: computers

What is your name? |: sammy.
What is your marital status: married or single? |: single.
What is your annual income? (RM) |: 38000.
What is your age? |: 55.
Name: sammy
You are advised to invest in: gold

Below is the incomplete Prolog program that contains a rule based representation of the investment knowledge. The program also contains an input/output interface module to extract the needed information about income, mortgage, age and marital status from the user.

Incomplete Prolog Program

```
%-------------------------------------------------------------------------------
% The main predicate: Define dynamic predicates.

main:-  dynamic(marital_status/2),
        dynamic(income/2),
        dynamic(mortgage/2),
        dynamic(age/2),
        find_out(Name, Invest).

%-------------------------------------------------------------------------------
% find_out/2 - get data from users

find_out(Name, Invest) :-   write('What is your name? '),
                            read(Name),
                            invest(Name, Invest),
                            write('Name: '), write(Name), nl,
                            write('Invest: '), write(Invest), nl, nl,
                            find_out(New_Name, New_Invest).
```

```
%-----------------------------------------------------------------------------
% invest/2 – provide a suggestion to invest, either oil, telecommunications, or
% computers based on information on risk.

invest(Name, oil) :- stable_risk(Name), !.

invest(Name, gold) :- moderate_risk(Name), !.

invest(Name, computers) :- high_risk(Name).

%-----------------------------------------------------------------------------

% Please write the rule stable_risk(Name) here.

%-----------------------------------------------------------------------------

% Please write the rule moderate_risk(Name) here.

%-----------------------------------------------------------------------------

% Please write the rule high_risk(Name) here.
%-----------------------------------------------------------------------------

%-----------------------------------------------------------------------------
% ask_marital_status/2 – find information on a user's marital status

ask_marital_status(Name, Marital_Status) :-
        marital_status(Name, Marital_Status), !.

ask_marital_status(Name, Marital_Status) :-
        write('What is your marital status: married or single? '),
        read(Marital_Status),
        assert(marital_status(Name, Marital_Status)).

%-----------------------------------------------------------------------------
% ask_income/2 – find information on a user's annual income

ask_income(Name, Income) :- income(Name, Income), !.

ask_income(Name, Income) :- write('How much is your annual income? (RM) '),
                            read(Income),
                            assert(income(Name, Income)).
```

```
%-----------------------------------------------------------------------------------------
% ask_mortgage/2 – find information on a user's mortgage

ask_mortgage(Name, Mortgage) :- mortgage(Name, Mortgage), !.

ask_mortgage(Name, Mortgage) :- write('How much is your mortgage? (RM) '),
                                read(Mortgage),
                                assert(mortgage(Name, Mortgage)).

%-----------------------------------------------------------------------------------------
% ask_age/2 – find information on a user's age

ask_age(Name, Age) :- age(Name, Age), !.

ask_age(Name, Age) :- write('What is your age? '),
                      read(Age),
                      assert(age(Name, Age)).
```