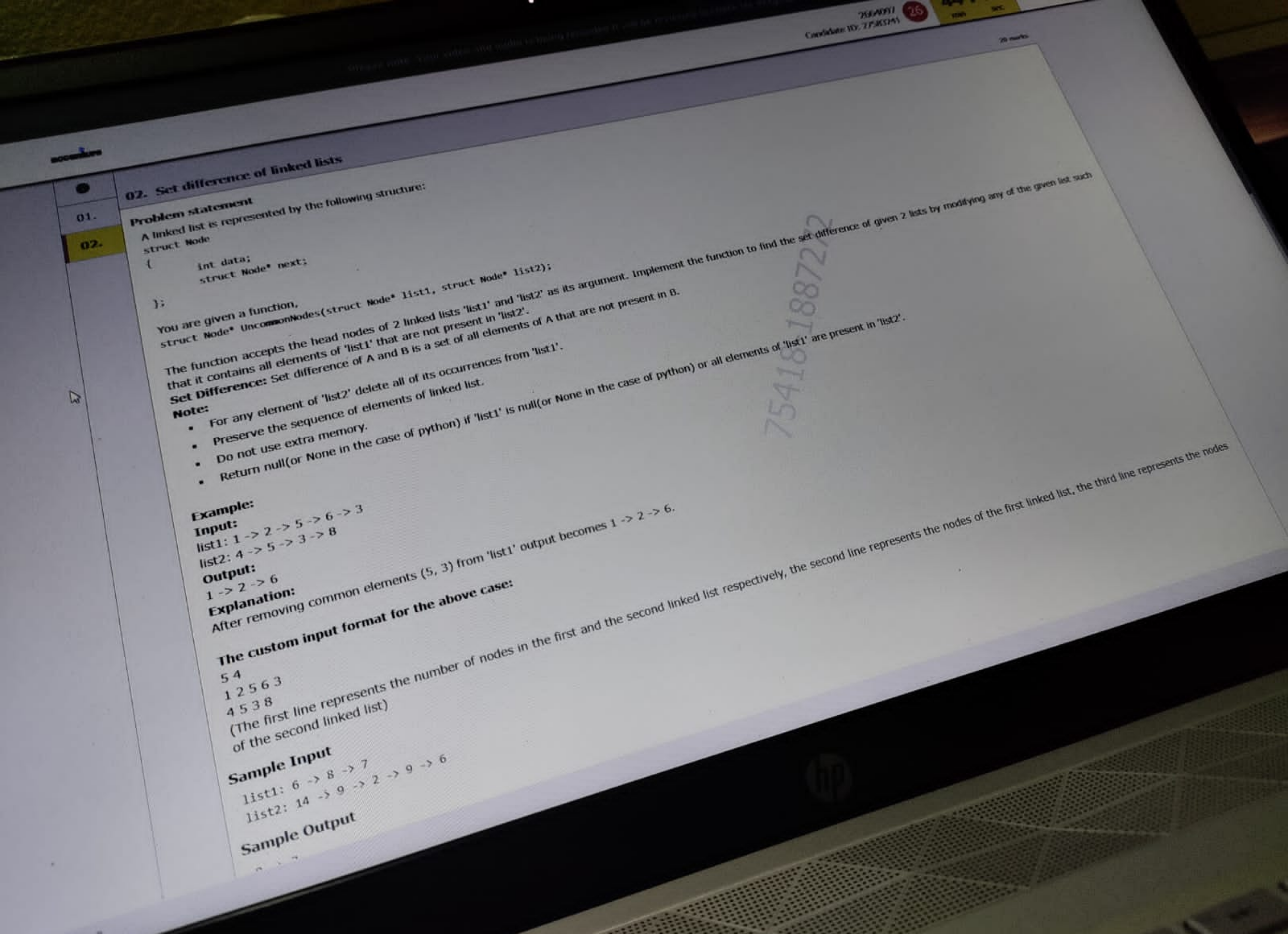


01.

02.

