CSc 372 — Comparative Programming Languages
Spring 2026 (McCann)

http://www2.cs.arizona.edu/classes/cs372/spring26/

Homework #1

(50 points)

Due Date: February 4th, 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) Imperative programming languages are grouped together because they share a small set of common characteristics. What are those characteristics?
2. (5 points) Consider your answer to Question 1 in the context of the Von Neumann architecture. How do the characteristics of imperative languages follow from the design of that architecture?
3. (10 points) Programs that are written in the C++ language are typically sent to a program called a *compiler* that converts the program statements into the machine language of that computer's CPU. Rather than being compiled, Java and Python programs are usually translated to *byte code* and executed on a *virtual machine*. What are **two** advantages and **two** disadvantages of the virtual machine approach, as compared to the compilation approach? For each, briefly explain why it is an advantage or disadvantage.

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 this content, updating the documentation as appropriate:

```
#!/usr/bin/ruby

##########################################################################
#   Assignment:  Homework #1:  Ruby Exercise
#       Author:  Your Name (Your E-mail Address)
#
#       Course:  CSc 372
#   Instructor:  L. McCann
#          TAs:  Muaz Ali, Daniel Reynaldo
#     Due Date:  February 4, 2026
#
#  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) 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 an 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:

```
------------------------------------------------------------------------
--   Assignment:  Homework #1:  Haskell Exercise
--       Author:  Your Name (Your E-mail Address)
--
--       Course:  CSc 372
```

```
--   Instructor:  L. McCann
--          TAs:  Muaz Ali, Daniel Reynaldo
--     Due Date:  February 4, 2026
--
--  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 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:

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   Assignment:  Homework #1:  SWI-Prolog Exercise
%       Author:  Your Name (Your E-mail Address)
%
%       Course:  CSc 372
%   Instructor:  L. McCann
%          TAs:  Muaz Ali, Daniel Reynaldo
%     Due Date:  February 4, 2026
%
%  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 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.