

IN353

Advanced Software Development Including Web and Mobility Using JavaScript and PHP

ExcelTrack™ Module Preview

This guide contains a preview of the materials for all course modules in this series:

IN353M1 Programming Data Structures

IN353M2 Recursion, Sorting, and Searching

IN353M3 User Interface Development and Data Validation

IN353M4 Multitier Architecture

IN353M5 Web Services Development

IN353M6 Mobility Development and Cross-Compiling

Be sure to follow the detailed directions found within the actual course modules after you receive access. The content in this guide is only a preview and course content may have changed.

**Course Materials**

**Title**: [*Designing multi-device experiences*](https://libauth.purdueglobal.edu/login?url=https://learning-oreilly-com.libauth.purdueglobal.edu/library/view/designing-multi-device-experiences/9781449340391/?ar&orpq&email=%5eu)

**Edition**: 1st (2014)

**Author**: Levin, M.

**Publisher**: O’Reilly.

**Book ISBN:** 978-1-4493-4038-4

**Title:** [*Learning PHP, MySQL & JavaScript*](https://learning-oreilly-com.libauth.purdueglobal.edu/library/view/learning-php-mysql/9781492093817/)

**Edition:** 6th (2021)

**Author:** Nixon, Robin

**Publisher:** O’Reilly

**Book ISBN:** 978-1-492-09382-4

**Title:** [*Learning JavaScript Data Structures and Algorithms - Second Edition*](https://libauth.purdueglobal.edu/login?url=https://search.ebscohost.com/login.aspx?direct=true&db=nlebk&AN=1285066&site=eds-live&ebv=EB&ppid=pp_Cover)

**Edition:** 2nd (2016)

**Author:** Loiane Groner

**Publisher:** Packt

**Book ISBN:** 978-1-78528-549-3

**Title**: [*Programming PHP, 4th Edition*](https://learning-oreilly-com.libauth.purdueglobal.edu/library/view/programming-php-4th/9781492054122/)

**Edition**: 4th (2020)

**Author**: Kevin Tatroe, Peter MacIntyre

**Publisher**: O’Reilly Media, Inc.

**ISBN**:9781492054122

**Title:** [*Progressive Web Apps*](https://libauth.purdueglobal.edu/login?url=https://learning-oreilly-com.libauth.purdueglobal.edu/library/view/progressive-web-apps/9781617294587/?ar=&email=Jfz56E6E6RsAoCdP76y3FA%3D%3D&tstamp=1678750282&id=187E76C872E6932B76BB864ADA5A6BDBAA580837?ar&orpq&email=%5eu)

**Edition:** 1st (2018)

**Author:** David Hume

**Publisher:** Manning Publications

**Book ISBN:** 9781617294587

| **Course Code** | IN353M1 |
| --- | --- |
| **Competency Title** | Programming Data Structures |
| **Outcome(s)** | **IN353M1-1:** Apply the common linear structures of lists, stacks, and queues |

## Learn: Reading and Research

**Required Reading:**

### *Learning PHP, MySQL & JavaScript:*

### Chapter 3: “Introduction to PHP”

### Chapter 5: “Functions and Objects”

### Chapter 6: “Arrays”

### Chapter 12: “Form Handling”

### Chapter 14: “Exploring JavaScript”

### Chapter 19: “Introduction to CSS”

### *Programming PHP,* 4th edition:

### Chapter 4: “Functions”

*Learning JavaScript Data Structures and Algorithms*,2ndedition:

* Chapter 3: “Stacks”
* Chapter 4: “Queues”

### Required Videos:

* [*Recursion*](https://kapextmediassl-a.akamaihd.net/IST/media/IT391/1502A/Videos/Unit3_General_Recursion/Unit3_General_Recursion.html)
* [*Recursion in Web Development*](https://kapextmediassl-a.akamaihd.net/IST/media/IT391/1502A/Videos/Unit3_Recursion_Web/Unit3_Recursion_Web.html)
* [*Queues — Web*](https://kapextmediassl-a.akamaihd.net/IST/media/IT391/1502A/Videos/Unit2_Queues_Web/Unit2_Queues_Web.html)
* [*Stacks — Web*](https://kapextmediassl-a.akamaihd.net/IST/media/IT391/1502A/Videos/Unit2_Stacks_Web/Unit2_Stacks_Web.html)

## Competency Assessment

This Competency Assessment assesses the following outcome(s)

**IN353M1-1:** Apply the common linear structures of lists, stacks, and queues.

**Assessment 1 Purpose**

The purpose of the assessment in this module is to study and practice implementing recursion to solve problems of a repetitious nature.

**Assessment 1 Requirements**

This assessment has two sections that will be written as two sections of the same program. Please note that all assessments must work as webpages and not as command-line programs. The beginning of each section will be delineated with the following comment block. Be sure to insert the correct values for the section number.