## 02. Set difference of linked lists

**Problem statement**

A linked list is represented by the following structure:

```
struct Node
{
        int data;
        struct Node* next;
};
```

You are given a function,

`struct Node* UncommonNodes(struct Node* list1, struct Node* list2);`

The function accepts the head nodes of 2 linked lists 'list1' and 'list2' as its argument. Implement the function to find the set difference of given 2 lists by modifying any of the given list such that it contains all elements of 'list1' that are not present in 'list2'.

**Set Difference:** Set difference of A and B is a set of all elements of A that are not present in B.

**Note:**

- For any element of 'list2' delete all of its occurrences from 'list1'.
- Preserve the sequence of elements of linked list.
- Do not use extra memory.
- Return null(or None in the case of python) if 'list1' is null(or None in the case of python) or all elements of 'list1' are present in 'list2'.

**Example:**

**Input:**

list1: 1 -> 2 -> 5 -> 6 -> 3

list2: 4 -> 5 -> 3 -> 8

**Output:**

1 -> 2 -> 6

**Explanation:**

After removing common elements (5, 3) from 'list1' output becomes 1 -> 2 -> 6.

**The custom input format for the above case:**

5 4

1 2 5 6 3

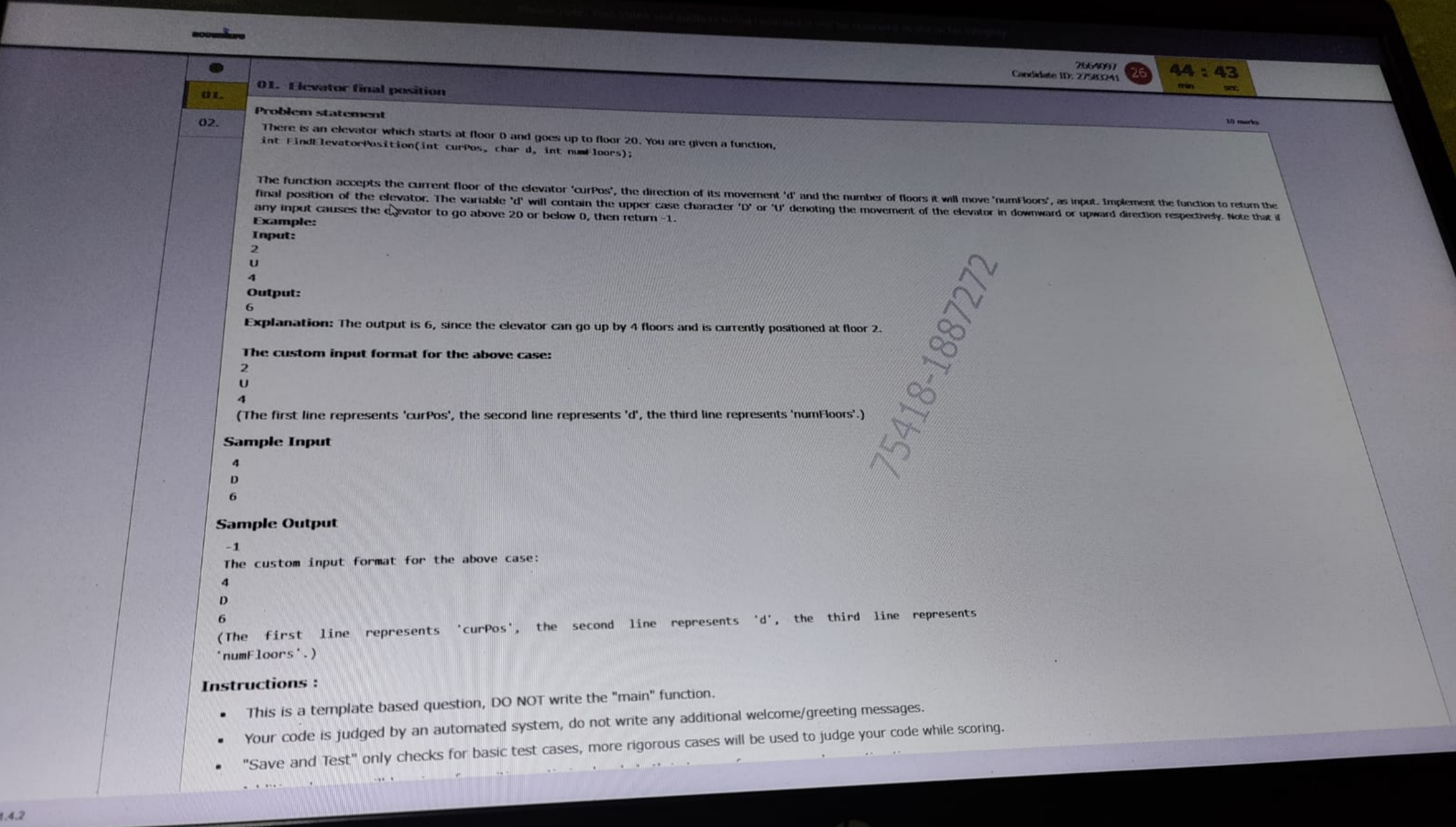
4 5 3 8

(The first line represents the number of nodes in the first and the second linked list respectively, the second line represents the nodes of the first linked list, the third line represents the nodes of the second linked list)

