# CSC/DSCI 2720: Data Structures Assignment 4

## Due: 04/09/2024 @ 11:59 PM ET

Answer the below questions.
You may use whatever IDEs / editors you like, but you must submit your responses on iCollege as .py files. Failure to comply with this simple requirement will result in a score of Zero. Please, be careful not to be assigned a Zero score this way.

Few Rules to be followed, else will receive a score of ZERO.

(1) Your submissions will work exactly as required.

(2) Your files shall not be incomplete or worse corrupted such that the file does not compile at all. Make sure you submit a file that compiles.

(3) Your submission will show an output. Should you receive a Zero for no output shown do not bother to email me with "but the logic is perfect"!

Note that your program's output must exactly match the specs (design, style) given here for each problem to pass the instructor's test cases.
Design refers to how well your code is written (i.e., is it clear, efficient, and elegant), while Style refers to the readability of your code (commented, correct indentation, good variable names).

1. In this problem we will explore a way to delete the root node of the Binary Search Tree (BST) while maintaining the Binary Search Tree (BST) property after deletion. (60 points)

Your implementation will be as stated below:

[1] Delete the root node value of the BST and replace the root value with the appropriate value of the existing BST.

[2] Perform the BST status check by doing an In-Order Traversal of the BST such that even after deletion the BST is maintained.

You will need to use the given class to represent tree node:

---

```
# Definition for a binary tree node.

class TreeNode:
    def __init__(self, val):
        self.val = val
        self.left = None
        self.right = None
```

---

Below is an illustrated sample of Binary Tree nodes for your reference:

```
root = TreeNode(4)
root.left = TreeNode(2)
root.right = TreeNode(6)
root.left.left = TreeNode(1)
root.left.right = TreeNode(3)
root.right.left = TreeNode(5)
root.right.right = TreeNode(7)
```

2. Repeated DNA Sequences: The **DNA sequence** is composed of a series of nucleotides abbreviated as 'A', 'C', 'G', and 'T'. (40 points)

   For example, "ACGAATTCCG" is a DNA sequence. When studying DNA, it is useful to identify repeated sequences within the DNA.

   Given a string s as input that represents a DNA sequence, return all the 10-letter-long sequences (substrings) in an array that the substrings occur more than once in a DNA molecule. You may return the answer in any order. Hint: You may a HashMap or a HashSet to check if a sequence already occurred.

   **Example 1:**
   **Input:** s = "AAAAACCCCCAAAAACCCCCCAAAAAGGGTTT"
   **Output:** ["AAAAACCCCC","CCCCCAAAAA"]

   **Example 2:**
   **Input:** s = "AAAAAAAAAAAAA"
   **Output:** ["AAAAAAAAAA"]

Very Very Important:

(1) Your code should be well commented which explains all the steps you are performing to solve the problem. A submission without code comments will immediately be deducted 15 points!

(2) As a comment in your code, please write your test-cases on how you would test your solution assumptions and hence your code. A submission without test cases will immediately be deducted 15 points! Example of cases to be tested for are like: What if the array input which is expected does not exist - that is, input is a null. How should your code handle such a situation? Maybe output some message like “Null input case, so no output”? What if the length of the array is one? … so on and so forth.

Please Remember: Although, written as comments - You will address your test cases in the form of code and not prose :)