CSc 372 — Comparative Programming Languages
Spring 2017 (McCann)

http://www.cs.arizona.edu/classes/cs372/spring17

Project: Learn a New (to You) Language

Due Dates:

Part 1:	April 12th, 2017, at the beginning of class
Part 2:	May 3rd, 2017, at the beginning of class

Overview: This class exposes you to (hopefully new-to-you) languages from the three major language paradigms that aren't strictly imperative (namely, object-oriented, functional, and logic) to broaden your language horizon, but also to help you to learn the new languages you will encounter in the upcoming decades of your professional lives. This assignment is meant to support that latter goal.

Assignment: This assignment is in two parts. Part 1 is meant to help you get started in a timely fashion. The real work is in Part 2.

Part 1: You have four weeks total to work on this project, but just one week to make two key choices: Your team size and the language you'll study.

1. Team Composition: You may work on this project alone or with one other student. Teams of two will have some additional work to complete (see below). Of course, both partners must agree to work together. Should poor team chemistry or any other factor(s) force the team to be dissolved, both members are to (a) notify the TAs and (b) continue on the project alone.
2. Language Selection: You may choose any language you wish, subject to these conditions:
   (a) The language is NOT: A machine or assembly language, an OS shell (e.g., bash, ksh, ...), Java, C, C++, Ruby, Haskell, Prolog, Python, any language very similar to any of those, and any language you already know in more than trivial detail.
   For example, Processing is not an acceptable choice because it is really just Java. Objective C, on the other hand, is acceptable (but we encourage you to look outside of the general C language family). We can't read minds, so we don't have a good way to know that you already know a language, meaning that the last language restriction (no prior knowledge) is enforced through the honor system.
   (b) The language has a translator (compiler or interpreter) that is accessible to everyone. Note that this doesn't mean it has to be free, just accessible, but we expect you'll opt for free. You're welcome to change to another compiler/interpreter later, but you need to have at least one identified within the first week.

   See the "Want to Learn More?" section of this handout, below, for some resources that may help you select a language.

When you have your team composition and language set, *everyone* is to use the form link in the Hand In section (below) to tell the TAs (Patrick and Andrea) the following info:

1. Your name (and your teammate's name, if you have a partner).
2. The language you plan to study.
3. The URL of the site offering the language translator you've identified.
4. Your reason for selecting that language. (Why? We're curious!)

(Continued...)

Part 2: Your study of your chosen language will result in three products. For teams of two, all parts are required. For teams of one, see the "Working Alone?" section, below the list.

1. Create a web page, located wherever you wish to put it (including having it be just an HTML file with visibility limited to your laptop, but we hope you'll be proud enough of it to share it with the world), containing at least the following sections:

   (a) History – a brief history of the language (who created it, why, its current version, where to get a translator for it, and anything else you think is interesting about it)

   (b) Paradigm – what kind of language is it (OO/Functional/Logic/other), and how you know.

   (c) Typing System – is the language strongly or weakly typed, are type declarations required, can a programmer create new types, and are functions first-class objects?

   (d) Control Structures – how can control flow be controlled? (That is, what are the languages selection and repetition options?)

   (e) Support for Data Abstractions – which abstractions are provided, how can a programmer create new ones?

   (f) Syntax – think about the syntax used by your language. What are the syntax choices that appeal to you? Are there any syntactic choices that you'd like to see changed (why, and to what)?

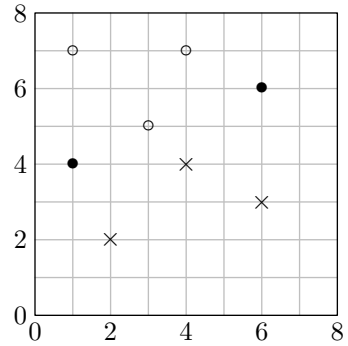
   (g) Semantics – briefly explain how your language is scoped (static or dynamic?), which kinds of constants are supported, how storage is allocated (which combination of static, stack-dynamic, and/or heap-dynamic?), and how garbage is managed.