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//\*\*\*\*Assessment 1 #, Section #

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Assessment 1 Instructions**

**Section 1:**

1. Add your section comment to the code.
2. Print a blank line.
3. Print “\*\*\*\*\*\*\*\*\*\*   Section 1   \*\*\*\*\*\*\*\*\*\*”.
4. Print a blank line.
5. Create a function called Fibonacci which will recursively calculate the Fibonacci sequence for the number passed to it.
6. Call the Fibonacci function and pass it the value 10.

**Section 2:**

1. Add your section comment to the code.
2. Print “\*\*\*\*\*\*\*\*\*\*   Section 2   \*\*\*\*\*\*\*\*\*\*”
3. Print a blank line.
4. Create a function called factorial which will recursively calculate the factorial of an integer value passed to it.
5. Create a for loop to use the factorial function to find and print the factorials of every number from 1 to 4.
6. When printing the factorial values, print them showing the number and its factorial. For example, the factorial of 3 would be printed as 4! = 24

**Assessment 2 Purpose**

The purpose of this assessment is to study and practice the implementation of the queue and stack data structures.

**Assessment 2 Requirements**

This assessment has two parts, A and B.

Both parts consist of several sections. Each section will create and manipulate a data structure. You may create one or more code files for this. Please note that all assessments must work as webpages and not as command-line programs.

In both the Part A and Part B programs, the beginning of each section will be delineated with the following comment block. Be sure to insert the correct values for the section number and program.

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//\*\*\*\*Assessment 2 #, Part (A or B), Section #

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Assessment 2 Instructions — Part A**

**Section 1:**

1. Add your section comment to the code.
2. Print “\*\*\*\*\*\*\*\*\*\*  Part A Section 1   \*\*\*\*\*\*\*\*\*\*”
3. Print a blank line.
4. On a new line, request the user to enter a word.
5. Create a stack structure.
6. Create a function called revString() which will accept a string value as input, use a loop to push each character of the string value onto the stack structure, and then pop each off in reverse.
7. On a new line, print “Your word in reverse is:” followed by the reversed word.

*Hint*: If the user entered “part”, the reverse of “part” is “trap”. (Obviously, not all words will spell a real word in reverse.)

**Section 2:**

1. Add your section comment to the code.
2. Print a blank line.
3. Print “\*\*\*\*\*\*\*\*\*\*  Part A Section 2   \*\*\*\*\*\*\*\*\*\*”.
4. Print a blank line.
5. Create a queue structure named ‘shoppers’ and load it with the names “Jane”, “Bob”, “Liza”, “Tom”, “Mary”.
6. On a new line, print “The number of shoppers at the grocery store is” followed by the number of names in the queue.
7. On a new line, print “The first shopper in line is” followed by the name of the first person in the queue.
8. Add “Stephen” and “Ellen” to the queue.
9. Remove three more shoppers from the queue.
10. On a new line, print “The number of shoppers now in line is” followed by the current number of shoppers in the queue.
11. On a new line, print “The shopper currently first in line is” followed by the name of the person first in line in the queue.

**Assessment 2 Instructions — Part B**

Reminder: Part B of this assessment can be a separate program from Part A.

**Section 1:**

1. Add your section comment to the code.
2. Print “\*\*\*\*\*\*\*\*\*\* Part B Section 1 \*\*\*\*\*\*\*\*\*\*”
3. Print a blank line.
4. Create a queue structure and add the following people in this order: “Jim”, “Bob”, “Susan”, “Liz”, and “Alex”.
5. On a new line, print “The number of people in line at the bank is” followed by the number of people in the queue.
6. On a new line, print “The names of those in line at the bank are” followed by the names in the queue in order without removing them from the queue.
7. On a new line, print “The first customer in line is” followed by the name of the first customer in line. Remove the customer from the queue.
8. Add “Andy” and “Rhonda” to the queue.
9. Remove three more people from the queue.
10. On a new line, print “The number of customers in line now is” followed by the number of people in the queue.
11. On a new line, print “<name> is now at the head of the line”. Be sure to use the first name of the queue to replace <name>.

**Section 2:**

1. Add your section comment to the code.
2. Implement a stack structure.
3. Print a blank line.
4. On a new line, print “\*\*\*\*\*\*\*\*\*\*\*   Part B Section: 2   \*\*\*\*\*\*\*\*\*\*\*” to the console.
5. Print a blank line.
6. On a new line, print “Please enter a sentence.”
7. Add each word of the sentence to the stack.
8. Remove each word and print it on a new line so that the words are readable but the sentence is written in reverse.

