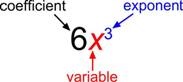
Don’t take any input from the user.

Write a Java program that performs function operations on ***single-linked lists***.

In the assignment, two-variable functions must be represented as single linked lists.

A *two-variable function* has terms consisting of both *x* and *y* variables, and it is represented by *f(x, y)*.

|  |  |  |  |
| --- | --- | --- | --- |
| Coefficient | Exponent *x* | Exponent *y* | next |



Example:

f(x,y) = 2x5y6 + x5y4 + 4x5y3 + 6x4y4 + x3y2 + 6xy+ 2

head

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 2 | 5 | 6 |  |  | 1 | 5 | 4 |  |  | 4 | 5 | 3 |  |  | 6 | 4 | 4 |  |  | 1 | 3 | 2 |  |  | 6 | 1 | 1 |  |  | 2 | 0 | 0 | null |

The program should perform the following function operations:

1. Create two functions randomly
2. Sum two functions
3. Find *n*th order partial derivatives of the final function

------------------------------------------------------------------------------------------------------------------------------------------------------

**1. Create two functions randomly**

Write a method (“add”) that adds a term to the function.

Call this method many times in a loop to create a function with 7 terms by generating three random numbers (coefficient, exponent x, exponent y) each time between 0 and 6.

Each term must be different from each other, for example the following expression is wrong 2x3y2 + 4x3y2.

The terms in the function must be in descending order (greatest to least), firstly according to the exponent of *x*, after that (if equal) sorted according to the exponent of *y*. The function only contains plus (+) operation.

Example:

SingleLinkedList f1 = new SingleLinkedList();

// write in a loop by generating three random numbers

f1.add(1,5,4); f1.add(6,1,1); f1.add(2,5,6); f1.add(6,4,4); f1.add(2,0,0); f1.add(1,3,0); f1.add(4,5,3);

f(x,y) = 2x5y6 + x5y4 + 4x5y3 + 6x4y4 + x3y2 + 6xy+ 2

head

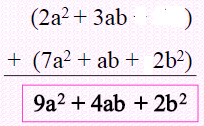
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 2 | 5 | 6 |  |  | 1 | 5 | 4 |  |  | 4 | 5 | 3 |  |  | 6 | 4 | 4 |  |  | 1 | 3 | 2 |  |  | 6 | 1 | 1 |  |  | 2 | 0 | 0 | null |

// Also create the second function

SingleLinkedList f2 = new SingleLinkedList();

// Print the functions on the screen

------------------------------------------------------------------------------------------------------------------------------------------------------

**2. Sum two functions**

Write a method that sums two functions. Print on the screen.

// The single linked lists of function 1 and function 2 must remain the same after sum operation.

// Function 3 (summed one) must also be stored as a single linked list.

------------------------------------------------------------------------------------------------------------------------------------------------------

**3- Find *n*th order partial derivatives of the final function**

Generate a number *n* randomly (between 1 and 4) and then print the *n*th order partial derivatives of the function 3.

Example: 2x2 + 3xy + 6x + 7y

n=2

Outputs: 4 (the second order partial derivative of *f3* with respect to *x*)

0 (the second order partial derivative of *f3* with respect to *y*)

-----------------------------------------------------------------------------------------------------------------------------------------------------------------------