

Lab 6 A

Problem 1: Building the Order of Soldiers

You are responsible for organizing soldiers into a specific order for a parade. Each soldier is assigned a unique number, and they are lined up in a specific order. Occasionally, new soldiers join the line, and you need to insert them at specific positions. Your task is to place these new soldiers into the line while maintaining the existing order. If the specified position is out of bounds, insert the soldier at the end of the line.

Functions to Implement:

- `struct Node* insert_at_pos(struct Node* head, int pos, int data);`
- `struct Node* createNode(int data);`
- `void printList(struct Node* head);`

Input Format:

- The first line contains an integer n , representing the number of insertions.
- The next n lines each contain two integers: `pos` and `val`. The integer `pos` represents the position at which a new soldier (with number `val`) should be inserted in the line.

Constraints:

- $1 \leq n \leq 1000$
- $1 \leq \text{val} \leq 10^6$

Output Format:

Print the final order of soldiers after all insertions have been made. Each soldier's number should be separated by a space.

Input 1:

```
4
1 10
2 20
3 30
```

```
2 15
```

Output 1:

```
10 15 20 30
```

Input 2:

```
3
1 100
5 200
2 150
```

Output 2:

```
100 150 200
```

Explanation:

In the first example, we insert the soldiers in the following sequence:

- First, insert 10 at position 1, resulting in the list: 10
- Next, insert 20 at position 2, resulting in: 10 20
- Then, insert 30 at position 3, resulting in: 10 20 30
- Finally, insert 15 at position 2, giving: 10 15 20 30

In the second example, even though the position for the second insertion is out of bounds, 200 is inserted at the end of the list.

Important Notes:

- Input and output handling is already provided. You only need to implement the required functions.
- The positions are 1-based, meaning position 1 refers to the start of the list.

Problem 2: Sum of Two Numbers Represented by Linked Lists

You are given two non-negative integers represented as linked lists, where each node contains a single digit. Your task is to write a function that adds these two numbers and returns the sum as a linked list.

Approach:

- Traverse both linked lists, digit by digit, and compute their sum.
- If the sum of the digits exceeds 9, carry over the extra value to the next digit.
- Continue until both lists are fully traversed, and make sure to account for any remaining carry value.
- The result should also be stored as a linked list, similar to the input format.

Hint:

- Working with **reversed** linked lists are easier!

Input Format:

- First line consists of an integer n representing the size of first linked list.
- Second line consists of an n characters representing the first number.
- Third line consists of an integer m representing the size of second linked list.
- Second line consists of an m characters representing the second number.

Output Format:

The sum of the two numbers.

Constraints:

- Each linked list contains digits between 0 and 9.
- The numbers do not have leading zeroes, except for the number 0 itself.
- $1 \leq n, m \leq 100$

Example Input 1:

```
4
9340
3
721
```

Example Output 1:

10061

Example Input 2:

3
342
3
465

Example Output 2:

807

Problem 3: Reverse Linked List

You are helping a teacher organize the data for student records. The teacher has stored the student IDs in a singly linked list, with each node containing a student ID and a link to the next student ID. However, the teacher realizes that the IDs need to be displayed in reverse order to prioritize recent entries. Your task is to write a program in C to reverse the linked list, so that the last student ID becomes the first, the second-to-last becomes the second, and so on.

Input Format:

- The first line contains an integer n — the number of student IDs in the linked list.
- The next n lines contain one integer each, representing the student IDs in the order they were added.

Constraints:

- $1 \leq n \leq 3 \cdot 10^4$
- Each student ID is a non-negative integer.

Output Format:

- Print the student IDs in reversed order, with each ID on a new line.

Example:

[Copy](#)**Input:**

5

1 2 3 4 5

Output:

5 4 3 2 1

Submission Guidelines

Do not rename any files given in the handout. Only write the code in the specified C files in the respective directories.