LAB Assignment Practice Session

Q1 Problem Name: **Infinite Circular Doubly Linked List**

You are given a circular doubly linked list that contains integers as the data in each node. These data on each node is distinct. You have developed a special algorithm that prints the three continuous elements of the list, starting from the first element or head of the list and runs for infinite time. For example, if the list is {1,9,12,7}, then the output of the algorithm will be {1,9,12,9,12,7,12,7,1,...}. The output contains the infinite number of elements because it is a circular list.

You are given only a part of the output that has been returned by the algorithm. Your task is to determine the number of elements available in the original list and print the respective elements.

Note

- It is guaranteed that the provided part of the output of the algorithm is sufficient enough to calculate the size of the original list.
- Please read the sample explanation carefully and use the following definition of the doubly linked list:

```
class Node {
    Object data;
    Node next;
    Node prev;

}
```

Input format

- First line: Integer N that denotes the length of the list which is returned as the output of the algorithm
- Next line: N space-separated integers that denote the elements of the list which is returned as the output of the algorithm

Output format

Your task is to print the output in the following format:

- First line: Length of the original list
- Second line: Space-separated integers that denote the elements of the original list

Constraints

$1 \leq N \leq 10^5$
$1 \leq A[i] \leq 10^9$

SAMPLE INPUT

```
10
7 12 8 12 8 13 8 13 7 13
```

Q2. Problem Name: **Memory Efficient Doubly Linked List**

Normally, a doubly linked list requires two fields for previous and next pointers other than data field(s). An XOR list requires only one field with each node other than data field(s).

The idea is to store XOR of previous and next addresses in this field.

As XOR is a special kind of operation where if you XOR the answer with one of the numbers in question, you get the other number in question.

Example: If A XOR B = C then, A XOR C = B as well as, B XOR C = A

By using this property we can create an XOR list which requires only one field for addressing.

Consider a list as follows:

A -> B -> C -> D

Here, address variable (let's call it **x_ptr**) of each node will contain following information:

Node A: x_ptr = NULL XOR address(B)

Node B: x_ptr = address(A) XOR address(C)

Node C: x_ptr = address(B) XOR address(D)

Node D: x_ptr = address(C) XOR NULL

We can traverse this linked list in both forward and reverse directions. To get the next node we will require the address of previous node.

Example: Operations to traverse from A to D (We have Node A) are as follows :

B = NULL XOR A->x_ptr

C = A XOR B->x_ptr

D = B XOR C->x_ptr

Similarly, List can be traversed in backward direction.