

Homework #1

(50 points)

Due Date: January 20th, 2017, 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 ≥ 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, hand-in a printout of your solutions **and** submit your electronically-formatted version of your solutions (the turnin folder is `cs372h1`). If you need to submit your solutions within the 24-hour late window, place your printout in Dr. McCann's mailbox in CS 713 as soon as you are able to do so. Solutions submitted more than 24 hours after the due date and time will not be accepted.

1. (5 points) 1.6
2. (5 points) 1.10
3. (5 points) 1.15
4. (5 points) 1.16

5. (10 points) We will be using Ruby as an example of an object-oriented language (to contrast with Java).

Homepage: <https://www.ruby-lang.org>

- (a) Log into `lectura.cs.arizona.edu` using your SSH client (or a terminal window on a Mac in one of our computer labs) and, using a text editor, create a file named `fibonacci.rb` ('rb' for Ruby) with this content, updating the documentation as appropriate:

```
1  #!/usr/bin/ruby
2
3  #####
4  #   Assignment:  Homework #1:  Ruby Exercise
5  #       Author:  Your Name (Your E-mail Address)
6  #       Grader:  Patrick Hickey / Andrea Padula
7  #
8  #       Course:  CSc 372
9  #   Instructor:  L. McCann
10 #       Due Date: January 20, 2017
11 #
12 # Description:  A simple type-in exercise to ensure that students
13 #               are able to successfully use ruby on lectura.
14 #
15 #   Language:   Ruby
16 # Ex. Packages: None.
17 #
18 # Deficiencies: None.
19 #####
20
21 class DemoRuby
22
```

```

23     def fibonacci_iterative (n)
24         if n == 0 || n == 1
25             n
26         else
27             older = 0
28             old = 1
29             for i in 2..n
30                 current = older+old
31                 older = old
32                 old = current
33             end
34             current
35         end
36     end
37
38 end
39
40 newObject = DemoRuby.new
41 puts "The first 10 Fibonacci numbers are:"
42 for i in 0..9
43     puts newObject.fibonacci_iterative(i)
44 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 do the upcoming Ruby assignment 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 `lectura.cs.arizona.edu` using your SSH client (or a terminal window on a Mac in one of our computer labs) and, using a text editor, create a file named `fibonacci.hs` (‘hs’ for Haskell) with this content, updating the documentation appropriately:

```

1  -----
2  -- Assignment: Homework #1: Haskell Exercise
3  -- Author: Your Name (Your E-mail Address)
4  -- Grader: Patrick Hickey / Andrea Padula
5  --
6  -- Course: CSc 372
7  -- Instructor: L. McCann
8  -- Due Date: January 20, 2017
9  --
10 -- Description: A simple type-in exercise to ensure that students
11 --              are able to successfully use haskell on lectura.
12 --
13 -- Language: Haskell (ghc)
14 -- Ex. Packages: None.
15 --
16 -- Deficiencies: None.
17  -----
18
19 fibStep :: (Integer,Integer) -> (Integer,Integer)
20 fibStep(u,v) = (v,u+v)
21
22 fibPair :: Integer -> (Integer,Integer)
23 fibPair n
24   | n == 0    = (0,1)
25   | otherwise = fibStep (fibPair (n-1))
26
27 fastFib :: Integer -> Integer
28 fastFib = fst . fibPair

```

```

29 main = do
30   putStrLn "The first 10 Fibonacci numbers are:"
31   print ([fastFib(i) | i <- [0..9] ])
32

```

- (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 do the upcoming Haskell assignment 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 your SSH client and, using a text editor, create a file named `connecticut.pl` ('pl' for Prolog) with this content, updating the documentation as appropriate:

```

1  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
2  %   Assignment:  Homework #1:  SWI-Prolog Exercise
3  %   Author:    Your Name (Your E-mail Address)
4  %   TAS:      Patrick Hickey / Andrea Padula
5  %
6  %   Course:    CSc 372
7  %   Instructor: L. McCann
8  %   Due Date:  January 20, 2017
9  %
10 %   Description: A simple type-in exercise to ensure that students
11 %               are able to successfully use SWI-Prolog on lectura.
12 %
13 %   Language:   Prolog (swipl)
14 %   Ex. Packages: None.
15 %
16 %   Deficiencies: None.
17 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
18
19 %%% Facts:  Which Connecticut counties border which others?
20
21 bordering(fairfield,litchfield).
22 bordering(fairfield,newhaven).
23 bordering(litchfield,hartford).
24 bordering(litchfield,newhaven).
25 bordering(newhaven,middlesex).
26 bordering(newhaven,hartford).
27 bordering(hartford,tolland).
28 bordering(hartford,middlesex).
29 bordering(hartford,newlondon).
30 bordering(tolland,windham).
31 bordering(tolland,newlondon).
32 bordering(middlesex,newlondon).
33 bordering(windham,newlondon).
34
35 %%% Rules:
36
37 % adjacent(X,Y) -- Counties X and Y share a border.
38
39 adjacent(X,Y) :- bordering(X,Y).
40 adjacent(X,Y) :- bordering(Y,X).
41
42 % nearby(X,Y) -- Intent:  Counties X and Y are separated by no more than
43 %                   one other county.
44
45 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 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 do the upcoming Prolog assignment 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.