```
% Homework 2 (hw2.txt)-CSC429/629-Fall 2022
% Due: on Canvas Oct. 13, 2022 11:59 pm.
% Total points: 5
% Lateness penalty: -1 per day.

% This is a 1 or 2 person assignment. Type your name(s) below:
% 1.Ege Keser
% 2.Muizz Muhammad
% If 2 persons work on this, they only need to turn in
% one paper copy, but both of you should submit a
% duplicate copy on Canvas under each of your names.
% You may not collaborate with anyone except your partner.
% By submitting this assignment, you implicitly agree to
% abide by the Academic Integrity Policy.

% Edit this file to add your solution in Step 1 and 2 below.
% Make your solution readable using indentation and white
% space. Do not use a line length that will wrap when printed.
% To run this file, rename it with a .pl extension if your are
% using SWI Prolog, or the proper extension for whatever Prolog
% you are using. (You may use any standard Prolog interpreter.)

% What to turn in:
% 1. Submit a copy of your program on Canvas named hw2.txt.
%    The digital copy will be used in case the grader needs to
%    run your program to verify that it works, and to determine
%    the submission time if late.  Each student on a team should
%    submit a duplicate copy on Canvas under his/her own name.
% 2. Submit a copy of your screen shots on Canvas as
%    a .pdf file. Turn in one document with all
%    screen shots. Do not use a phone to photograph the screen.
%    Eact student on a team should submit a copy on Canvas
%    under his/her own name.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   This homework uses a forward-chaining (FC) rule interpreter.

%   Instructions for using the FC rule interpreter:
%   1. Copy fig15_7_mod_1.6.17.pl (my version of Bratko's)
%      from Canvas to your computer.  (The 'not' operator
%      has been fixed.)
%   2. Start Prolog on your HW2 program. Before running HW2,
%      you need to tell Prolog to also use fig15_7_mod_1.6.17.pl
%      (in SWI Prolog, you would use menu commands: File
%      menu -> Consult -> fig15_7_mod_1.6.17.pl)
%   3. Warnings: if you run the program more than once,
%      retract the facts, or just QUIT prolog and start
%      over to flush the facts from memory.  Also, note that
%      the right-hand side of the expert system rules may not
%      contain 'and', 'or', or 'not'!

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Problem: Implement advisor, a forward-chaining expert system
% that advises UNCG undergraduate students on required and
% elective CSC courses to take. First the advisor has a dialog
% with the student about what CSC courses he has passed and
% what topics he likes. Then the advisor uses forward chaining
% rules to recommend courses to the student. The advisor should
% only recommend required courses if the student has passed
% their prerequisites and only recommend electives if they
% are on topics interesting to the student. (If you don't know
% about UNCG courses, go to the department's computer science
% web pages to get the information you need.)

% Here are some sample interactions. Note that the FC rule
% interpreter wrote "Derived: ", and "No more facts". You
% can ask the user for information using any questions that
% you wish.

% ?- advisor.
% What required courses have you passed?
% CSC130?: y
% CSC230?: y
% CSC330?: n
% ... etc.
% Derived: take_csc330
% Derived: take_csc350
% No more facts.

% ?- advisor.
% What required courses have you passed?
% ...
% CSC350?: y
% CSC330?: y
% CSC429?: n
% ...
% What topics do you like?:
% HCI?: n
% AI?: y
% ....
% Derived: take_CSC429
% No more facts


% Directives (must be at the top of your SWI Prolog program.
% Syntax may differ in other Prolog interpreters. This is
% needed so the program can assert the fact predicate.
:- dynamic fact/1.

% Operator definitions needed for forward chaining if-then rules.
:- op(800,fx,if).
:- op(700,xfx,then).
:- op(300,xfy,or).
:- op(200,xfy,and).


% Instructions:
```

```
advisor :- ask_user, forward.
% The above line runs your program. You will implement
% 'ask_user' in Step 1 below. Then 'forward' starts
% Bratko's forward chaining rule interpreter.


% Step 1 (2 points):
% Implement the Prolog predicate ask_user here.
% ask_user asks questions, reads in the answers, and
% asserts the user's answers as facts. Assert has
% this syntax: assert(fact(...)), where ... is some
% information such as passed_csc130.

