CS204(2025)
Linked List-II
Instructions for Implementation

- Write the program in C or C++ using standard input/output.
- Follow the input/output format strictly.
- Ensure your code:
- Handles edge cases correctly.
- Meets the time and space complexity constraints.
- Do not use STL containers like vector, set, or map in C++. Use raw arrays unless explicitly allowed.
- You are encouraged to write clean, modular, and well-documented code.

Q1. Rotate a Circular Linked List by K (Clockwise)
Problem Statement:
Given a circular singly linked list and a non-negative integer K, rotate the list clockwise by K positions. Print the list starting from the new head. If the list is empty, print -1.
Constraints:
$1 \le N \le 10^4$

$0 \le$ node value $\le 10^5$

$0 \le K \le 10^9$

Input Format:
Line 1: integer K
Line 2: list values separated by spaces and terminated by -1
Output Format:
One line containing the rotated list (space-separated) or -1.
Test Cases:
Input:

```
2
1 2 3 4 5 -1
```

Output:

```
4 5 1 2 3
```

Input:

```
0
10 20 30 40 50 60 -1
```

Output:

```
10 20 30 40 50 60
```

Q2. Delete All Occurrences of X in a Doubly Linked List
Problem Statement:
Given a doubly linked list and an integer X, delete every node whose value equals X and print the resulting list. If the list becomes empty, print -1.
Constraints:
1 ≤ N ≤ 10^4

0 ≤ node value ≤ 10^5

Input Format:
Line 1: integer X
Line 2: list values separated by spaces and terminated by -1
Output Format:
One line containing the filtered list (space-separated) or -1
Test Cases:
Input:
3
1 3 3 2 3 4 3 -1

Output:
1 2 4

Input:
10
10 10 10 10 10 -1

Output:
-1

Q3. Merge Two Sorted Doubly Linked Lists
Problem Statement:
You are given two non-decreasing doubly linked lists. Merge them into a single non-decreasing list, preserving duplicates.
Constraints:
0 ≤ N1, N2 ≤ 10^4

0 ≤ node value ≤ 10^5

Input Format:
Line 1: list A values separated by spaces and terminated by -1
Line 2: list B values separated by spaces and terminated by -1
Output Format:
One line containing the merged list (space-separated) or -1 if empty
Test Cases:
Input:

```
1 3 5 -1
2 3 4 6 -1
```

Output:

```
1 2 3 3 4 5 6
```

Input:

```
-1
-1
```

Output:

```
-1
```

Q4. Split a Circular Linked List into Two Halves
Problem Statement:
Given a circular singly linked list of N nodes, split it into two circular lists:
If N is even, both halves have N/2 nodes.

If N is odd, the first half has (N+1)/2 nodes and the second has (N−1)/2 nodes.

Print both halves starting from their respective heads (two lines). If the list is empty, print -1.
Constraints:
0 ≤ N ≤ 10^4

0 ≤ node value ≤ 10^5

Input Format:
Single line: list values separated by spaces and terminated by -1
Output Format:
If empty, print -1. Otherwise, print two lines: first half, then second half.
Test Cases:
Input:
```
1 2 3 4 5 6 -1
```

Output:
```
1 2 3
4 5 6
```
Input:
```
10 20 30 40 50 -1
```
Output:
```
10 20 30
40 50
```

Q5. Sorting IP Addresses Using Insertion Sort on a Doubly Linked List

Problem Statement

You are given a set of IPv4 addresses (e.g., 172.16.117.9). Each IP address consists of four parts (octets) separated by dots.

Implement a Doubly Linked List where each node stores an IP address in four separate integer fields:

```
// Structure to store an IP address in a doubly linked list
typedef struct Node {
   int octet1;      // First part of IP
   int octet2;      // Second part of IP
   int octet3;      // Third part of IP
   int octet4;      // Fourth part of IP
   struct Node* prev; // Pointer to previous node
   struct Node* next; // Pointer to next node
} Node;
```

Insert the given IP addresses into the doubly linked list.

