CSE 240 Homework 12 Scheme & Prolog, Spring 2019 (50 points)

Due Saturday, April 13, 2019 at 11:59PM, plus a 24-Hour grace period

---

Introduction

The aim of this assignment is to make sure that you understand and are familiar with the concepts covered in the lectures. By the end of the assignment, you should have

- strong concept of functional paradigm.
- strong concept of pairs and list.
- understood the use of let-form and unnamed procedure.
- applied recursion to solve complex problems.
- Scheme higher order functions
- strong concept of logic / declarative paradigm;
- strong concept of facts, rules, and questions in Prolog.

Reading: textbook chapters 4 and 5, as well as the course notes (slides).

You are expected to do the majority of the assignment outside the class meetings. Should you need assistance, or have questions about the assignment, please contact the instructor or the TA during their office hours.

Preparation: Complete the **multiple choice questions** in the textbook exercise section. The answer keys can be found in the course Web site. These exercises can help you prepare for your weekly quiz and the exam. You are encourage to read the other exercise questions and make sure you understand these questions in the textbook exercise section, which can help you better understand what materials are expected to understand after the lectures and homework on each chapter.

You are expected to do the majority of the assignment outside the class meetings. Should you need assistance, or have questions about the assignment, please contact the instructor during office hours.

You are encouraged to ask and answer questions on the course discussion board. (However, **do not share your answers** in the course discussion board.)

Practice Exercises (no submission required)

Scheme Part:

1 What is the major difference between a while-loop and a recursive procedure? What is tail-recursion?

2 What are the major steps of developing a recursive procedure?

3 Given the following Scheme program:

3 Given the following program:

```
(define mymax1 (lambda (lst)
            (if (= (length lst) 1)
               (car lst)
               (let ((m (mymax1 (cdr lst))))
                 (if (> (car lst) m)
                    (car lst)
                    m)))
))
```

3.1 Add a procedure to handle the case of empty list, that is, if the given list is empty, the program should return "error: list empty".

3.2 Describe the four design steps used to solve the recursive problem and give the solution obtained in each step.

3.3 Compare and contrast the algorithm (approach) used in this program and the divide-and-conquer algorithm.

3.4 Use C to re implement the program.

4 Follow the four design steps to solve each of the following recursive problems, using Scheme and using C, respectively.

4.1 Count the number of the elements in a list

4.2 Sum a list

4.3 Reverse a list

5 For each recursive procedure you write, describe the four design steps that you use to solve the recursive problem and give the solution obtained in each step.

6 Fibonacci Numbers are defined by

$$fib(n) = \begin{cases} 0 & n = 0 \\ 1 & n = 1 \\ fib(n-1) + fib(n-2) & n \geq 2 \end{cases}$$

Follow the four design steps to write a recursive procedure to implement the function

6.1 Define the size-n problem

6.2 Define the stopping conditions and the return values.

6.3 Define the size-m problems

6.4 Construct the size-n solution from the hypothetic size-m solutions.

6.5 Give the complete procedure that computes Fibonacci Numbers for any given integer n >= 0.

Prolog Part:

1 Look up the Unix command table in Appendix B.1 and read the Prolog tutorial in B.4.

2 Log onto general.asu.edu and execute the Unix commands and Prolog exercises.

3 Follow the Prolog tutorial in Text Appendix B.5 or the tutorials given in the course Web site, start GNU Prolog and try following simple programs (build-in functions). Don't forget the **period** at the end of the statement.

```
| ?- write(hello).        % hello is a constant */
| ?- write(Hello).        % Hello is a variable. Its address will be displayed*/
| ?- write('hello world').    % a string is printed */
| ?- read(Y), write('The variable entered is '), write(Y), nl. /* nl prints a
     newline. Type a period and an enter at the end of the input */
| ?- X is 2+2.
| ?- Y is 5*8.
| ?- Y is 2**10.
| ?- length([a, b, x, y, 2, 45, z], L).
| ?- append([a, b, c, d], [4, 6, 8], LL).
| ?- append(X,Y,[a,b,c]). Then, type ";" to obtain all possible answers.
| ?- X is [1 | [2 | [3 | []]]], write(X). Explain the output.
```

Programming Exercise (50 points)

1. Create a recursive procedure called (**alternate** lst1 lst2 lst3) that returns a list with alternative values from the 3 given lists. Use comments to indicate in the code the four steps of the fantastic approach. Assume *that all 3 lists have the same length.* Hint: (append lst1 lst2 lst3) returns the appended list. [8 points]

Test case: (**alternate** '(1 2 3) '(a b c) '(m n o)) should give '(1 a m 2 b n 3 c o).