**Sample Input**

```
list1: 6 -> 8 -> 7
list2: 14 -> 9 -> 2 -> 9 -> 6
```

**Sample Output**



## 01. Elevator final position

**Problem statement**

10 marks

There is an elevator which starts at floor 0 and goes up to floor 20. You are given a function,

```
int FindElevatorPosition(int curPos, char d, int numFloors);
```

The function accepts the current floor of the elevator 'curPos', the direction of its movement 'd' and the number of floors it will move 'numFloors', as input. Implement the function to return the final position of the elevator. The variable 'd' will contain the upper case character 'D' or 'U' denoting the movement of the elevator in downward or upward direction respectively. Note that if any input causes the elevator to go above 20 or below 0, then return -1.

**Example:**

**Input:**

```
2
U
4
```

**Output:**

```
6
```

**Explanation:** The output is 6, since the elevator can go up by 4 floors and is currently positioned at floor 2.

**The custom input format for the above case:**

```
2
U
4
```

(The first line represents 'curPos', the second line represents 'd', the third line represents 'numFloors'.)

**Sample Input**

```
4
D
6
```

**Sample Output**

```
-1
```

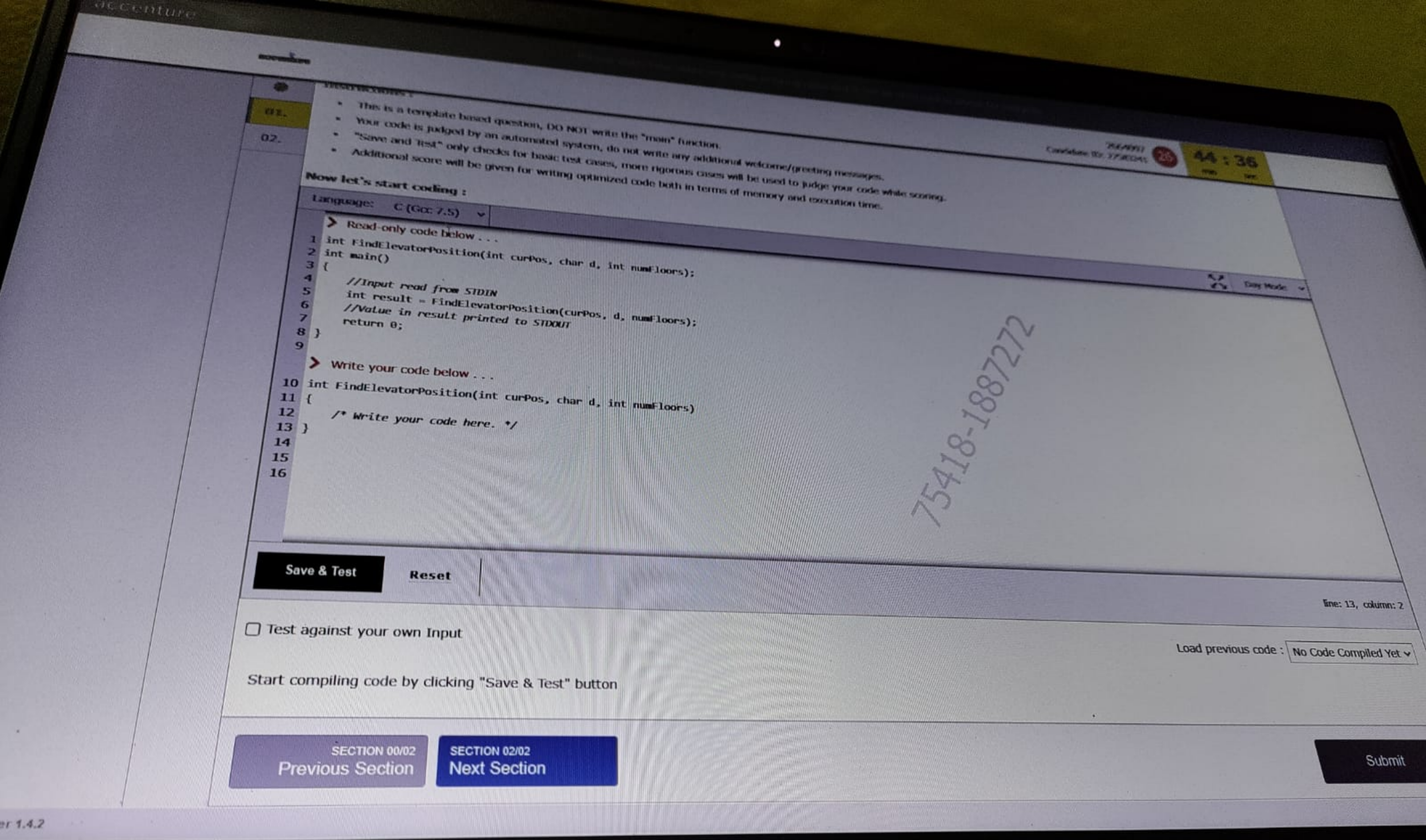
The custom input format for the above case:

```
4
D
6
```