Implement Insertion Sort on the doubly linked list to sort the IP addresses in non-decreasing lexicographic order (compare first octet, then second if equal, then third, then fourth).

Print the sorted list of IP addresses.

If the input is empty, print -1.

Constraints

0 ≤ N ≤ 10^4 (number of IP addresses)

0 ≤ each octet ≤ 255

Input Format

Each line contains IP addresses separated by spaces and terminated by #.

Output Format

One line containing the sorted IP addresses (space-separated).

Test Case 1

Input:

172.16.117.9 10.0.0.1 192.168.0.5 172.16.1.1 #

Output:

10.0.0.1 172.16.1.1 172.16.117.9 192.168.0.5

Test Case 2

Input:

#

Output:

-1

Q6. Josephus Problem using Circular Linked List
Problem Statement:
There are N people standing in a circle, numbered from 1 to N. Starting from person 1, we eliminate every k-th person in the circle. After each elimination, counting resumes from the next person still in the circle. This process continues until only one person remains, who is the survivor.
Your task is to simulate this process using a Circular Linked List and print the position of the survivor.
Constraints:
1 ≤ N ≤ 10^5

1 ≤ k ≤ N

Input Format:
Line 1: N people numbered as 1 2 … N -1
Line 2: integer k
Output Format:
One line containing the survivor position
Test Cases 1:
Input:
1 2 3 4 5 -1
2

Output :
3
Elimination order: 2 → 4 → 1 → 5
Survivor: 3

Input 2:
1 -1
1
Output 2:
1

Q7. Bank Counter Simulation (Round Service)

Problem Statement:
Design and implement a program in C/C++ to simulate a Bank Counter Service using a Circular Linked List.

Each customer is represented as a node in the circular linked list, where each node contains:
- Customer Name (string)
- Service Time Required (integer, in minutes)

The clerk serves the customers in round-robin fashion, giving 5 minutes per turn.
- If a customer's remaining service time becomes 0 (or less), that customer is removed from the circular list.
- Continue until all customers are served.
- Print the order in which customers finish their service.

Constraints:
- 1 ≤ Number of customers ≤ 10^4
- 1 ≤ Service time ≤ 10^5

Input Format:
- Line 1: Integer k, the total number of customers.
- Next k lines: Customer Name and Service Time Required (space-separated).

Output Format:
- One line containing the order in which customers are fully served (space-separated).

Test Case 1:
Input:

3
C1 10
C2 4
C3 7

Output:
C2 C3 C1

Test Case 2:
Input:
4
A 3
B 8
C 6
D 4
Output:
A D C B

Q8. Train Compartment Management Using Doubly Linked List
Problem Statement:
A train consists of multiple compartments connected in both directions, represented using a doubly linked list.
Each node stores:
Compartment name
List of passengers traveling in that compartment (maximum of 5 passengers per compartment).
A TTE (Travelling Ticket Examiner) can start searching from any given compartment and move in either direction of the train to find a passenger.
Given the initial compartment where the TTE is present and the target passenger name, the program should search for the passenger across the train and:
Print the compartment name if the passenger is found.
Print -1 if the passenger is not present in the train.

Input Format:
First line: Total number of compartments n
Next n lines: Compartment name followed by passenger names (terminated by -1)
Last line: Compartment name where the TTE starts followed by the target passenger name

---

Output Format:
Print the compartment name where the passenger is found
Print -1 if the passenger is not found

Constraints:
$1 \leq n \leq 100$
Each compartment can have at most 5 passengers
Passenger names are strings without spaces

Example 1:
Input:
3
C1 A B -1
C2 C D E -1

```
C3 F G -1
C2 G
Output:
C3
```

```
Example 2:
Input:
2
C1 Ram Shyam -1
C2 Ravi Mohan -1
C1 Sita
Output:
-1
```