**After you receive access to each module, you will be able to see the submission requirements and a Checklist Rubric for the Competency Assessment.**

| **Course Code** | IN353M2 |
| --- | --- |
| **Competency Title** | Recursion, Sorting, and Searching |
| **Outcome(s)** | **IN353M2-2:** Develop procedures to solve data structures and algorithm problems.  |

## Learn: Reading and Research

**Required Reading:**

*Learning JavaScript Data Structures and Algorithms,* 2nd edition:

* Chapter 5: “Linked List”
* Chapter 6: “Sets”
* Chapter 8: “Trees”
* Chapter 9: “Graphs”
* Chapter 10: “Searching and Sorting Algorithms”

## Learn: Recorded Instructional Video

**Required Videos:**

* [*Data Structures*](https://kapextmediassl-a.akamaihd.net/IST/media/IT391/1502A/Videos/Unit1_Data_Structures/IT391_Unit1_Data_Structures.html)
* [*Linked Lists — Web*](https://kapextmediassl-a.akamaihd.net/IST/media/IT391/1502A/Videos/Unit1_Linked_Lists_Web/Unit1_Linked_Lists_Web.html)
* [*Trees — Web*](https://kapextmediassl-a.akamaihd.net/IST/media/IT391/1502A/Videos/Unit1_Trees_Web/Unit1_Trees_Web.html)
* [*Big O*](https://kapextmediassl-a.akamaihd.net/IST/media/IT391/1502A/Videos/Unit4_BigO/Unit4_BigO.html)
* [*Searching — Web*](https://kapextmediassl-a.akamaihd.net/IST/media/IT391/1502A/Videos/Unit4_Searching_Web/Unit4_Searching_Web.html)
* [*Sorting — Web*](https://kapextmediassl-a.akamaihd.net/IST/media/IT391/1502A/Videos/Unit4_Sorting_Web/Unit4_Sorting_Web.html)

## Competency Assessment

This Competency Assessment assesses the following outcome(s):

**IN353M2-2:** Develop procedures to solve data structures and algorithm problems.

**Assessment 1 Purpose**

The purpose of this assessment is to provide you with an opportunity to demonstrate knowledge of creating and manipulating various container structures.

**Assessment 1 Requirements**

This assessment has two parts, A and B. Both parts consist of several sections. Each section will create and manipulate a data structure. You can complete a single program for both parts, or a separate program for Part A and Part B.

In both the Part A and Part B programs, the beginning of each section will be delineated with the following comment block. Be sure to insert the correct values for the section number and program.

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//\*\*\*\*Assessment 1 #, Part (A or B), Section #

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Please note that all assessments must work as webpages and not as command-line programs.

**Assessment 1 Instructions — Part A**

**Section 1:**

1. Add your section comment to the code.
2. Create a string list and fill it with the profession names shown here:
	* Software Engineer
	* Programmer
	* Database Admin
	* Network Admin
	* Web Developer
	* Program Manager
3. Create a set and load it from the list.
4. Create a sorted set and load it from the first set.
5. On a new line print “\*\*\*\*\*\*\*\*\*\*\*   Section: 1   \*\*\*\*\*\*\*\*\*\*\*”
6. On a new line, print “Original List:” to the console.
7. On a new line, print the contents of the unsorted set.
8. On a new line, print “Sorted List:” to the console.
9. On a new line, print the contents of the sorted list.

**Section 2:**

1. Add your section comment to the code.
2. Create a linked list structure named ‘books’ and load it with the titles shown:
	* To Kill a Mockingbird
	* Huckleberry Finn
	* Pride and Prejudice
	* Brave New World
	* Lord of the Flies
	* Alice in Wonderland
	* The Old Man and the Sea
	* Atlas Shrugged
3. Print a blank line.
4. On a new line print “\*\*\*\*\*\*\*\*\*\*\*   Section: 2   \*\*\*\*\*\*\*\*\*\*\*”
5. On a new line, print “Original Book List:”
6. On a new line, print the contents of the linked list.
7. Sort the linked list alphabetically.
8. On a new line, print “Sorted Book List:”
9. On a new line, print the contents of the linked list.
10. Remove the second item from the list.
11. Remove the first item from the list.
12. Remove the last item from the list.
13. On a new line, print “Book List After Deletions:”
14. On a new line, print the contents of the linked list.
15. On a new line, print “The number of items in my book list is:” followed by the total number of items in the list.
16. Determine if the title “Brave New World” can be found in the list and print on a new line whether it exists in the current list.

**Section 3:**

1. Add your section comment to the code.
2. Create a class called Node which can represent a node in a binary tree.
3. Create a BinaryTree class.
4. Add an insert function to the BinaryTree class to insert new values.
5. Add a traverse function to the BinaryTree class to traverse the tree in order and print each node’s value.
6. Define a new instance of BinaryTree called myBinaryTree.
7. Use the insert function to add the following values: 50, 30, 45,12, 29.
8. Print a blank line.
9. On a new line print “\*\*\*\*\*\*\*\*\*\*\*   Section 3   \*\*\*\*\*\*\*\*\*\*\*”
10. On a new line, print to the console “The contents of the binary tree are:”
11. Use the traverse function to print the contents of the binary tree.

