PG2 –SORTING, SEARCHING

CONTENTS

OBJECTIVES

Learn to load and parse CSV files. Implement various sorting and search algorithms. Utilize recursion to solve problems. Clone arrays into new lists.

You will be loading in a CSV file that contains unsorted data and you can store that information in a List (each item is a string). The user will be able to sort that data using the different sorting algorithms that we covered in the lecture: Bubble, Merge. Search a sorted list with a binary search algorithm.

Topics Covered

Cloning, bubble sort, merge sort, binary search, CSV, Split, File loading

Project Setup

A **C# .NET Core console application** has been provided for you in your **GitHub repo**. **Use the provided solution.**

NOTE: use the Input class from Lab 1.

PART 1

The Menu

Show a menu to the user so they can select one of the algorithms (bubble, merge and binary search), save a sorted list, and Exit. (Use the ReadChoice method you created in the first lab) After calling any of the sorting methods, you should **display** the unsorted list along with the sorted list.

1. Bubble Sort
2. Merge Sort
3. Binary Search
4. Save
5. Exit

EXAMPLE OUTPUT:

```
Bubble Sort
--------------------------------------------------------------------------------
Detective Comics                                    Anarky
Batman                                              Arkham Manor
World's Finest Comics                               Azrael
Star-Spangled Comics                                Azrael volume 2
The Brave and the Bold                              Azrael: Agent of the Bat
The Joker                                           Batgirl
Batman Family                                       Batgirl volume 3
Man-Bat                                             Batgirl volume 4
Batman and the Outsiders                            Batman
The Outsiders                                       Batman '66
The Adventures of the Outsiders                     Batman 80-Page Giant
```

The items on the left are the unsorted values and the items on the right are the sorted.

GRADING: 20 POINTS

COMMON MISTAKES:

-1: you should print the sorted results side-by-side with the unsorted

-3: the Exit option does not exit

PART 2

Load the file

Write a method to read the file and return a list of strings. Open and read the line from the **inputFile.csv** file. The line in the file contains a list of comic book titles separated by commas. Split the string and store each title in a **List of strings**.

GRADING: 10 POINTS

COMMON MISTAKES:

-2: you have a full or relative path to the file that is specific to your machine

-2: you are not closing the file after you read it

-2: you are not parsing the data correctly

-1: you are not cloning the original correctly. Setting List<string> list2 = list1; only points list2 to the same thing that list1 points to.

-1: not converting the array to a list.

-2: did not create a method for reading the file

PART 3

Bubble Sort

Write a method to implement the Bubble sort algorithm. You want to keep the original list unsorted so make sure to clone the original list each time you call the Bubble sort. **Your code must follow the pseudocode**.

Turn this Wikipedia pseudocode into C#:

```
procedure bubbleSort(A : list of sortable items)
    n := length(A)
```

```
    repeat
        swapped := false
        for i := 1 to n - 1 inclusive do
            if A[i - 1] > A[i] then
                swap(A, i - 1, i)
                swapped = true
            end if
        end for
        n := n - 1
    until not swapped
end procedure
```

NOTE: **swap** is a method that is not provided by C#. You can create your own method or you can insert the swap logic inside the if. See the lectures slides for how to swap 2 items in a list.

GRADING: 20 POINTS

COMMON MISTAKES:

-5: Bubble sort can be more efficient. The inner for loop should track whether a swap happens. If the inner loop does not swap, then you can break out of the outer loop.

-5: the bubble sort can be optimized more according to the pseudo-code. You can shorten the for loop by 1 after the for loop completes. Store the length of the list in a variable and subtract 1 from it after the for loop. This would mean 1 fewer item to compare each time you run the for loop.

-1: in BubbleSort, you should set swapped = false right before the for loop.

-1: the while condition in bubble sort is incorrect. You need to loop while a swap has happened.

-2: did not create a method for Bubble sort

PART 4

Merge Sort

Write a method to implement the Merge sort algorithm. You want to keep the original list unsorted so make sure to clone the original list each time you call the Merge sort. **Your code must follow the pseudocode**.

Turn this Wikipedia pseudocode into C#:

```
function merge_sort(list m) is
    // Base case. A list of zero or one elements is sorted, by definition.
    if length of m ≤ 1 then
        return m
```

```
    // Recursive case. First, divide the list into equal-sized sublists
    // consisting of the first half and second half of the list.
    // This assumes lists start at index 0.
    var left := empty list
    var right := empty list
    for each x with index i in m do
        if i < (length of m)/2 then
            add x to left
        else
            add x to right

    // Recursively sort both sublists.
    left := merge_sort(left)
    right := merge_sort(right)

    // Then merge the now-sorted sublists.
    return merge(left, right)
```

```
function merge(left, right) is
    var result := empty list

    while left is not empty and right is not empty do
    {
        if first(left) ≤ first(right) then
            add first(left) to result
            left := rest(left) //remove the first item
        else
            add first(right) to result
            right := rest(right) //remove the first item
    }

    // Either left or right may have elements left; consume them.
    // (Only one of the following loops will actually be entered.)
    while left is not empty do
    {
        add first(left) to result
        left := rest(left) //remove the first item
    }
    while right is not empty do
    {
        add first(right) to result
        right := rest(right) //remove the first item
    }
    return result
```

GRADING: 20 POINTS

COMMON MISTAKES:

-2: the exit condition needs to be if the count of the list <= 1.

-2: did not write a method for Merge Sort

PART 5

Binary Search

Write a method to implement the Binary Search algorithm (use a recursive approach). **Your code must follow the pseudocode.**

1. Clone the original list and sort the cloned list (call Sort on the list).
2. Loop over the sorted list.
3. Call ***your*** binary search method to search the sorted list for each title in the sorted list.
   HINT: the index returned from your binary search should match the index.

