CSc 372 — Comparative Programming Languages
Fall 2020 (McCann)

http://u.arizona.edu/~mccann/classes/372

Homework #1

(50 points)

Due Date: September 9th, 2020, at the beginning of class

Solutions to homeworks in this class should be written using a word processor and are to be electronically submitted as a single PDF file (using lectura's `turnin` utility).

Write complete, legible answers to each of the following questions. A problem identified as "C.q" references question q from the end of chapter C of the Louden/Lambert text, 3rd edition. Show your work, when appropriate, for possible partial credit. This is not a group project; do your own work. We will post our solutions $\geq$ 24 hours after the due date (remember, you can use one late day on homeworks, so we can't give solutions on the due date).

On the due date, by the start of class, submit your electronically–formatted PDF version of your solutions (the `turnin` folder is `cs372h1`). Solutions submitted more than 24 hours after the due date and time will not be accepted.

1. (5 points) 1.6
2. (5 points) 1.7
3. (5 points) 1.10
4. (5 points) 1.15
5. (10 points) We will be using Ruby as an example of an object–oriented language (as a contrast with Python and Java).

   Homepage: `https://www.ruby-lang.org`

   (a) Log into your CS account on `lectura.cs.arizona.edu` using your SSH client (if you are using Windows) or using SSH from the terminal window (if you are using MacOS, Linux, or other variety of UNIX) and, using a text editor, create a file named `fibonacci.rb` ('rb' for Ruby) with the following content, updating the documentation appropriately:

```
#!/usr/bin/ruby

#######################################################################
#   Assignment:  Homework #1:  Ruby Exercise
#       Author:  Your Name (Your E-mail Address)
#
#       Course:  CSc 372
#   Instructor:  L. McCann
#        TA(s):  Tito Ferra and Josh Xiong
#     Due Date:  September 9, 2020
#
#  Description:  A simple type-in exercise to ensure that students
#                are able to successfully use ruby on lectura.
#
#     Language:  Ruby
# Ex. Packages:  None.
#
# Deficiencies:  None.
#######################################################################

class DemoRuby
```

```

    def fibonacci_iterative (n)
        if n == 0 || n == 1
            n
        else
            older = 0
            old = 1
            for i in 2..n
                current = older+old
                older = old
                old = current
            end
            current
        end
    end

end

newObject = DemoRuby.new
puts "The first 10 Fibonacci numbers are:"
for i in 0..9
  puts newObject.fibonacci_iterative(i)
end
```

(b) Run the program: `ruby fibonacci.rb`

(Note: The first line of this file allows the program to be executed w/o typing "ruby". Here's how: (1) Tell the OS that the `fibonacci.rb` file is executable by typing this command at your shell prompt: `chmod +x fibonacci.rb` (2) Run the file: `./fibonacci.rb`)

(c) Copy/paste the output into your homework document.

(d) (OPTIONAL) If you expect to work on the upcoming Ruby assignment locally (on your own computer), take this opportunity to visit the Ruby site, download the current version for your OS, install it, and try this exercise using it.

6. (10 points) We will be using Haskell as an example of an functional language.

   Homepage: `https://www.haskell.org/`

   (a) Log into your CS account on `lectura.cs.arizona.edu` using your SSH client (if you are using Windows) or using SSH from the terminal window (if you are using MacOS, Linux, or other variety of UNIX) and, using a text editor, create a file named `fibonacci.hs` ('hs' for Haskell) with the following content, updating the documentation appropriately:

```
-----------------------------------------------------------------------
--   Assignment:  Homework #1:  Haskell Exercise
--       Author:  Your Name (Your E-mail Address)
--
--       Course:  CSc 372
--   Instructor:  L. McCann
--        TA(s):  Tito Ferra and Josh Xiong
--     Due Date:  September 9, 2020
--
--  Description:  A simple type-in exercise to ensure that students
--                are able to successfully use haskell on lectura.
--
--     Language:  Haskell (ghc)
-- Ex. Packages:  None.
--
-- Deficiencies:  None.
-----------------------------------------------------------------------

fibStep :: (Integer,Integer) -> (Integer,Integer)
fibStep(u,v) = (v,u+v)

fibPair :: Integer -> (Integer,Integer)
fibPair n
  | n == 0    = (0,1)
  | otherwise = fibStep (fibPair (n-1))
```

```

fastFib :: Integer -> Integer
fastFib = fst . fibPair

main = do
  putStrLn "The first 10 Fibonacci numbers are:"
  print ([fastFib(i) | i <- [0..9] ])
```

(b) Compile the program: `ghc fibonacci.hs`

(c) Run the program: `./fibonacci`

(d) Copy/paste the output into your homework document.

(e) (OPTIONAL) If you expect to work on the upcoming Haskell assignment locally (on your own computer), take this opportunity to visit the Haskell site, download the current version for your OS, install it, and try this exercise using it.

7. (10 points) We will be using SWI–Prolog to explore how a logic programming language works.

   Homepage: `http://www.swi-prolog.org/`

   (a) Log into `lectura.cs.arizona.edu` using SSH and, using a text editor, create a file named `connecticut.pl` (‘pl’ for Prolog) with this content, updating the documentation as appropriate:

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   Assignment:  Homework #1:  SWI-Prolog Exercise
%       Author:  Your Name (Your E-mail Address)
%
%       Course:  CSc 372
%   Instructor:  L. McCann
%        TA(s):  Tito Ferra and Josh Xiong
%     Due Date:  September 9, 2020
%
%  Description:  A simple type-in exercise to ensure that students
%                are able to successfully use SWI-Prolog on lectura.
%
%     Language:  Prolog (swipl)
% Ex. Packages:  None.
%
% Deficiencies:  None.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%% Facts:  Which Connecticut counties border which others?

bordering(fairfield,litchfield).
bordering(fairfield,newhaven).
bordering(litchfield,hartford).
bordering(litchfield,newhaven).
bordering(newhaven,middlesex).
bordering(newhaven,hartford).
bordering(hartford,tolland).
bordering(hartford,middlesex).
bordering(hartford,newlondon).
bordering(tolland,windham).
bordering(tolland,newlondon).
bordering(middlesex,newlondon).
bordering(windham,newlondon).

%%% Rules:

  % adjacent(X,Y) -- Counties X and Y share a border.

adjacent(X,Y) :- bordering(X,Y).
adjacent(X,Y) :- bordering(Y,X).

  % nearby(X,Y) -- Intent:  Counties X and Y are separated by no more than
  %                 one other county.

nearby(X,Y) :- bordering(X,Z), bordering(Z,Y).
```

(b) Launch SWI-Prolog: `swipl`

(c) Load the file: `[connecticut].`

(d) Type the following queries. Copy/paste both the queries and the displayed results into your homework document. Should SWI–Prolog display the result **true** without a period at the end, press the semicolon key and it will continue. Don't forget the trailing periods!

   i. `bordering(windham,newlondon).`
   ii. `bordering(hartford,tolland).`
   iii. `bordering(tolland,hartford).`
   iv. `adjacent(newhaven,middlesex).`
   v. `adjacent(middlesex,newhaven).`
   vi. `adjacent(newlondon,windham).`
   vii. `nearby(middlesex,newhaven).`
   viii. `nearby(newhaven,middlesex).`
   ix. `nearby(fairfield,hartford).`

(e) Exit SWI-Prolog: `halt.`

(f) (OPTIONAL) If you expect to work on the upcoming Prolog assignment locally (on your own computer), take this opportunity to visit the SWI–Prolog site, download the current version for your OS, install it, and try this exercise using it.