**Assessment Instructions — Part B**

Reminder: Part B of this assessment should be a separate program from Part A.

**Section 1:**

1. Add your section comment to the code.
2. Utilize a try/catch block where needed.
3. Define a list called mammals that contains 6 strings.
4. Load the list with the 6 strings below. Precision is important in software development, so be sure to use the correct spelling and case for each string.
	* Bear
	* Gorilla
	* Tiger
	* Polar Bear
	* Lion
	* Monkey
5. Implement a set called setMammals and populate the set from the list called mammals.
6. Print “\*\*\*\*\*\*\*\*\*\*\*   Part B Section: 1   \*\*\*\*\*\*\*\*\*\*\*” to the console.
7. On a new line, print “Contents of the set are:”
8. On a new line, print a list of every element in the set.
9. Create a new set called sortedMammals derived from the setMammals set.
10. On a new line, print “Contents of the sorted set are:”
11. On a new line, print a list of every element in the sorted set.
12. On a new line, print “The first item in the set is:”  followed by the first item in the sorted set.
13. On a new line, print “The last item in the set is:”  followed by the last item in the sorted set.

**Section 2:**

1. Add your section comment to the code.
2. Utilize a try/catch block where needed.
3. Implement a new linked list called myFriends.
4. Add the following friends’ names and phone numbers to myFriends.
	* Fred 602-299-3300
	* Ann 602-555-4949
	* Grace 520-544-9898
	* Sam 602-343-8723
	* Dorothy 520-689-9745
	* Susan 520-981-8745
	* Bill 520-456-9823
	* Mary 520-788-3457
5. Print a blank line.
6. On a new line, print “\*\*\*\*\*\*\*\*\*\*\*  Part B Section: 2   \*\*\*\*\*\*\*\*\*\*\*” to the console.
7. Print a blank line.
8. On a new line, print “The contents of my friends list:”
9. Print the list of items from the linked list with each friend on a new line.
10. Remove Bill from the list.
11. Remove the first and last elements from the list.
12. Add code that changes Mary’s phone number to 520-897-4567.
13. Print a blank line.
14. On a new line, print “The updated contents of my friends list:”
15. Print the list of items from the linked list with each friend on a new line.
16. On a new line, print “The number of friends in my list is:” followed by the number of items in the list.
17. Add code to check if Fred is still in the list, and on a new line, print a statement as to whether or not Fred is still present in the list.

*Hint:* In PHP, use SplDoublyLinkedList class and functions push(), pop(), offsetUnset(), offsetSet(), count().

**Section 3:**

Note: In Part A, Section 3, you were provided code to create a binary tree. Copy that code into this section of Part B and then add the following.

1. Add your section comment to the code, at the beginning of this section of code.
2. Utilize a try/catch block where needed.
3. Write a function called printInOrder() which traverses the binary tree in order, printing the values along the way.
4. Write a function called printPreOrder() which traverses the binary tree in order, printing the values along the way.
5. Write a function called printPostOrder() which traverses the binary tree in order, printing the values along the way.
6. Write a function called traverse() which will do the following.
	1. Print a blank line.
	2. On a new line print, “\*\*\*\*\*\*\*\*\*\*  Part B Section 3   \*\*\*\*\*\*\*\*\*\*”
	3. Print a blank line.
	4. On a new line, print “Traversing the binary tree in order:”
	5. Call the printInOrder() function, passing it the root node.
	6. On a new line print, “Traversing the binary tree in pre-order:”
	7. Call the printPreOrder() function, passing it the root node.
	8. On a new line, print “Traversing the binary tree in post-order:”
	9. Call the printPostOrder() function, passing it the root node.

*Hint:* There are always three steps to a traversal of a binary tree.

1. Visit the current node.
2. Traverse its left subtree.
3. Traverse its right subtree.

The order in which you perform these three steps results in the different traversal orders:

* Pre-order traversal: (1) (2) (3)
* In-order traversal: (2) (1) (3)
* Post-order traversal: (2) (3) (1)

**Assessment 2 Purpose**

The purpose of this assessment is to study and practice implementing sorting and searching algorithms to quickly locate or organize lists of data.

**Assessment 2 Requirements**

This assessment has two parts, A and B. Both parts consist of several sections. Each section will sort and/or search a list of data. You may complete a single program or two programs, one for Part A and one for Part B.

In both the Part A and Part B programs, the beginning of each section will be delineated with the following comment block. Be sure to insert the correct values for the section number and program.

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//\*\*\*\*Assessment 2 #, Part (A or B), Section #

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Please note that all assessments must work as webpages and not as command-line programs.

