

<http://www2.cs.arizona.edu/classes/cs372/fall126/>

Homework #1

(50 points)

Due Date: September 16th, 2026, at the beginning of class

Directions

1. **This is an INDIVIDUAL assignment; do your own work! Submitting answers created by other people or generated by AIs (e.g., ChatGPT) are a few examples of NOT doing your own work.**
 2. Write complete answers to each of the following questions, in accordance with the given directions. Create your solutions as a PDF document such that each question is on a separate page; all parts of a multi-part question may be on the same page. Show your work, when appropriate, for possible partial credit.
 3. If you have questions about any aspect of this assignment, help is available from the class staff via piazza.com and our office hours.
 4. When your answers are ready to be turned in, do so on gradescope.com. Be sure to assign pages to problems after you upload your PDF. Need help? Visit <https://help.gradescope.com/> and search for “Submitting an Assignment.”
 5. Remember that you can use at most one late day on a homework assignment.
 6. Start early! Getting help is much easier n days before the due date/time than it will be n hours before.
-

Louden/Lambert Chapter 1: Introduction

1. (5 points) As you have (and will continue to) hear me say, advances in programming language design are driven, in large part, by a desire for more abstraction. Languages usually offer less-abstract ways of doing things, too. Using Python or Java, describe a computational task that can be accomplished in the language, and provide a pair of code examples from the language that show a less-abstract way and a more-abstract way of doing that task.
2. (5 points) The concepts of a ‘data structure’ and an ‘abstract data type’ sound alike, but there are some key differences. What are two ways in which these ideas differ?
3. (10 points) You are probably very familiar (whether or not you wish to be!) with syntax errors, thanks to the many programs you have written. How do syntax errors differ from lexical and semantic errors? To support your answer, provide at least two examples of each of those three types of errors (so, six examples total).

Language Introductions

These questions ask you to connect to lectura.cs.arizona.edu using an SSH client program. We have recorded a video that shows how to do this. You can find it on the class site in D2L, in the Content area.

4. (10 points) We will be using Ruby as an example of an object-oriented language (to compare/contrast with Python and Java).

Language 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 G-S 903) and, using a text editor, create a file named `fibonacci.rb` ('rb' for Ruby) with the following 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  #
7  #   Course:     CSc 372
8  #   Instructor: L. McCann
9  #   TAs:        fill in their names here!
10 #   Due Date:   September 16, 2026
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) Execute (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 (Program #1) 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.

5. (10 points) We will be using Haskell as an example of a functional language.

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 G-S 903) and, using a text editor, create a file named `fibonacci.hs` ('hs' for Haskell) with this content, updating the documentation appropriately:

```

1  -- Assignment: Homework #1: Haskell Exercise
2  -- Author: Your Name (Your E-mail Address)
3  --
4  --
5  -- Course: CSc 372
6  -- Instructor: L. McCann
7  -- TAs: fill in their names here!
8  -- Due Date: September 16, 2026
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
30 main = do
31   putStrLn "The first 10 Fibonacci numbers are:"
32   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 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.

6. (10 points) We will be using SWI-Prolog to explore how a logic programming language works.

Language 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:

[illegible]

```

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 on the statements!
- 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.