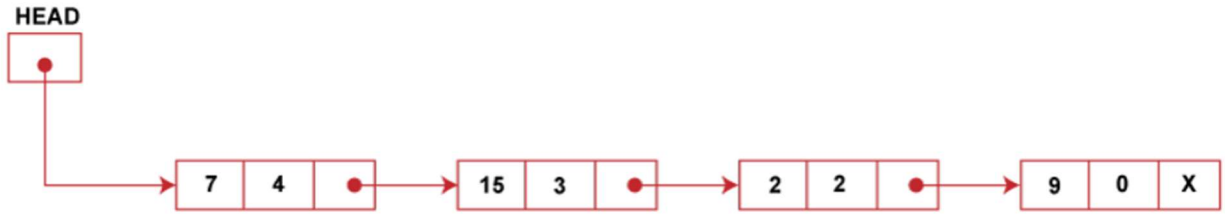
Linked List - Exercise

1) **Polynomial Manipulation:**

Consider a polynomial represented using a linked list with each node containing each term of the polynomial. Hence the structure of a node (term) is as follows:

Coefficient	Exponent	Link

For easing arithmetic operation, the nodes are arranged in decreasing order of their exponent. For example a polynomial $P(x) = 7x^4 + 15x^3 - 2x^2 + 9$ is represented as:



Perform the following operations:

a) Create a polynomial by inserting terms considering the following cases:
   i) Terms are input in decreasing order of their exponents
   ii) Terms are input in any order of their exponents
   Note: terms should be arranged in decreasing order of their exponent

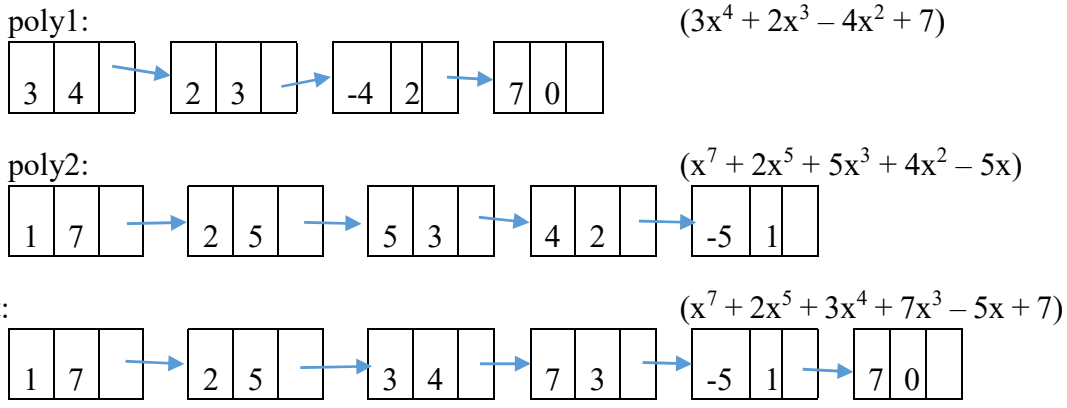
   Function Declaration: struct Node * createPolynomial(struct Node * P, int coeff, int exp);

b) Given two polynomials, implement a C function to add the two polynomials.

Function Declaration: struct Node * addPolynomial(struct Node * poly1, struct Node * poly2);

Sample:

Input:



2) **After Kth Largest:**

Given an increasing order sorted singly linked list and an integer k > 0, write a function (afterKthLargest) to return a list of all nodes occurring after the k^{th} largest element in the list. The order of elements in the resultant list should be the same as that in the input list.

Function Declaration: struct Node* afterKthLargest(struct Node * head, int k);
Sample:

Input: 1 → 2 → 4 → 8 → 9 → 12, k = 3
Output: 9 → 12

Input: 1 → 2 → 4 → 8 → 9 → 12, k = 5
Output: 4 → 8 → 9 → 12

Input: 1 → 2 → 4 → 8 → 9 → 12, k = 10
Output: 1 → 2 → 4 → 8 → 9 → 12

Input: 1 → 2 → 4 → 8 → 9 → 12, k = 1
Output: NULL

3) **Remove Duplicates:**

Given a singly linked list, write a function (removeDuplicates) to remove all elements that occur more than once in the list. The order of elements in the resultant list should be the same as that in the input list.
Function Declaration: struct Node* removeDuplicates(struct Node * head);
Sample:

Input: 1 → 2 → 4 → 2 → 9 → 3
Output: 1 → 4 → 9 → 3

Input: 1 → 2 → 4 → 10 → 9 → 3
Output: 1 → 2 → 4 → 10 → 9 → 3

Input: 1 → 2 → 1 → 10 → 8 → 2 → 1
Output: 10 → 8

Input: 1 → 2 → 1 → 1 → 1 → 2
Output: NULL