(The first line represents 'curPos', the second line represents 'd', the third line represents 'numFloors'.)

**Instructions :**

- This is a template based question, DO NOT write the "main" function.
- Your code is judged by an automated system, do not write any additional welcome/greeting messages.
- "Save and Test" only checks for basic test cases, more rigorous cases will be used to judge your code while scoring.



- This is a template based question, DO NOT write the "main" function.
- Your code is judged by an automated system, do not write any additional welcome/greeting messages.
- "Save and Test" only checks for basic test cases, more rigorous cases will be used to judge your code while scoring.
- Additional score will be given for writing optimized code both in terms of memory and execution time.

**Now let's start coding :**

Language: C (Gcc 7.5)

Read-only code below . . .

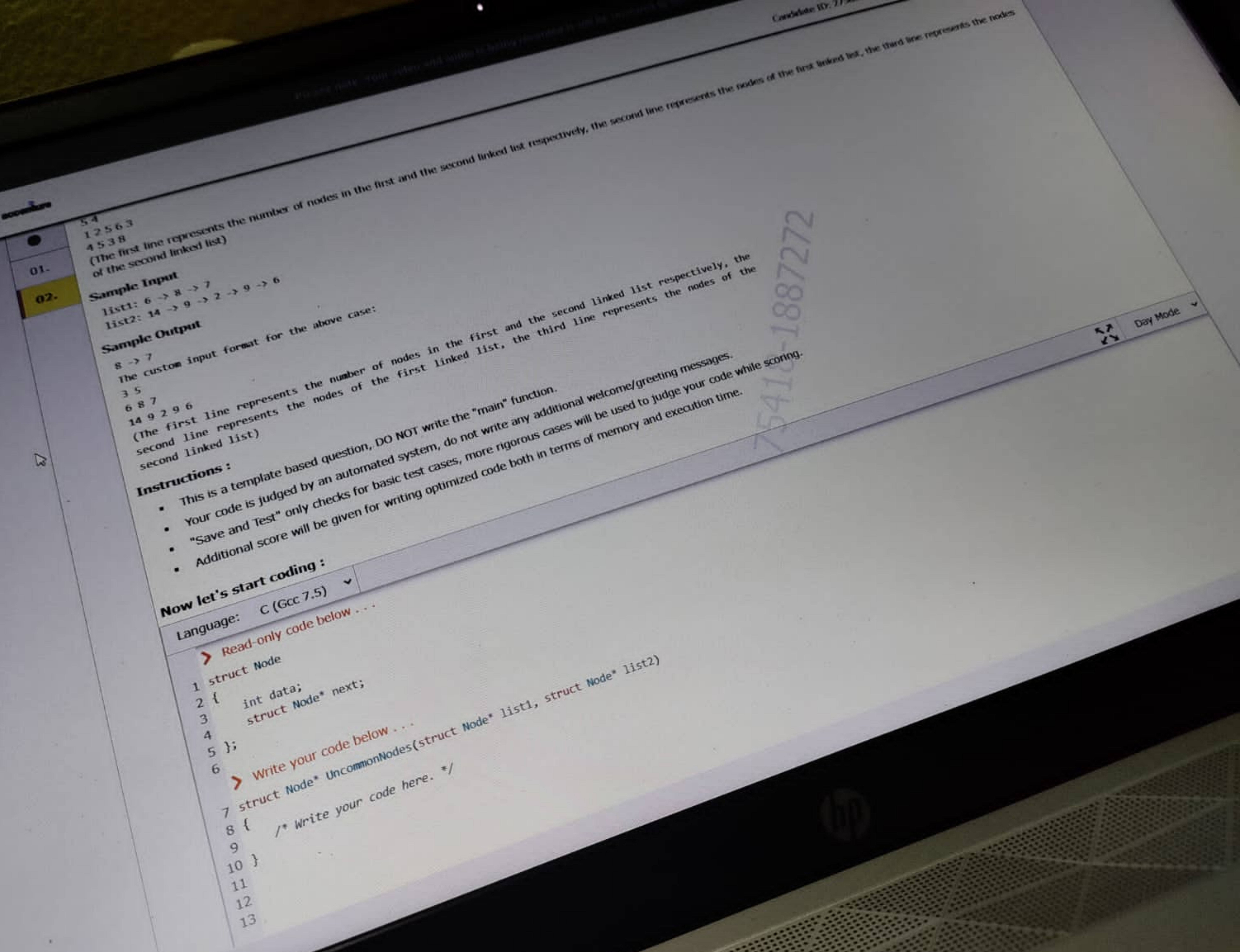
```c
int FindElevatorPosition(int curPos, char d, int numFloors);
int main()
{
    //Input read from STDIN
    int result = FindElevatorPosition(curPos, d, numFloors);
    //Value in result printed to STDOUT
    return 0;
}
```

Write your code below . . .

```c
int FindElevatorPosition(int curPos, char d, int numFloors)
{
    /* Write your code here. */
}
```

01.
02.
44 : 36
Day Mode
Save & Test
Reset
line: 13, column: 2
Test against your own Input
Load previous code : No Code Compiled Yet
SECTION 00/02 Previous Section
SECTION 02/02 Next Section