**Assessment 2 Instructions — Part A**

**Section 1:**

1. Add your section comment to the code.
2. Print “\*\*\*\*\*\*\*\*\*\*   Part A Section 1 – Quick Sort   \*\*\*\*\*\*\*\*\*\*”
3. Print a blank line.
4. Create a list of the following integers: 6501, 9503, 7557, 5535, 5601, 9001, 9888, 8801, 9767, 7815.
5. Create a function called sortAsc() which will perform a quick sort on the list to sort the data into ascending order.
6. On a new line, print “The list unsorted is” followed by printing the original unsorted list’s contents.
7. Call the function sortAsc(), passing it the necessary data to sort the list into ascending order.
8. Print a blank line.
9. On a new line, print “The list sorted is” followed by printing the sorted list’s contents.

**Section 2:**

1. Add your section comment to the code.
2. Print a blank line.
3. Print “\*\*\*\*\*\*\*\*\*\*   Part A Section 2 – Bubble Sort   \*\*\*\*\*\*\*\*\*\*”.
4. Print a blank line.
5. Create a list with the following integers: 5, 90, 35, 45, 150, 3.
6. Create a function called bubbleSort() which will perform a bubble sort on the list and sort the data into ascending order.
7. On a new line, print “The list unsorted is” followed by printing the original unsorted list’s contents.
8. Call the function bubbleSort(), passing it the necessary data to sort the list into ascending order.
9. Print a blank line.
10. On a new line, print “The list sorted is” followed by printing the sorted list’s contents.

**Section 3:**

1. Add your section comment to the code.
2. Print a blank line.
3. Print “\*\*\*\*\*\*\*\*\*\*   Part A Section 3 – Binary Search   \*\*\*\*\*\*\*\*\*\*”.
4. Print a blank line.
5. Create a function called binarySearch() which will accept a presorted integer list and a value to find in the list. The function should return a string stating at which index the value was found in the list or a message that the value is not located in the list.
6. Call the binarySearch() function, passing it the sorted list from Section 1 and the integer 8801 as the search value.
7. Print a blank line.
8. On a new line, print the message returned by the binarySearch() function.
9. Call the binarySearch() function a second time, passing it the sorted list from Section 1 and the integer 7777 as the search value.
10. Print a blank line.
11. On a new line, print the message returned by the binarySearch() function.

**Assessment 2 Instructions — Part B**

Reminder: Part B of this assessment should be a separate program from Part A.

**Section 1:**

1. Add your section comment to the code.
2. Print “\*\*\*\*\*\*\*\*\*\* Part B Section 1 – Bubble Sort   \*\*\*\*\*\*\*\*\*\*”
3. Print a blank line.
4. Create a list called studentGrades and populate the list with the following grades: 65, 95, 75, 55, 56, 90, 98, 88, 97, and 78.
5. Create a function called sortListDescBS() which implements a bubble sort algorithm that will sort the grade list from highest to lowest.
6. Create a function called sortListAscBS() which implements a bubble sort algorithm that will sort the grade list from lowest to highest.
7. Create a function called printList() which will print the contents of an integer list passed to it.
8. On a new line, print “The unsorted list of grades is” followed by the contents of the unsorted list, using the printList() function.
9. Print a blank line.
10. Call the sortListDescBS() function.
11. On a new line, print “The grades in descending order are” followed by the newly sorted list, using the printList() function.
12. Call the sortListAscBS() function.
13. On a new line, print “The grades in ascending order are” followed by the newly sorted list, using the printList() function.

**Section 2:**

1. Add your section comment to the code.
2. Print a blank line.
3. On a new line, print “\*\*\*\*\*\*\*\*\*\*\*   Part B Section: 2 – Quick Sort   \*\*\*\*\*\*\*\*\*\*\*” to the console.
4. Print a blank line.
5. Create a function called sortListDescQS() which implements a quick sort algorithm, using recursion, that will sort the grade list from highest to lowest.
6. Create a function called sortListAscQS() which implements a quick sort algorithm, using recursion, that will sort the grade list from lowest to highest.
7. For the following steps, use the original unsorted grades list from Section 1.
8. On a new line, print “The unsorted list of grades is” followed by the contents of the unsorted list, using the printList() function.
9. Print a blank line.
10. Call the sortListDescQS() function.
11. On a new line, print “The grades in descending order are” followed by the newly sorted list, using the printList() function.
12. Call the sortListAscQS() function.
13. On a new line, print “The grades in ascending order are,” followed by the newly sorted list, using the printList() function.