2. Create a recursive procedure called (pairs lst1 lst2) that returns a list of pairs from the 2 given lists. Use comments to indicate in the code the four steps of the fantastic approach. You can assume that both the lists have the same length. [8 points]

Test case: (**pairs** '(1 2 3) '(a b c)) should give '((1 . a) (2 . b) (3 . c)).

3. Using Scheme higher-order function map to implement the string encryption and description. [9 points]

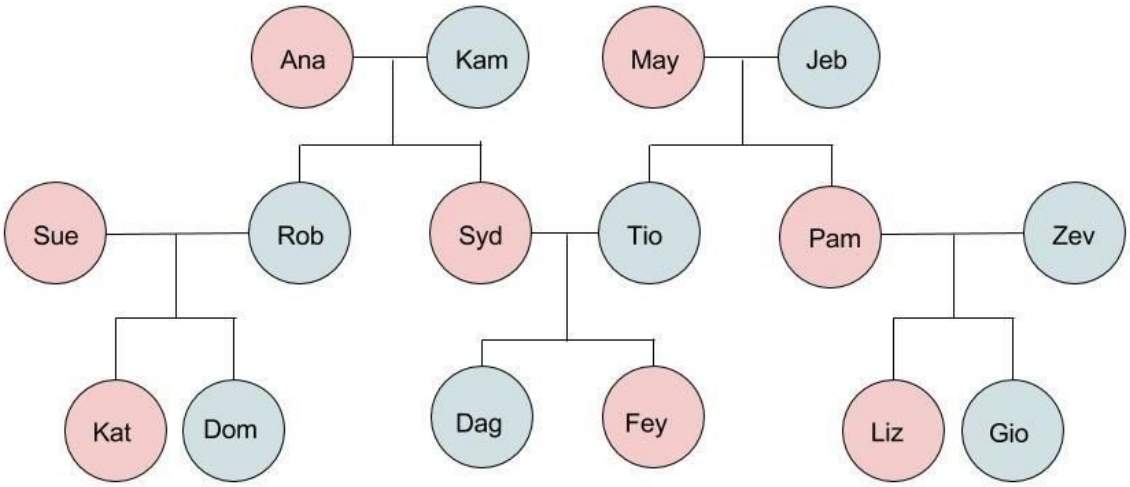
3.1 The encryption function must (1) take the encryption key (key can be between 1 and 4) from the keyboard using (read); and (2) encrypt alphabetic characters and digits only.

3.2 The decryption function must (1) take the encryption key from the keyboard using (read) and (2) decrypt the string generated from your encryption function.

```
Test cases:
(encrypt "Hello CSE240!")
(decrypt "Khoor FVH573!")
3
"Khoor FVH573!"
3
"Hello CSE240!"
```

Note: When you encrypt a valid character (letters and digits), you add a key to the character. Similarly, decryption subtracts the key from the characters. When you decrypt, you will have a different set of the valid characters, which is caused by +key. You can limit the allowed key value to be 1, 2, 3, and 4.

4. Consider the following diagram of a family tree: [25 points]




```
/* Database for family tree. It consists of facts and rules. */
/* A portion of the family tree has been implemented for you */

/* Facts */
male(kam).
male(rob).
female(ana).
female(syd).

father_of(kam, rob). /* kam is the father of rob */
father_of(kam, syd). /* kam is the father of syd */
mother_of(ana, rob). /* ana is the mother of rob */
mother_of(ana, syd). /* ana is the mother of syd */

/* Rules */
is_male(X) :-
     male(X);
     father_of(X, _).
```

Enter the program using a text editor, such as nano or pico, under Unix operating system and save the file as *family_tree.pl*. You may enter the program on your own computer using a simple editor, such as Notepad, and upload the program into your directory in *ASU General* server.

Compile the program using the Prolog command:

```
> gplc family_tree.pl
```

Enter GNU Prolog programming environment by executing the Unix command *gprolog*.

