Problem 1:Arraylist

Create a class called OurArrayList and implement the following methods. **Do NOT create any auxiliary memory (don't create an extra arraylist or a normal array, etc..).** Include a main method and test all your methods with appropriate examples.

a. Write a method called **scaleByK()** that takes an ArrayList of integers as a parameter and replaces every integer of value K with K copies of itself. For example, if the list stores the values (4, 1 , 2, 0 ,3) before the method is called, it should store the values (4, 4, 4, 4, 1, 2, 2, 3, 3, 3) after the method finishes executing. Zeroes and negative numbers should be removed from the list by this method.

b. Write a method **markLength4()** that takes an ArrayList of Strings as a parameter and that places a String of four asterisks ("****") in front of every String of length 4. For example, suppose that an ArrayList called "list" contains the following values:

(this, is, lots, of, fun, for, every, Java, programmer)

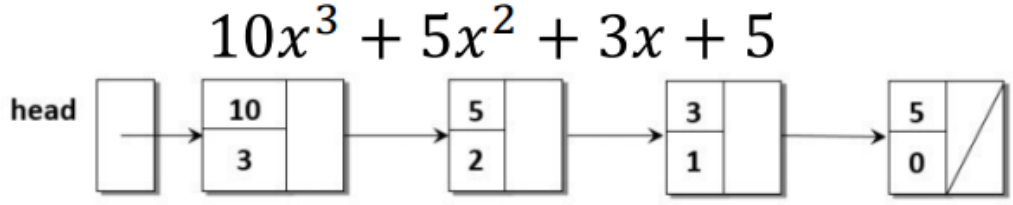
And you make the following call**: markLength4(list)**; Then list should store the following values after the call:

(****, this, is, ****, lots, of, fun, for, every, ****, Java, programmer)

Notice that you leave the original Strings in the list (this, lots, Java) but include the four asterisk String in front of each to mark it. You may assume that the ArrayList contains only String values, but it might be empty.

Problem 2: **Polynomials**

In this exercise, we will implement a $Polynomial$ class to represent a polynomial equation using linked list where each node corresponds to a term cx^e, where c is the coefficient and e is the exponent. The following is an example of the representation structure of the polynomial:



1. Write the class $PolynomialNode$ that contains the fields $coef$ and exp of type int and an attribute $next$ that refers to the next node.
2. In the $PolynomialNode$ class, write 3 constructors that are defined as follows:

a. $PolynomialNode$ (): initializes the $coef$ and exp to 0, and $next$ to $null$.
b. $PolynomialNode$ ($int\ coef$, $int\ exp$): assigns values $coef$ and exp, and $next$ to $null$.
c. $PolynomialNode$ ($int\ coef$, $int\ exp$, $PolynomialNode\ next$): assigns values $coef$, exp, and $next$.
3. Write the class $Polynomial$ having a default constructor that initializes a private attribute "$PolynomialNode\ head$" to $null$.
4. In the $Polynomial$ class, create the following methods:
a. $public\ boolean\ isEmpty$(){… }; returns true if the $head$ list is empty.
b. $public\ void\ insert$($int\ coef$, $int\ exp$){… }; adds a $PolynomialNode$ to the front of the $head$ list.
c. $public\ void\ remove$(){… }; removes the front node of the $head$ list.
d. $public\ String\ toString$(){… }; prints a given polynomial on the screen (for example the output string of the previous example is: 10x^3+5x^2+3x+5).
e. $public\ Polynomial\ add$($Polynomial\ p$){… }; sums the 2 given polynomials ($this$ and p). The result will return a $Polynomial$ object representing the summation. For example, the sum of the polynomials: $[10x3 + 5x2 + 3x + 5]+[3x3 - 7x2 + 3x - 6]=[13x3 - 2x2 + 6x - 1]$. Note: if the sum of the coefficients is 0 then you should not include the node in the list.
f. $public\ int\ evaluate$($int\ x$){… }; calculates and returns the value of the polynomial for a given value x.
g. $public\ Polynomial\ derivate$(){… }; calculates the derivative of a polynomial and returns a new $Polynomial$ Linked List. For example, if the polynomial list represents $[10x3 + 5x2 + 3x + 5]$ then the returned polynomial list after derivation is $[30x2 + 10x + 3]$
5. Run the main method provided with the zip folder to validate your work.

Problem 3: **ArrayList**

Write a method called **removeInRange()** that accepts three parameters , an ArrayList of Strings, a beginning String, and an ending String, and removes from the list any Strings that fall alphabetically between the start and end Strings inclusive. For example, if the method is passed a list containing the elements("to", "be", "or", "not", "to", "be", "that", "is" , "the", "question"), "free" as the start String, and "rich" as the end String, the list's elements should be changed to ("to", "be", "to", "be", "that", "the"). The "or", "not", "is", and "question" should be removed because they occur alphabetically between "free" and "rich". And if the method is passed the same list with "be" as a starting string and "right" as the end String, the list's elements should be changed to ("to", "to", "that", "the"). You may assume that the start String alphabetically precedes the ending String.