**Section 3:**

1. Add your section comment to the code.
2. Print a blank line.
3. On a new line, print “\*\*\*\*\*\*\*\*\*\*\*   Part B Section: 3 – Sequential Search   \*\*\*\*\*\*\*\*\*\*\*” to the console.
4. Print a blank line.
5. Write a function named seqSearch() which accepts an integer list and a value to find in the list. Have the seqSearch() function return a string message stating the index in the list where the value sought was located or a message stating the value was not located in the list.
6. Call the seqSearch() function, passing it a copy of the grades list sorted in ascending order and the value 75.
7. On a new line, print “The contents of the grade list are” followed by the sorted grade list used.
8. Print a blank line.
9. On a new line, print the message returned from the search function.
10. Call the seqSearch () function a second time, passing it the same copy of the sorted grades list along with the value 60.
11. Print a blank line.
12. On a new line, print the message returned from the search function.

**Section 4:**

1. Add your section comment to the code.
2. Print a blank line.
3. On a new line, print “\*\*\*\*\*\*\*\*\*\*\*   Part B Section: 4 – Binary Search   \*\*\*\*\*\*\*\*\*\*\*” to the console.
4. Print a blank line.
5. Write a function named binarySearch() which accepts an integer list and a value to find in the list. Have the binarySearch() function return a string message stating the index in the list where the value sought was located or a message stating the value was not located in the list.
6. Call the binarySearch() function, passing it the same copy of the sorted grades list from Section 3 and the value 56.
7. On a new line, print “The contents of the grade list are” followed by the sorted grade list used.
8. Print a blank line.
9. On a new line, print the message returned from the search function.
10. Call the seqSearch () function a second time, passing it the same copy of the sorted grades list along with the value 50.
11. Print a blank line.
12. On a new line, print the message returned from the search function.

**After you receive access to each module, you will be able to see the submission requirements and a Checklist Rubric for the Competency Assessment.**

| **Course Code** | IN353M3 |
| --- | --- |
| **Competency Title** | User Interface Development and Data Validation |
| **Outcome(s)** | **IN353M3-3:** Analyze best practices for interactive user interface design.**GEL-1.02:** Demonstrate college-level communication through the composition of original materials in Standard English.**GEL-2.02:** Interpret data as they apply to a discipline specific situation. |

## Learn: Reading and Research

**Required Reading:**

*Designing Multi-Device Experiences*:

* Chapter 1: “An Ecosystem of Connected Devices”
* Chapter 2: “The Consistent Design Approach”
* Chapter 3: “The Continuous Design Approach”
* Chapter 4: “The Complementary Approach”
* Chapter 5: “Integrated Design Approaches”
* Chapter 6: “Beyond Core Devices”
* Appendix A: “Companies, Products, and Links”

### *Programming PHP,* 4th edition:

### Chapter 8: “Web Techniques”

### Chapter 14: “Security”

### *Learning PHP, MySQL & JavaScript*:

### Chapter 17: “JavaScript and PHP Validation and Error Handling”

## Learn: Recorded Instructional Video

**Required Videos:**

* [*Complementary Design*](https://kapextmediassl-a.akamaihd.net/IST/media/IT391/1502A/Videos/IT391_Unit5_Complementary_Design/IT391_Unit5_Complementary_Design.html)
* [*Consistent Design*](https://kapextmediassl-a.akamaihd.net/IST/media/IT391/1502A/Videos/IT391_Unit5_Consistent_Design/IT391_Unit5_Consistent_Design.html)
* [*Continuous Design*](https://kapextmediassl-a.akamaihd.net/IST/media/IT391/1502A/Videos/IT391_Unit5_Continuous_Design/IT391_Unit5_Continuous_Design.html)
* [*Integrated Design*](https://kapextmediassl-a.akamaihd.net/IST/media/IT391/1502A/Videos/IT391_Unit5_Integrated_Design/IT391_Unit5_Integrated_Design.html)
* [*Wireframe*](https://kapextmediassl-a.akamaihd.net/IST/media/IT391/1502A/Videos/IT391_Unit5_Wireframes/IT391_Unit5_Wireframes.html)

### [*Graphical User Interface — Web Development*](https://kapextmediassl-a.akamaihd.net/IST/media/IT391/1502A/Videos/Unit6_GUI_Web/Unit6_GUI_Web.html)

## Competency Assessment

This Competency Assessment assesses the following outcome(s):

**IN353M3-3:** Analyze best practices for interactive user interface design.

**GEL-1.02:** Demonstrate college-level communication through the composition of original materials in Standard English.

**GEL-2.02:** Interpret data as they apply to a discipline specific situation.

**Assessment 1 Purpose**

The purpose of this assessment is to design a basic interface that implements graphical user interface elements and takes in some data from a user that can then be accepted by a program and reported back through a form to an end user. The use of wireframe concepts will provide a framework for capturing the information and displaying the captured information to the end user in a proper format based on the requirements of the end user.

**Assessment 1 Requirements**

This assessment will consist of two parts (Part A and Part B).