   (h) Desirable Language Characteristics – In Topic 2, we covered four categories of language characteristics that are generally thought of as 'desirable': (i) Efficiency, (ii) Regularity, (iii) Security/Reliability, and (iv) Extensibility. Choose any three of these four, and discuss features of your language that support (or limit!) them.

2. Because there's no substitute for using a language to learn the language, you are to write a complete, well-documented program that uses two 'training' sets of 2D points and an un-weighted k Nearest Neighbors (kNN) algorithm to classify a third set of 'testing' points.

   kNN is a machine-learning algorithm used to classify each item of the testing data into a group (one of two, in this case, because we have two training sets) based on which group has the majority of the item's k neighbors. For this program, we'll let $k = 3$. Note that the testing data is not being added to either training set, just classified.

   For example, consider the figure below. Let the first set of training data (the ∘'s) consist of the points (1,7), (3,5), and (4,7), and let the second set (the ×) contain the points (2,2), (4,4), and (6,3). Further, assume that the testing data has just the points (6,6) and (1,4). (6,6) is classified as belonging to the second set because two of the three closest points to it ((4,7), (4,4), and (6,3)) are in the second set. (1,4) can't be classified as belonging to either, as the two training points closest to it ((2,2) and (3,5)) are equidistant from it, and there's a tie for the third between (1,7) and (4,4).



(Continued...)

Your output should be a list of the testing data points and the group (first, second, or tie) into which it would be placed based on its $k = 3$ nearest neighbors.

You may assume that each set will hold a maximum of 50 points, with all coordinates being integers, within a grid with axes both labeled 0, 1, 2, ..., 49. How the three sets of points enter your program is up to you — read from a file, hard-coded, whatever. We don't care, so long as you can justify your choice as being reasonable. (Be aware that "we're lazy" isn't likely to be classified as a 'reasonable' justification).

3. A live demo with one of the TAs. Because the TAs can't be expected to install all of the translators and test each program, you'll demo your program for them. They may also have some questions to ask you about your language or your program.

Working Alone? For teams of size one, you do not have to do the "Support for Data Abstractions" and "Syntax" sections, and you need only do one of the "Desirable Language Characteristics" categories. We realize that this isn't a huge reduction in the amount of work, but the "To partner or not to partner?" answer is up to you.

Data: Please see the program description, above.

Output: The program description says that the output is to be a list. If your choice of language supports graphical output and you wish to investigate it by producing a plot of the points instead of the list, please feel free to do so.

Hand In:

- *Part 1:* Everyone is to submit language and team info via this form:

  `https://goo.gl/forms/yrk4ystEDRLc5xWt1`

  (This link is available from the class web page, beneath the link to this handout.)

- *Part 2:* You (or your partner) are required to submit your files (your web page file(s) and your kNN program file(s)) using the `turnin` facility on lectura. The submission folder is `cs372project`. Instructions are available from the document of submission instructions linked to the class web page. Submit all files as-is, *without* packaging them into `.zip`, `.jar`, `.tar`, etc., files.

Want to Learn More?

- There are many ways to determine a language's popularity. Here are two frequently-cited language popularity rankings that might give you some ideas for a language to study:
  - `Tiobe Index` — A monthly list of language popularity
    `https://www.tiobe.com/tiobe-index/`
  - `PYPL Index` — Languages ranked by tutorial searches
    `http://pypl.github.io/PYPL.html`
- kNN has lots of variants. Its Wikipedia page is a good place to start:

  `https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm`

Additional Hints, Reminders, and Requirements:

- We think you'll find that investigating the web page content and writing the kNN program will be mutually supporting activities: Studying characteristics of the language will help you learn features for use by your program, and constructing the program will expose items and oddities for you to add to the web page.
- **Start Early!** You have a week to make your partner and language choices. Give yourself time to make good selections!