**Assignment 1 – Coding and Complexity**

**Deadline: Thursday, January 25, 2025, 9:59 pm**

**Objectives:**

- Familiarize yourself with the basics of asymptotic analysis of algorithms.
- Start designing algorithms to solve problems.

**Instructions:**

You are not allowed to use advanced features of Java such as **Collections (e.g., ArrayList, LinkedList), or contains**. You MUST use arrays of characters and loops.

For problems marked as **(code)** – You should write a Java program that will be run by the instructor. You should provide a main method that calls your algorithm and prints to console both the input and output.

For problems marked as **(text)** – You should write an answer. The answer can be either prose, an algorithm (pseudo-code or code), a proof, or mathematical equations. Whichever form of text answer is appropriate to solve the problem. In case the answer is an algorithm (code or pesudo code), it will be assessed mostly on logic rather than on whether it compiles.

**Problems:**

1. **(code) Common subsequence [30 points]**

   A subsequence of a string is a new string generated from the original string with some characters (can be none) deleted without changing the relative order of the remaining characters.

   For example, "ace" is a subsequence of "abcde".

   Write an algorithm to solve the following problem:

   Given two strings text1 and text2, return the length of their longest common subsequence. If there is no common subsequence, return 0.

   **Examples:**

   Input: text1 = "abc", text2 = "abc"
   Output: 3
   Explanation: The longest common subsequence is "abc" and its length is 3.

   Input: text1 = "almanacs", text2 = "albatross"
   Output: 3
   Explanation: The longest common subsequence is "alas" and its length is 4.

   Input: text1 = "almanac", text2 = "ferris"
   Output: 0
   Explanation: There is no such common subsequence, so the result is 0.

2. **(Text) Common Substring [15 points]**
A substring of a string is a new string generated from the original string without changing the order of the characters.

For example, "liar" is a subsequence of "peculiar".

Given two strings text1 and text2, write an algorithm that returns the longest common substring. If there is no common subsequence, return an empty string “”. Case sensitive. Therefore, A and a are consider different characters. If there are multiple common substrings with the same length, your algorithm can return any of them.

**Examples:**

Input: text1 = "spy family", text2 = "jujutzu"
Output: ""
Explanation: There is no common substring between the two strings.

Input: text1 = "gears of war", text2 = "History of warriors"
Output: "of war"
Explanation: The longest common substring is "of war".

Input: text1 = "spy family", text2 = "jujutsu kaisen"
Output: "a" or "i" or "s"
Explanation: There are three possible common substrings between the two strings. In this case, your algorithm might return any of them

3. **(code) Not Fibbonacci [15 points]**
Write a program that produces the NotFibonacci sequence, where the next number in the sequence is produced by combining the previous two numbers. The rule for generating the NotFibonacci sequence is follows:
   - Start with the initial values of 0 and 1.
   - For each subsequent term, take the previous term, multiply it by 3, add it to the term before the previous multiplied by 2, and append the result as the next term in the sequence.

$$n_i = (3 * n_{i-1}) + (2 * n_{i-2})$$

The program should be able to take an input of the number of terms in the sequence and produce the sequence up to that number. For example, if the user inputs 10, the program should output the first 10 numbers in the NotFibonacci sequence: 0, 1, 3, 11, 39, 61, 182, 547, 1640, 4921.

**Hint:** Use Long or BigInteger instead of int or Integer

4. **(code) Where in Sequence [10 points]**
   Write a program to solve the following problem.
   Given an integer output the position of that number in the NotFibonacci sequence from problem 3. If the given input is not a number in the sequence, return the position of the closest number lower than the input.

   Example 1:
   Input: 8
   Output: 4
   Explanation: The number 8 is not in the NotFibonacci sequence. Therefore, the closest lower number is 7 which is the 4th number in the NotFibonacci sequence.

   Example 2:
   Input: 1640
   Output: 9
   Explanation: The number 1640 is the 9th number in the NotFibonacci sequence.

5. **(code) Remove Element [10 points]**
   Write a program to solve the following problem.
   Given an integer array **nums** and an integer **val**, remove all occurrences of **val** in **nums** in-place. Then return the number of elements in **nums** which are not equal to **val**.

   You need to:
   - Change the array **nums** such that the first k elements are the elements which are not equal to val.
   - Return the number of elements in **nums** which are not equal to **val**.

   This is your first LeetCode problem, https://leetcode.com/problems/remove-element/, you should solve it on LeetCode as that will allow you to check that your solution is correct. However, when submitting your solution to me, it should be part of your IntelliJ project.

6. **(text) Algorithm Analysis [20 points]**
   For each of the algorithms you wrote for problems 1-5, explain their time complexity using Big-O and Big-Ω notation. Explain how you arrived at your answer.

**Extra credit:**
**(text)** Generate a plot of the first 1000 numbers in the NotFibonnacci sequence [5 points]
**Hint:** You will encounter an issue with the numbers being very large. When this happens, simply explain what the problem is and include your plot.

**Grading Rubric:**

| Item | Points |
|---|---|
| Common substring | 30 |
| Common subsequence | 15 |
| NotFibonnacci | 15 |
| Where in Sequence | 10 |
| Remove Element | 10 |
| Algorithm analysis | 20 |
| Extra credit | 5 |
| **Total Points** | 100 |

**Deliverables:**

On Canvas, (1) The link to a GitHub repository with the implementation of the (code) problems done using an IntelliJIDEA project and (2) a pdf with the answers to the (text) problems. Make sure you double-check that your code was uploaded correctly to GitHub and that the link and pdf file was uploaded correctly on Canvas! I will not be accepting excuses after the deadline that the files were not uploaded correctly.

You may submit a physical (by hand) solutions to (text) problems. Please be sure your work is written clearly and readable.

**Compiling -- There will be an automatic 5% penalty for each compile error in your code that has to be fixed in the grading process -- up to a maximum of 10 compile errors. After 10 errors fixed, if it still fails compilation, the submission will be marked as a 0. (Bottom line: Make sure your code compiles before you submit it!!!)**

**Failure to submit all the required files in the appropriate format will result in a 10% penalty.**