Execute the program *family_tree* by typing GNU Prolog command

```
|?- [family_tree].
```

To exit from GNU Prolog, type your end-of-file character at the main Prolog prompt ^d (Ctrl-d).

```
| ?- ^d
```

Ask questions by typing, e.g.:

```
|?- father_of(kam, rob).
|?- mother_of(ana, syd).
```

Please read Textbook Appendix B.4 Prolog Tutorial for more detail. You can also find a complete set of GNU Prolog commands at http://www.gprolog.org/manual/gprolog.html

Now, you can start to add your code into the program.

4.1 Complete the program by adding facts and rules for the remaining members on the family tree. A portion has already been completed for you above for clarification. Please pay close attention when adding the remaining family members. All letters should be lowercase. [5]

For all of the following questions, please label them. For example, if Question 1.0 asks you to define a rule called is_male (X) that returns "yes" (or "true") if X is the father of a member of the family, then your code should look like:

```
/* Question 1.0 */
is_male(X) :-
        male(X);
        father_of(X, _).
```

4.2 Define (add into the database) a rule called is_female(X) that returns "yes" (or "true") if X is a female or the mother of a member of the family. [4]

Note: the system will return a "yes", if it finds a "true" answer and there exists no more true answers. The system will return "true ?" if it finds a "true" answer and there are still possibly further matches. In this case, if you type "enter", it will return "yes" and stop. If you type ";", it will continue to search for further answers.

4.3 Define a rule called parent_of(X, Y) that returns “yes” (or “true”) if X is a parent of Y. [4]

4.4 Define a rule called sibling_of(X, Y) that returns “yes” (or “true”) if X is a sibling of Y. [4]

4.5 Define one rule called grandmother_of(X, Z) that returns “yes” (or “true”) if X is a grandmother of Z and one rule called grandfather_of(X, Z) that returns “yes” (or “true”) if X is a grandfather of Z. [4]

4.6 Define a rule called descendent_of(X, Y) that returns “yes” (or “true”) if X is a descendent of Y. Note: you will need to use recursion as well as the parent rule defined above. [4]

Grading of Programming Assignment

The grader will grade your program following these steps:

(1) Compile the code. If it does not compile, 20% of the points given will be deducted. For example, if there are 20 points possible, you will earn 16 points if the program fails to compile.

(2) The grader will read your program and give points based on the points allocated to each component, the readability of your code (organization of the code and comments), logic, inclusion of the required functions, and correctness of the implementations of each function.

(3) No need for comments if rule/procedure is very short (1-3 lines). Please insert comments otherwise.

What to Submit?

You are required to submit your solutions in a compressed format (.zip). Zip all files into a single zip file. Make sure your compressed file is labeled correctly: *lastname_firstname**12**.zip*.

For this home assignment, the compressed file MUST contain the following:

hw12.rkt	(Scheme program)
family_tree.pl	(Prolog program)

No other files should be in the compressed folder.

If multiple submissions are made, the most recent submission will be graded, even if the assignment is submitted late.

Where to Submit?

All submissions must be electronically submitted to the respected homework link in the course web page where you downloaded the assignment.

Late submission deduction policy

- No penalty for late submissions that are received within 24 hours after the deadline;
- 10% grade deduction for every day it is late after the grace period;
- No late submission after Tuesday at 11:59PM.

Academic Integrity and Honor Code.

You are encouraged to cooperate in study group on learning the course materials. However, you may not cooperate on preparing the individual assignments. Anything that you turn in must be your own work: You must write up your own solution with your own understanding. If you use an idea that is found in a book or from other sources, or that was developed by someone else or jointly with some group, make sure you acknowledge the source and/or the names of the persons in the write-up for each problem. When you help your peers, you should never show your work to them. All assignment questions must be asked in the course discussion board. Asking assignment questions or making your assignment available in the public websites before the assignment due will be considered cheating.
The instructor and the TA will **CAREFULLY** check any possible proliferation or plagiarism. We will use the document/program comparison tools like MOSS (Measure Of Software Similarity: http://moss.stanford.edu/) to check any assignment that you submitted for grading. The Ira A. Fulton Schools of Engineering expect all students to adhere to ASU's policy on Academic Dishonesty. These policies can be found in the Code of Student Conduct:

```
http://www.asu.edu/studentaffairs/studentlife/judicial/academic_integrity.htm
```

ALL cases of cheating or plagiarism will be handed to the Dean's office. Penalties include a failing grade in the class, a note on your official transcript that shows you were punished for cheating, suspension, expulsion and revocation of already awarded degrees.