McGill
School of Continuing Studies | École d'éducation permanente

CCCS 315: Data Structures and Algorithms

Assignment #1

Type of Assignment

- *Individual Work*
- Evaluated*: This assignment is marked out of 100 points and is worth 10% of your final grade*
- Estimated Time*: 120 minutes*

Description

In this Assignment, you will study, Analyze (Big-O), and implement different LinkedLists and Arrays. You also start with a simple recursive programming and compare it with loops.

Learning outcomes being met through this assessment:

- Recognize the use of the linked-list, array, and recursive programming
- Apply Big-O notation to analysis of an algorithm

Steps to complete the assignment

This assignment is designed similarly to job interview. First, we will introduce a problem. You then need to apply critical thinking, design a solution, evaluate the solution (Big-O), then improve it if needed. Lastly, you will implement your solution.

Part 1: LinkedList

If you remember, we have already completed a Linked List in class. For this assignment, create your own code. (Do not use the build in function or classes of Java or from the textbook).

- Create a LinkedList class: Call the class **MyLinkedList**
- Your linked list is of an **int type.**
- For this Linked List you need to add the following **methods**:
  1. **Constructor**: This one can be omitted but can be added (Optional)
  2. **Add:** Add the elements to the head of the Linked List
  3. **Remove**: Given a value in int, if the Linked List has it, delete it (if the linked list has multiple places the same value, only the first occurrence needs to be deleted)
     a. **For example,** 1->2->3->1->3 and we try to delete 3 we should get 1->2->1->3
  4. **Size**: Returns the size of the Linked List, if empty, return 0
  5. **Contain**: Check if a Linked List has a value, if it does, return **true, else False**
  6. **ToString:** Returns a string
  7. **Compare:** compare two linked list to check if they have the same value
     a. **Ex:** A=1->2->3 , B=1->3->2 return false
     b. **Ex:** A=1->2->3 , B=1->2->3 return true

- Write a main function or **Main** class to test all of the methods,
    - Create a 2 linked list and test all of your methods.

Part2: LinkedList insert

Here we will create a method to find the smallest value in an array:

- We provided the code for you to start
- Main is done, you need to create **minFinder** function
    - First do it with loops
- Create another function **minFinder_recursive**, and use recursive function to create the code to find the min value (you may add more functions/methods as needed)
- Write the Pseudocode (https://en.wikipedia.org/wiki/Pseudocode) for **BOTH** of the functions and do a time analysis in Big-O for both (This step is really important)
    - For the pseudocode, you do not need to be super detailed but in steps by steps tell us how your code works
    - **Submit it in PDF**

```java
import java.util.Random; // this creates random for our program
public class App {


    public static int minFinder(int[] arr){
        int index = 0;
        //YOUR CODE GOES HERE
        // you have to do two ways, once with loops, once with Recusive (For recursive you need more functions or methods, so create them)



        //once you find the minimum value, return the index of it in the array
        return index;
    }
    public static void main(String[] args) throws Exception {
        int[] myArr = new int[100];
        Random myRand = new Random(); // creating Random object
        // Range for random to select from
        int min = 5;
        int max = 1000;

        int indexMin=0;

        for (int i = 0; i < myArr.length; i++) {
            myArr[i] = myRand.nextInt(max-min+1) + min; // storing random integers in an array
         }
         // here we check the last item of array to see if it is full:
         System.out.println(myArr[ myArr.length-1]); // printing last element, not the biggest or smallest but just the last one
         //now we find the minimum calling the function you wrote:
         indexMin=minFinder(myArr);
         System.out.println("The minimum is at location: "+indexMin+" The value is: "+myArr[indexMin]);
    }
}
```

Code can be downloaded in our github:
https://github.com/farhadrclass/CCCS315/blob/main/Assignment_1/src/App.java

Evaluation Criteria

- Format Requirements (10%)
  - Comments and CamelCase is important
  - Style
- Part 1 (30%)
  - Each method has 4% = 24%
  - Testing them and main = 6%
- Part 2 (40%)
  - Recursive method: 10%
  - Loop method: 10%
  - Pseudocodes: 5% (each 2.5%)
  - Big-O analysis: 15%

Submitting your Work

- You have to zip all the Java code for part 1 and part 2 (.java codes and if missing you will lose 85%)
- **PDF for Pseudocode**

Notes:

- Cheating or copying online without referencing correctly or explaining your code before and after online source has automatic failure as a grade.
  - If your references are not correct, you will lose marks.
  - (In Comments) Show your code before searching online AND showing what the issue was.
  - (In Comments) The reference.
  - (In Comments) How the online or book reference helped you change your code.
  - **Copying 80% from online with reference still results in ZERO!**
    - **Modify it to fit your code, do not blindly copy it and remember you are here to learn !**
- Do NOT share your code with anyone except your TA and me.
  - otherwise you will get a grade of ZERO.