Part 2: Create A Polynomial

Polynomial* create_polynomial(int[] terms, N)

This function will take an array of pairs and an integer, N, which specifies how many pairs you select from this array to create a linked list of terms. You can assume the array will always be terminated by (0, -1).

If N is greater than the # of pairs in the array, (not including (0, -1)) 0, or less than 0, the function will simply create a linked list of all the pairs in the array. The terms created in the linked list **must** be sorted in **descending order, in accordance with exponents**.

Do not sort by coefficients.

If there are pairs in the array that have the **same exponent and the same coefficient, i.e. duplicate pairs**, then **only one** of said pairs should be part of an array.

For example, if we had the array, [(3, 5), (3, 5), (0, -1)], with N = 2, the created linked list should be:
[Term (3, 5) -> NULL].
# of Terms = 1

If there are pairs in the array that have the **same exponent and different coefficients,** they should be treated as “like terms” in a polynomial, where you add the coefficients and make a new term using the result of the addition.

For example, if we had the array, [(3, 5), (2, 5), (0, -1)], with N = 2, the created linked list should be:
[Term (5, 5) -> NULL].
# of Terms = 1

If N is in the range [0 < N < # of pairs] the function will create a linked list of terms with the **N highest pairs** in the array. Highest pairs are decided by **exponent only**, and **not coefficient**. If you had these two pairs, (3, 5) and (4, 5), the function should simply select the pairs in the order they appear. Again, you do **NOT select by coefficient.**

You select the N highest pairs, remove duplicate pairs, and add like terms. You do it in this order. You DO NOT remove duplicates and then add like terms during the selection process!

For example, if we had the array, [(4, 3), (6, 4), (7, 1), (4, 3), (4, 4)], with N = 4, your **selected pairs that will be used to create the linked list are:** [(6, 4), (4, 4), (4, 3), (4, 3)]. These are the **4 highest pairs in the array.**

Notice how **duplicates were not removed and like terms were not merged.**

Now, using this list of selected pairs, we will remove duplicates/merge like terms, resulting in:

[Term (10, 4) -> Term (4, 3) -> NULL] # of Terms = 2.

This function should create a polynomial struct with a pointer to the linked list of terms. You should return the pointer to the polynomial struct in $v0. If all of the terms in the polynomial cancel out, or there are simply no valid pairs for selection to create a linked list, **return 0 in $v0. Do NOT return a pointer to a null head of the polynomial struct. Just 0.**