## Project Instructions

While completing your research on multiparadigm languages in Module 2, you selected three languages used in industry that interested you.

For this project, you will either write a **research paper** OR **write code**. **Coding**

### Project

- If you choose to code, you will write small code that displays the major difference between two languages in two different paradigms.

  Submission guidelines:

  - The codes should be well commented. Comment effectively so that it is easier to understand.  For each concept, include a description.
  - You can submit the codes in a zip file on blackboard.
    - ReadMe file: Mention the programming language and compilers used.
  - You can also code your program in GitHub. In the blackboard submit URL to your GitHub account.

### Research Paper

- If you choose to write a research paper, you will be comparing three different programming languages along with their usage in different industry. Your paper must be two pages or more, single-spaced. Feel free to select APA/MLA/IEEE styles for citations.

When comparing languages, for either option, you need to compare at least three concepts. Examples include *garbage collection method, implementation models used, types used, and parameter passing methods.* Make sure you do not compare syntax differences.

*Not acceptable*

- *hello world or greatest of two/three numbers.*
- *Program to check if a string is palindrome or not.*
- *Reverse words in a given String.*
- *Program for n-th Fibonacci number*
- *Program for factorial of a number*
- *Program to print all Prime numbers in an Interval.*

*Acceptable*

- *Searching Algorithms:*
    - *Binary Search:*
        - *Implementation: Write a binary search algorithm to search for a specific element in a sorted array.*
        - *Real-time Example: Implement a phone book search, where users can search for a contact by name. The contact list should be sorted by name, and your binary search algorithm should find the contact quickly.*
    - *Linear Search:*
        - *Implementation: Implement a linear search algorithm to find an element in an unsorted array.*
        - *Real-time Example: Create a program that searches for a specific keyword in a large text document.*
    - *Hash Tables:*
        - *Implementation: Create a hash table data structure and implement search operations using open addressing or chaining.*
        - *Real-time Example: Implement a simple dictionary with word definitions, allowing users to search for words and retrieve their meanings quickly.*
- *Sorting Algorithms:*
    - *Heap Sort:*
        - *Implementation: Implement the Heap Sort algorithm to sort an array of integers or strings. You will need to build a max-heap and repeatedly extract the maximum element.*
        - *Real-time Example: Sort a list of student records based on their scores, where each record contains the student's name and exam score. Use Heap Sort to efficiently rank students by their scores.*
    - *QuickSort:*
        - *Implementation: Implement the QuickSort algorithm to sort an array of integers or strings.*
        - *Real-time Example: Sort a list of customer orders by order date in an ecommerce application.*
    - *MergeSort:*
        - *Implementation: Implement the MergeSort algorithm to sort a collection of data.*
        - *Real-time Example: Sort and merge multiple log files from different servers based on timestamps for efficient log analysis.*
    - *BubbleSort –*
        - *Implementation: Though not efficient, BubbleSort is a good teaching tool. Implement BubbleSort on a small dataset.*
        - *Real-time Example: Simulate the sorting of a deck of cards using BubbleSort. Shuffle the cards and sort them using the algorithm.*