Assignment 1

Question 1:

A railway station uses a linked list to model tracks where each node represents a train. Trains can be added to the end (new arrival) or removed from any position (departure). Your task is to implement this system.

Operations:

- add_train(id): Add a train with id to the end of the list.

- depart_train(id): Remove the first occurrence of id.

- emergency_block(pos): Remove the train at position pos (0-based) due to a track blockage.

- display_tracks(): Print all trains on the tracks.

Sample Input:
add_train("T123")
add_train("T456")
add_train("T789")
depart_train("T456")
emergency_block(0)
display_tracks()

Sample Output:
T789

Question 2:

A binary image is represented by rows of 0 (black) and . (white). Store each row as a linked list where each node contains a count of consecutive identical pixels. Mirror the image (flip horizontally) and print the result.

Sample Input:
Row 0: 000..00..0
Row 1: 0. .. 0

Sample Output (Mirrored):
Row 0: 0..00. 000
Row 1: 0. 0

Question 3:

Task:
- Create a linked list of 26 nodes (a-z) with frequency counts initialized to 0.
- Traverse a string and update frequencies.
- Print the frequencies in a-z order.

Sample Input:
"abaac"

Sample Output:
a : 3
b : 1
c : 1
d : 0
...
z : 0

Question 4:

Store two polynomials using singly linked lists (each node stores coefficient and exponent). Perform addition and return the resultant polynomial as a new linked list.

Sample Input:
Polynomial 1: 5x^3 + 2x^2 + 3x + 4
Polynomial 2: 4x^2 + 3

Sample Output:
5x^3 + 6x^2 + 3x + 7

Instructions:

- Implement each function as a separate function in C or C++.

- Ensure the linked list is used efficiently.

- Test your code with multiple test cases.

Submission Guidelines:

- Submit a `.cpp` or `.c` file with properly commented code.

- Clearly mention any assumptions made.