Show the search title, the index and the index returned by your binary search method.

EXAMPLE OUTPUT:

```
Anarky                                  Index: 0        Found Index: 0
Arkham Manor                            Index: 1        Found Index: 1
Azrael                                  Index: 2        Found Index: 2
Azrael volume 2                         Index: 3        Found Index: 3
Azrael: Agent of the Bat                Index: 4        Found Index: 4
Batgirl                                 Index: 5        Found Index: 5
Batgirl volume 3                        Index: 6        Found Index: 6
Batgirl volume 4                        Index: 7        Found Index: 7
Batman                                  Index: 8        Found Index: 8
Batman '66                              Index: 9        Found Index: 9
```

Turn this Wikipedia pseudocode into C#:

```
// initially called with low = 0, high = N-1. A is a sorted list.
  BinarySearch(A[0..N-1], searchTerm, low, high) {
      if (high < low)
          return -1 // -1 means not found
```

```
    mid = (low + high) / 2
    if (A[mid] > searchTerm)
        return BinarySearch(A, searchTerm, low, mid-1)
    else if (A[mid] < searchTerm)
        return BinarySearch(A, searchTerm, mid+1, high)
    else
        return mid //the searchTerm was found
}
```

GRADING: 20 POINTS

COMMON MISTAKES:

-1: in Binary Search, you should only call the CompareTo method once and store the result instead of calling it twice.

-1: Binary Search should return the index if found or -1 if not found

-5: binary search code was not modified to work with strings and doesn't return the correct index.

-2: the binary search needs an exit condition for when min > max. If this condition happens, then you need to return -1 to indicate that the search item was not found. You should check the condition at the top of the binary search method.

-2: in binary search, you need to calculate the mid like this: min + (max – min)/2 OR (max + min) / 2.

-2: when recursively calling binary search, you need to do mid+1 or min-1 so you are not re-evaluating the mid point again.

-2: the lab requirements for binary search were to loop over the sorted list and call your binary search for each item in the list. Print the word, the index, and the index returned from your binary search.

-2: did not write a method for Binary Search

PART 6

NuGet & Json.NET

NuGet is the package manager for .NET – it's a place to grab helpful code from 3rd parties. For this lab, you'll need to use NuGet to grab Json.NET.

To add a reference to Json.NET, right-click the References node under your class library project and select "Manage NuGet Packages…". Select the "Browse" link in the top-left of the page that is loaded in the IDE. Enter "Newtonsoft.Json" in the search box. Select the item in the list of search results and in the right panel of the page, select Install.

Save

Now you have the information you need to add logic to the menu for the "**Save**" option. **Write a method** to serialize a sorted list to a save file. Take a clone of the unsorted, sort using one of your sort algorithms, then save the sorted list to a json file.

- Ask the user for the name of the save file. **Use ReadString to get the name of the file**.
- **If the name does not have the json extension**, add it to the file name. Look at the Path methods GetExtension, HasExtension, and ChangeExtension to make sure you get the extension set correctly.
- You will need to **serialize** the list in JSON format. Use the JSON.net library.

GRADING: 10 POINTS

COMMON MISTAKES:

- -2: not using ReadString to get the file name from the user
- -2: not ensuring the filename has a .json extension.
- -2: not changing the extension correctly
- -4: not serializing a sorted list
- -4: not serializing in JSON format
- -2: did not write a method for saving the data

RUBRIC

FEATURE	VALUE	GRADE
PART 1: The Menu	20	
PART 2: Load the file	10	
PART 3: Bubble Sort	20	
PART 4: Merge Sort	20	
PART 5: Binary Search	20	
PART 6: Saving	10	
TOTAL	100	

PROGRAMMER'S CHALLENGE

As with every programmer's challenge, remember the following…

1. Do the rubric first. Make sure you have something to turn in for the assignment.
2. When attempting the challenge, don't break your other code.
3. You have other assignments so don't sacrifice them to work on the challenges.

Sorting Challenge

Add the ability to sort the list in the opposite direction. This is an alternating behavior such that one time, the sort is ascending. Then the next time that same sort method is selected, it sorts in descending order. It alternates between ascending and descending.

```
Batman volume 2                                  Batman Incorporated
Batman The Dark Knight volume                    Batman Gotham Knights
Batman and Robin volume 2                        Batman Gotham Adventures
Batwing                                          Batman Family
Birds of Prey volume 3                           Batman Eternal
Catwoman volume 4                                Batman Confidential
Detective Comics volume 2                        Batman Beyond volume 4
Nightwing volume 3                               Batman Beyond volume 2
Red Hood and the Outlaws                         Batman Beyond
Batman Arkham Unhinged                           Batman Arkham Unhinged
Batman Incorporated volume 2                     Batman and the Outsiders vo
Talon                                            Batman and the Outsiders
Legends of the Dark Knight volume 2              Batman and Robin volume 2
Batman Li'l Gotham                               Batman and Robin Eternal
Batman '66                                       Batman and Robin
Harley Quinn                                     Batman Adventures volume 2
Grayson                                          Batman 80-Page Giant
Batman Superman                                  Batman '66
Batman Eternal                                   Batman
Gotham Academy                                   Batgirl volume 4
Gotham by Midnight                               Batgirl volume 3
Arkham Manor                                     Batgirl
Batman Beyond                                    Azrael volume 2
Red Hood Arsenal                                 Azrael Agent of the Bat
Robin Son of Batman                              Azrael
We Are Robin                                     Arkham Manor
Batman and Robin Eternal                         Anarky
--------------------------------------------------------------------------------
Press any key to continue
```