**Part A**

Develop an idea for an interactive user interface for an application or website of your choice. You will write a 2- to 3-page paper that discusses your approach to creating your interactive user interface design. You will want to discuss the interface design principles that you applied, and you will also want to discuss how you considered consistent, continuous, and complementary approaches as you considered your user interface on multiple devices.

**Part B**

Create a wireframe for your interactive user interface. You can create this in Microsoft® Visio® or PowerPoint®.

**Assessment 1 Instructions — Part A**

1. Describe the software application or website for which you are designing the user interface.
2. Discuss what input is required from the end user.
3. Discuss how that input will be processed in the application.
4. Discuss what output needs to be displayed back to the end user.
5. Discuss how you will apply the core user interface design principles.
6. Discuss how porting this interface to other devices might influence usability.
7. Discuss how you intend to consider and apply the concepts of consistency, continuity, and complementarity about this user interface on multiple devices.

Your sources and content should follow proper APA citation style. For more information on APA style formatting, go to Academic Writer under the Academic Tools area of this course.

**Assessment 1 Instructions — Part B**

Be sure you provide all the detail necessary to illustrate all elements that will appear on the interactive user interface.

1. Navigation bar/links
2. User input elements
3. Submit buttons
4. Area for display of feedback to end user

**Assessment 2 Purpose**

The purpose of this assessment is to develop and implement a user interface that accepts user input.

This assessment will consist of two parts (Part A and Part B).

**Assessment 2 Instructions — Part A**

Consider the data that you will be asking the user to input and write a 1- to 2-page data input validation plan that includes the following:

* A discussion of the steps you will take in your interface design to ensure that data entry is correct in terms of content and format.
* A review of the data types for each field of data entry in the user interface that you developed.
* An explanation of the rationale for choosing each data type.

Your sources and content should follow proper APA citation style. For more information on APA style formatting, go to Academic Writer under the Academic Tools area of this course.

**Assessment 2 Instructions — Part B**

Please note that all assessments must work as webpages and not as command-line programs.

Implement a user interface that allows the user to enter data. You can create the interface based on your previously created wireframe, or you can create a new interface. Validate the data according to your data validation plan.

Test that your data validation plan is implemented and working correctly. Record the results of your test.

For the test, enter both correct and incorrect data and both correct and incorrect formats to demonstrate that validation is working. Your test report should list each input provided by the user and the result. The results should include screenshots of the user interface meeting the assessment standards.

Input Age (between 5 and 18):
User enters 19

Result: Age is out of range

Input Email: User enters happy.purdueglobal.edu
Result: email requires an @ symbol

**After you receive access to each module, you will be able to see the submission requirements and a Checklist Rubric for the Competency Assessment.**

| **Course Code** | IN353M4 |
| --- | --- |
| **Competency Title** | Multitier Architecture |
| **Outcome(s)** | **IN353M4-4:** Analyze client/server relationship. |

## Learn: Reading and Research

**Required Reading:**

### *Programming PHP,* 4th edition:

### Chapter 12: “XML”

## Learn: Learning Resources

**Web Resources:**

From the Internet, access the PHP manual, search on Simple XML, and read about parsing XML with PHP.

From the Internet, search for Simple XML parser and read about parsing XML with PHP.

From the Internet, search for XML Tutorial and read about XML JavaScript.

## Competency Assessment

This Competency Assessment assesses the following outcome(s):

**IN353M4-4:** Analyze client/server relationship.

**Purpose**

The purpose of this assessment is to design and implement a multi-tier application. The data will be stored in an XML file.

**Assessment Requirements**

You will first create an XML file using a text editor of your choice. Specifically, you will create an XML file that holds 24 final student grades. The application you create will read and parse the grades from the XML file (stored on the server). The application will then calculate the average grade, the lowest grade, and the highest grade and return this information to a browser page, using a form to display the values.

It is important to design the application before coding (implementation) begins.

Create a design document that shows the approach you will take in creating the application, including the description of each tier.

When you have completed the assessment, document the steps you took, and discuss any issues you faced and how you addressed each issue.

**Assessment Instructions**

Please note that all assessments must work as webpages and not as command-line programs.

1. Create an XML file that stores 24 student grades.
2. Store the XML file on your server.
3. Name the file StudentGrades.xml.

Write a multi-tier program that performs the following actions:

1. Read and parse the grades from the StudentGrades.xml.
2. Determine the lowest grade, the highest grade, and the average grade. Display these values to the screen in graphical user interface form, not console output.
3. Provide documentation of steps taken, issues faced, and how issues were resolved. (You can add this to the end of the design document.)