Submit
er 1.4.2

Start compiling code by clicking "Save & Test" button



Candidate ID: 27

01.

02.

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

(The first line represents the number of nodes in the first and the second linked list respectively, the second line represents the nodes of the first linked list, the third line represents the nodes
of the second linked list)

**Sample Input**

```
list1: 6 -> 8 -> 7
list2: 14 -> 9 -> 2 -> 9 -> 6
```

**Sample Output**

```
8 -> 7
The custom input format for the above case:
3 5
6 8 7
14 9 2 9 6
(The first line represents the number of nodes in the first and the second linked list respectively, the
second line represents the nodes of the first linked list, the third line represents the nodes of the
second linked list)
```

**Instructions :**

- This is a template based question, DO NOT write the "main" function.
- Your code is judged by an automated system, do not write any additional welcome/greeting messages.
- "Save and Test" only checks for basic test cases, more rigorous cases will be used to judge your code while scoring.
- Additional score will be given for writing optimized code both in terms of memory and execution time.

**Now let's start coding :**

Language: C (Gcc 7.5)

Read-only code below . . .

```c
struct Node
{
    int data;
    struct Node* next;
};
```

Write your code below . . .

```c
struct Node* UncommonNodes(struct Node* list1, struct Node* list2)
{
    /* Write your code here. */
}
```