CPSC 312

Assignment #3 - Puzzles

Due no later than 11:59pm on Tuesday, October 13th, 2015.

Please include your official full name, your student id as well as your ugrad id in your submission.

There are three questions in this assignment. Each question is worth 25 points. Full mark in this assignment is 50 points, so you can choose to solve any two questions that you like out of the three. The third one, if you write, will be your bonus point.

Note that your third question, if you choose to do one, has to be solved fully. You can't partially solve a problem and receive a bonus grade for it.

How to hand-in: The deliverables of this assignment are three individual prolog files; question1.pl, question2.pl and question3.pl. Please include the documentation asked by each question as comments inside these files (no need for a separate txt file). As before, code without documentation receives no grade. (question 1 only needs a copy of the program's output).

Use hand-in to submit your assignment: The course name is **cs312** and the assignment name **assign3**. Use these links to familiarize yourself with hand-in if not already familiar:

https://my.cs.ubc.ca/?q=node/1288

https://my.cs.ubc.ca/docs/hand-in

Collaboration and Academic Honesty: This assignment is meant to be done individually. This means no consulting with other students as well as other sources. The various exercises given in the class are your chance to discuss similar questions with your peers but the assignment must be a solo effort. The solution to most of these problems are not difficult to find. However, the purpose of this assignment is to prepare you for the midterm and the final exams where you will have to write similar programs without the aid of the internet, various logic programming textbooks, or your peers. So make sure you are using this opportunity to prepare yourself.

Question 1.
The text below is the first stanza of "O Fortuna", a medieval Latin poem, part of Carmina Burana collection. You can listen to the tune by the German Composer, Carl Orff here
Translation of each line is also given:

o fortuna	→	o fortune
velut luna	→	like the moon
statu variabilis,	→	youre changeable
semper crescis	→	ever waxing
aut decrescis;	→	and waning
vita detestabilis	→	hateful life
nunc obdurat	→	first oppresses
et tunc curat	→	and then soothes
egestatem,	→	poverty
potestatem	→	power
dissolvit ut glaciem.	→	melts like ice

a) You're going to compile a dictionary out of the corresponding words of Latin and English in the given text.
Your dictionary will be a data structure that contains static hand-coded data.
Once you have a dictionary, write a lookup function, that given each latin word from the above text, searches the dictionary and returns the english equivalent.
The words are mostly in the same order on the left and the right hand side, but not always.
It should be fairly easy to guess which word goes with which, but if you think you're completely unfamiliar with latin and can't guess what matches what, be sure to consult a dictionary or google translate.
write all the words in small cases for simplicity; dropping apostrophes is okay.

here are a few hints to help you write your dictionary:
1. what is the one and only data structure we use in prolog?
2. parentheses, structured data, or lists can be used to represent pairs.
3. every line of code in prolog has to be a fact or a rule. format everything as a relation.
4. wrap atoms that have space in them in single quotes.

b) now write a translate function that takes a sentence, in the form of a list of words, in latin and translates it word by word to english, returning the english sentence also as a list of words. don't worry about wrong word order in translation at this stage.

Query your translate function first with the first three lines of the poem, separately, and include the output in your solution.
Additionally, query it with these phrases:
"o luna statu velut fortuna"
"variabilis fortuna"
"statu velut luna"
"potestatem nunc curat tunc obdurat"
"egestatem detestabilis"
"statu crescis et decrescis"

include the output for each case.

c) Now, you will write the same program and run the same queries against it, except that this time you will not hand-code the dictionary, instead you will dynamically load the pairs using infinite lists (see lecture from Sep 24th).
Instead of declaring the dictionary as a static dataset, you need to pass it to your program as an argument. Have it start as an unknown variable and gradually through unification have values added to it. Remember that the scope of any unification is the current query. The dictionary will be empty again at the next call.
Make sure you have a handy way of filling in the dictionary, before making each query. Alternatively, you can run all of the queries in one go as well.
This time your output should also print the contents of the dictionary. Include the new code and the outputs in your solutions.