**After you receive access to each module, you will be able to see the submission requirements and a Checklist Rubric for the Competency Assessment.**

| **Course Code** | IN353M5 |
| --- | --- |
| **Competency Title** | Web Services Development |
| **Outcome(s)** | **IN353M5-5:** Evaluate web services. |

## Learn: Reading and Research

**Required Reading:**

### *Programming PHP,* 4th edition:

### Chapter 16: “Web Services”

## Learn: Learning Resources

## Suggested Additional Resources:

Google. (n.d.). [*Google Maps Platform Documentation*](https://developers.google.com/maps/documentation)*.* https://developers.google.com/maps/documentation

Google. (n.d.) [*Geocoding API*](https://developers.google.com/maps/documentation/geocoding/overview). https://developers.google.com/maps/documentation/geocoding/overview

## Competency Assessment

This Competency Assessment assesses the following outcome(s):

**IN353M5-5:** Evaluate web services.

**Purpose**

The purpose of this assessment is to utilize a third-party web service in an original program.

**Assessment Requirements**

For this assessment, you will use two web service application programming interfaces (APIs): Google Geocoding™ and Google Maps™. You will provide a form for the user to enter an address, city, state, and ZIP code. This information will be passed to the Google Geocoding API, which will convert the address to the latitude and longitude coordinates. You will want to study the Google Geocoding API to understand the data format requirements and think about how you will have to massage the form data to meet the API requirements. The latitude and longitude coordinates will then be passed to the Google Maps API, and you will display the map for the location in three different formats (e.g., road view, satellite, and hybrid).

Keep in mind the multi-tier architecture and try to use this approach of keeping GUI and business logic separate. Be creative in how you control the display of the three views. You might display all of them at once, use radio button controls to select between the three views, or any other options you want to use in your design.

**Assessment Instructions**

1. Create a form for entry of address of your choice (including address, city, state, ZIP).
2. The address data is passed to the Google Geocoding API to convert the address to latitude and longitude.
3. The latitude and longitude are then passed to Google Maps API, and the location is mapped in three different map formats.

**After you receive access to each module, you will be able to see the submission requirements and a Checklist Rubric for the Competency Assessment.**

| **Course Code** | IN353M6 |
| --- | --- |
| **Competency Title** | Mobility Development and Cross-Compiling |
| **Outcome(s)** | **IN353M6-6:** Design interactive web or mobile applications. |

## Learn: Reading and Research

**Required Reading:**

### *Learning PHP, MySQL & JavaScript:*

### Chapter 13: “Cookies, Sessions, and Authentication”

*Progressive Web Apps:*

* *Chapter 1: “Understanding Progressive Web Apps”*
* *Chapter 2: “First Steps to Building a Progressive Web App”*
* *Chapter 3: “Caching”*
* *Chapter 5: “Look and Feel”*
* *Chapter 7: “Offline Browsing”*

## Learn: Learning Resources

## Suggested Additional Resources:

You may also want to search for simple progressive web applications using PHP and JavaScript, as well as how to use ngrok, and use all those resources to assist in creating the progressive web application (PWA).

## Competency Assessment

This Competency Assessment assesses the following outcome(s):

**IN353M6-6:** Design interactive web or mobile applications.

**Assessment 1 Purpose**

The purpose of this assessment is to create a web application to read user input values and create and read cookies using both PHP and JavaScript.

**Assessment 1 Instructions**

**Step 1:** Create a web application that uses a form to take in five user values: username, password, city, state, and ZIP code.

Use some method to encrypt the password before creating the cookie for the user provided password.

Case 1: Use JavaScript to create cookies.

Case 2: Use PHP to create cookies.

NOTE: Both Case 1 and 2 can be in separate webpages or in the same web program.

You may read the cookies using any method you desire.

Print the cookies to the same webpage. You should not open a new webpage.

**Step 2:** Document the process.

After you have completed the programming part, write a 1-page paper in which you discuss your experience and any issues you encountered. If you did experience issues, please discuss how the issues were resolved. At the bottom of the paper, **include screenshots** that show evidence of your completed web application.

**Assessment 2 Purpose**

The purpose of this assessment is to allow you to demonstrate how to use VS Code editor, PHP, and JavaScript to build a PWA mobile application.

**Assessment 2 Instructions**

Please note that all assessments must work as webpages and not as command-line programs.

Create a progressive web application (PWA) of your choice. You will use the django-pwa package to create the PWA. Also, you will install and use [ngrok](https://ngrok.com/) to make the PWA work on a mobile device.

The PWA mobile app should contain at least one form. The form should contain at least one control/element. The user should be able to interact with the app in some simple way that causes a noticeable change in the display.

Take three screenshots for submission. One should show the state of the app’s display before interaction from the user, and a second should show the display after the user has invoked a change in the display. Take the third screenshot from the mobile device, showing the PWA working on it. (*Hint:* The change could be anything such as a button that changes color or text when clicked or a message displayed in a text box when a button is clicked, etc.).

**After you receive access to each module, you will be able to see the submission requirements and a Checklist Rubric for the Competency Assessment.**