Assignment 4. Due Date: Monday Jun 30, 12:30 pm (afternoon)

Notes

- Use python3 only.
- Comment your code clearly.
- Max Points: 16
- For this assignment, you need to **write recursive programs**.
- You are **not allowed to use any looping (for/while) constructs**.

1. **Recursive Operations on Polynomials**: A polynomial $P(x) = a_0 + a_1x + a_2x^2 + a_3x^3 + \ldots + a_nx^n$ can be stored as a sequence of its coefficients i.e. $a_0, a_1, a_2, a_3, \ldots a_n$. Any operation on polynomials can be defined in terms of the operation over their respective coefficients. In order to represent the coefficients of a polynomial, we will be using list data structure. In this framework, a list of length $n+1$ represents a polynomial of degree n, where the k^{th} element of the list represents the coefficient of the term x^k in the polynomial.

   (a) Write a recursive Python function *addPoly* which takes two polynomials represented as lists $l1$ and $l2$, respectively and returns a list $lsum$ representing the sum of polynomials $l1$ and $l2$. **(3 points)**

   (b) Write a recursive Python function *multPoly*($l1, l2$) which inputs two polynomials represented as lists $l1$ and $l2$, respectively, and returns a list $lmult$ representing the multiplication of polynomials $l1$ and $l2$. You are free to call *addPoly* function defined in part (a) above. You are also allowed to write any additional auxiliary recursive functions (no loops). **(4 points**)

   Note: You are free to use the slicing, append and extend operations on list(s). You can also use the '+' operator which takes two lists $l1$ and $l2$, and returns a new list with $l2$ appended at the end of $l1$. Operator $[v]$ returns a new list with a single element v in it. You are not allowed to use other operations such as insert, remove or pop on lists.

2. **Longest Common Subsequence:** Given a sequence S of elements, another sequence Q is said to be its subsequence if Q can be obtained from S by deleting 0 or more elements in S without changing the order of the remaining elements in S. We can think of a string as a sequence of characters. Then, subsequences of a string can be defined as above. For example, given the string "grass is green and sky is blue", "green and blue" and "grass and sky" are its subsequences whereas "sky is grass" is not.

   (a) Given two input strings, write a recursive Python function to find their longest common subsequence. **(4 points)**

   (b) Given two input strings of length m and n, what is the time complexity of your function in terms of m and n. Write it as a comment at the end of your program. **(1 point)**

For this question, you may need to use the slicing operation on strings. Given a string str, $str[k1 : k2]$ returns a new string with str sliced between indices $k1$ and $k2$, similar to the slicing operation defined over lists. Similarly, $'+'$ operator over two strings $str1$ and $str2$ returns a new string which is a concatenation of $str1$ and $str2$.

Food for Thought: (No Extra Credits): There is an $O(mn)$ algorithm which can solve the problem of finding the longest common subsequence. How would you design it? Think about storing the results of repeated calculations during the execution of your program in part (a) above.

3. In the minor exam, we looked at the problem of separating out odd and even numbers in a given list. In particular, given a list of numbers L, the question asked you to create a new list $Lsep$ with elements in L re-arranged such that all the odd numbers in $Lsep$ appeared before all the even numbers. See the Minor Question for details. In this question, you are required to do perform the same operation in place, i.e., without creating a new list, and using a recursive function. Unlike the minor question, we do not care about the specific order in which odd/even numbers appear in $Lsep$ (i.e. it is ok to have them in an order different from how they appear in L). (**4 points**).

   Note: Since the required function should be in place, you are not allowed to use the slicing operation on lists for this question.

   Hint: Think of the partition function used in QuickSort.