Assignment 5

Problem 1: Construct The Permutation

A permutation of `N` is a sequence of integers from 0 to `N-1` of length `N` containing each number exactly once. For example, (0), (4, 3, 0, 1, 2), (3, 2, 0,1) are permutations, and (1, 1), (0, 3, 1), (2, 3, 0) are not.

Bob has a secret permutation which would unlock immortality in the game "Epic Shooter 2" and provides clues to Eric to figure out this permutation.

Initially Eric is given the information of the starting element **(always 0)** and Bob would provide clues in an orderly manner to construct the permutation.

Each clue consists of 2 numbers `a` and `b`, where `a` is a number already known to Eric and `b` is a number which has not yet been part of the earlier clues. The tuple of `(a b)` implies that `b` should immediately follow `a` in the current construction.

Eric is desperate to unlock this feature in the game and needs your help in solving this problem.

Input Format:

- The first line of input contains a single integer `N` , the number of elements in the permutation.
- The next N-1 lines contain 2 integers a_i and b_i, representing the i'th clue for i in range from 1 to N-1.

Output Format:

Contains a single line containing `N` space seperated integers, the answer permutation

Constraints:

- $0 \leq N \leq 100000$
- $0 \leq a_i, b_i \leq N$

- It is guarenteed that at the time of the i'th query, a_i> is always known to Eric from before(from previous clues or starting number) and b_i is unknown.

Example:

Input 1:

```
8
0 5
5 7
5 6
0 3
3 4
0 1
1 2
```

Copy

Output 1:

```
0 1 2 3 4 5 6 7
```

Copy

Explanation

Initially Eric only knows that 0 is the starting element.
After query 1, he gets (0 5)
After query 2, he gets (0 5 7)
After query 3, he gets (0 5 6 7)
After query 4, he gets (0 3 5 6 7)
After query 5, he gets (0 3 4 5 6 7)
After query 6, he gets (0 1 3 4 5 6 7)
After query 7, he gets (0 1 2 3 4 5 6 7)
Thus the permutation is (0 1 2 3 4 5 6 7)

Input 2:

```
5
0 2
2 1
2 3
0 4
```

Copy

Output 2:

```
0 4 2 3 1
```
Copy

Problem 2: The Cursed Necklace Conundrum

In the magical land of Aelor, an ancient curse has befallen the royal family's precious necklace. The necklace, which consists of enchanted gems strung together, has been mixed up by a miscast spell. Now, the gems are arranged in a chaotic sequence, their magical properties disordered. Princess Elara needs your help to restore the gems to their proper order so the necklace can regain its power. But theres a twist—the gems cannot be moved to a new location; they must be rearranged right where they are. The solution is to use the oldest spell in the book: The Bubble Sort Incantation.

Input Format:

Do not make changes to main.c. Write your code in function.c. You are not allowed to change the function signature given as input.

Constraints:

- $1 \leq n \leq 10{,}000$
- $1 \leq array[i] \leq 100{,}000$
- Time Limit: 1s
- Memory Limit: 7Mb

Output Format:

Return the head pointer of the result Linked list.

Example 1:

Input

```
6
3 5 2 6 5 1
```
Copy

Output

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

Example 2:

Input

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

Copy

Output

```
1 2 3 4 6
```

Problem 3: Ron's Roster Riddle

Dumbledore's Army is gearing up for a crucial training exercise, but Ron's error in the roster has caused some unexpected issues. The students are standing in a line, though some names may appear multiple times due to a mix-up. Your task is to reverse their positions using a series of exponentially increasing group sizes. The tricky part? This must be done recursively and in-place—no new lists allowed.
Here's the challenge:

- Begin by reversing the first 2 students.
- Then reverse the next 4 students.
- After that, reverse the next 8 students.
- Continue this process, doubling the group size each time.

Beware—some names repeat in the list, and the total number of students can be as high as 1,00,000. You'll need to modify the list directly, without creating any new data structures, while ensuring the reversal is achieved recursively.

Input Format:

Do not make changes to main.c and main.h . Write your code in function.c. You are not allowed to change the function signature given as input and the recursive function should be called by the name recursion with void return type.

Constraints:

- $1 \leq n \leq 100000$
- $1 \leq$ `length(name)` ≤ 10(where name is the string denoting the name of each student)
- Time Limit: 1s
- Memory Limit: 7Mb

Output Format:

Return the head pointer of the result Linked list.

Example 1:

Input

```
6
Harry
Hermione
Ron
Ginny
Luna
Neville
```

Copy

Output

```
Hermione Harry Neville Luna Ginny Ron
```

Example 2:

Input

```
5
Harry
Hermione
Ron
Ginny
Luna
```

Copy

Output

```
Hermione Harry Luna Ginny Ron
```