ask_user :- nl, write('What required courses have you passed?'), nl,
write('CSC 130?'), read(Answer), process_csc130(Answer), nl, write('CSC
230?'), read(Answer2), process_csc230(Answer2), nl, write('CSC 250?'),
read(Answer3), process_csc250(Answer3), nl, write('CSC 261?'),
read(Answer4), process_csc261(Answer4), nl, write('CSC 330?'),
read(Answer5), process_csc330(Answer5), nl, write('CSC 350?'),
read(Answer6), process_csc350(Answer6), nl, write('CSC 362?'),
read(Answer7), process_csc362(Answer7), nl, write('CSC 339?'),
read(Answer8), process_csc339(Answer8), nl, write('CSC 471?'),
read(Answer9), process_csc471(Answer9), nl, write('CSC 340?'),
read(Answer10), process_csc340(Answer10), nl, write('CSC 462?'),
read(Answer11), process_csc462(Answer11), nl, write('Which topics do you
like?'), nl, write('AI?'), read(Answer12), process_ai(Answer12), nl,
write('Big Data?'), read(Answer13), process_bigdata(Answer13), forward.

process_csc130(Answer) :- Answer=='y', assert(fact(passed_csc130)).
process_csc130(Answer) :- Answer=='n', assert(fact(notpassed_csc130)).
process_csc230(Answer2) :- Answer2=='y', assert(fact(passed_csc230)).
process_csc230(Answer2) :- Answer2=='n', assert(fact(notpassed_csc230)).
process_csc250(Answer3) :- Answer3=='y', assert(fact(passed_csc250)).
process_csc250(Answer3) :- Answer3=='n', assert(fact(notpassed_csc250)).
process_csc261(Answer4) :- Answer4=='y', assert(fact(passed_csc261)).
process_csc261(Answer4) :- Answer4=='n', assert(fact(notpassed_csc261)).
process_csc330(Answer5) :- Answer5=='y', assert(fact(passed_csc330)).
process_csc330(Answer5) :- Answer5=='n', assert(fact(notpassed_csc330)).
process_csc350(Answer6) :- Answer6=='y', assert(fact(passed_csc350)).
process_csc350(Answer6) :- Answer6=='n', assert(fact(notpassed_csc350)).
process_csc362(Answer7) :- Answer7=='y', assert(fact(passed_csc362)).
process_csc362(Answer7) :- Answer7=='n', assert(fact(notpassed_csc362)).
process_csc339(Answer8) :- Answer8=='y', assert(fact(passed_csc339)).
process_csc339(Answer8) :- Answer8=='n', assert(fact(notpassed_csc339)).
process_csc471(Answer9) :- Answer9=='y', assert(fact(passed_csc471)).
process_csc471(Answer9) :- Answer9=='n', assert(fact(notpassed_csc471)).
process_csc340(Answer10) :- Answer10=='y', assert(fact(passed_csc340)).
process_csc340(Answer10) :- Answer10=='n', assert(fact(notpassed_csc340)).
process_csc462(Answer11) :- Answer11=='y', assert(fact(passed_csc462)).
process_csc462(Answer11) :- Answer11=='n', assert(fact(notpassed_csc462)).
process_ai(Answer12) :- Answer12=='y', assert(fact(interested_ai)).
process_ai(Answer12) :- Answer12=='n', assert(fact(notinterested_ai)).
process_bigdata(Answer13) :- Answer13=='y',
assert(fact(interested_bigdata)).
```

```
process_bigdata(Answer13) :- Answer13=='n',
assert(fact(notinterested_bigdata)).


% Step 2 (3 points): Put your forward chaining rules here.
% See Canvas for examples of the proper syntax.  Also,
% see my notes at the top of this file.

if notpassed_csc130 then take_csc130.
if passed_csc130 and notpassed_csc230 then take_csc230.
if passed_csc130 and notpassed_csc250 then take_csc250.
if passed_csc130 and notpassed_csc261 then take_csc261.
if passed_csc230 and passed_csc250 and notpassed_csc330 then take_csc330.
if passed_csc230 and passed_csc261 and notpassed_csc362 then take_csc362.
if passed_csc250 and notpassed_csc350 then take_csc350.
if passed_csc330 and notpassed_csc339 then take_csc339.
if passed_csc330 and notpassed_csc471 then take_csc471.
if passed_csc330 and notpassed_csc340 then take_csc340.
if passed_csc350 and notpassed_csc452 then take_csc452.
if passed_csc340 and passed_csc362 and notpassed_csc462 then take_csc462.
if passed_csc330 and passed_csc350 and interested_ai then take_csc429.
if passed_csc330 and interested_bigdata then take_csc410.




% Step 3: Follow the instructions at the top of this file
% for loading and running the FC rule interpreter.

% Step 4: Take screen shots showing at least 3 different
% recommendations based on the user's answers.
```