d) Now let's expand your translate function form section (b) so that it can be used for different languages.
These lines are from a poem by the German poet, Fredrich Schiller, "Ode to Joy", composed by Beethoven into a movement of his Symphony No.9. You can listen to it here:

(simplified text):		(simplified) translation:
Freude, schöner Götter funken	→	Joy, beautiful spark of gods,
Tochter aus Elysium,	→	Daughter from Elysium,
Wir betreten feuer trunken,	→	We enter, drunk with fire,
Himmlische, dein Heiligtum!	→	heavenly being, your sanctuary!
Deine Zauber binden wieder	→	Your magic brings together
Was die Mode streng geteilt;	→	what fashion has sternly divided.
Alle Menschen werden Brüder,	→	All men become brothers,
Wo dein sanfter Flügel weilt.	→	wherever your gentle wings hover.

Manually add word pairs from only the first four lines of this translation to a *new* german dictionary. Once again, if you can't guess which word corresponds with which consult a german dictionary such as (http://dictionary.reverso.net/german-english) and once again adopt small cases only.
Once you have this new dictionary, modify your lookup and translate (from section a) to be able to translate text from either languages. The user will provide the name of the source language together with the sentence.

Query your translation with these sentences and provide the output:

1) each of the four lines of the german poem.
2) the second line of the latin poem.

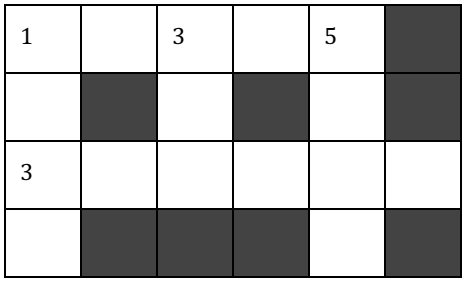
Then query your program with these sentences, but this time do not provide the source language. Put an unknown variable in its place. Your program should still be able to produce the correct result.

1) "dein schöner heiligtum"
2) "feuer götter aus elysium"
3) "dein funken"
4) "trunken himmlische"

e) What will happen if your translate program comes across a word it doesn't know? Modify your program from section d to replace any word it can't translate with ? and move on to the next word without failing.

Query it with: (you can include the source language here)
"deine schöner heiligtum"
"feure trunken"
"wo dein sanfter flügel weilt"
"dissolvit u glaciem"

Question 2.
Cross-Word puzzle



In this mini cross-word puzzle, five words of varying lengths are needed to solve the puzzle:
1 across (5 letters)
3 across (6 letters)
1 down (4 letters)
3 down (3 letters)
5 down (4 letters)

We have a set of vocabulary as such:
dog, run, top, five, four, lost, mess, unit, baker, forum, green, super, prolog, vanish, wonder, yellow

Write a prolog program containing a procedure called solution that finds 5 words out of the given vocabulary list fits each in the right place in the puzzle and outputs the solution.
You are free to format and represent your output (as well as the puzzle) however way you wish as long as these two pieces of information are evident in it: 1) the choice of words, 2) the place of each word or letter in the puzzle.
No use of built-in predicates is allowed. Any function you use has to be included with the code.
With your code and output, include also a brief description of how the puzzle is represented in your program and how the output of your program shows the information requested above. For every procedure in your program, write a one-line description of its declarative or procedural meaning, or simply what the procedure and its arguments do.

Question 3.
Word puzzle

Write a prolog program that solves this puzzle:

Yves, Dave, Brian, Ed and Mike live in a five-story building.
Yves doesn't live on the 5th floor
Dave doesn't live on the 1st floor
Brian doesn't live on the top or the bottom floor
Brian doesn't live on a floor adjacent to Mike or Dave
Ed lives on some floor above Dave

Who lives on what floor?

Provide with each line of your code a brief explanation of what it asserts as well as the output that shows the answer to the puzzle.

hints:
1. first and foremost think about how to generalize the information to represent it in logical statements.
2. remember that all the above assertions need to be true at the same time.
3. use arithmetic operators wherever you need.