**341. Flatten Nested List Iterator: -**

Medium Accepted: 384.2K Submissions: 614.7K Acceptance Rate: 62.5%

You are given a nested list of integers nestedList. Each element is either an integer or a list whose elements may also be integers or other lists. Implement an iterator to flatten it.

Implement the NestedIterator class:

* NestedIterator(List<NestedInteger> nestedList) Initializes the iterator with the nested list nestedList.
* int next() Returns the next integer in the nested list.
* boolean hasNext() Returns true if there are still some integers in the nested list and false otherwise.

Your code will be tested with the following pseudocode:

initialize iterator with nestedList

res = []

while iterator.hasNext()

append iterator.next() to the end of res

return res

If res matches the expected flattened list, then your code will be judged as correct.

**Example 1:**

**Input:** nestedList = [[1,1],2,[1,1]]

**Output:** [1,1,2,1,1]

**Explanation:** By calling next repeatedly until hasNext returns false, the order of elements returned by next should be: [1,1,2,1,1].

**Example 2:**

**Input:** nestedList = [1,[4,[6]]]

**Output:** [1,4,6]

**Explanation:** By calling next repeatedly until hasNext returns false, the order of elements returned by next should be: [1,4,6].

**Constraints:**

* 1 <= nestedList.length <= 500
* The values of the integers in the nested list is in the range [-106, 106].

**Code: -**

/\*\*

 \* // This is the interface that allows for creating nested lists.

 \* // You should not implement it, or speculate about its implementation

 \* class NestedInteger {

 \*   public:

 \*     // Return true if this NestedInteger holds a single integer, rather than a nested list.

 \*     bool isInteger() const;

 \*

 \*     // Return the single integer that this NestedInteger holds, if it holds a single integer

 \*     // The result is undefined if this NestedInteger holds a nested list

 \*     int getInteger() const;

 \*

 \*     // Return the nested list that this NestedInteger holds, if it holds a nested list

 \*     // The result is undefined if this NestedInteger holds a single integer

 \*     const vector<NestedInteger> &getList() const;

 \* };

 \*/

class NestedIterator {

public:

    vector<int> arr;

    int ind;

    void helper(vector<NestedInteger> &list){

        for(auto &v : list){

            if(v.isInteger()){

                arr.push\_back(v.getInteger());

                ++ind;

            }

            else{

                helper(v.getList());

                ++ind;

            }

        }

    }

    NestedIterator(vector<NestedInteger> &list) {

        ind = 0;

        for(auto &v : list){

            if(v.isInteger()){

                arr.push\_back(v.getInteger());

                ++ind;

            }

            else{

                helper(v.getList());

            }

        }

        ind = -1;

    }

    int next() {

        ++ind;

        return arr[ind];

    }

    bool hasNext() {

        return ind + 1 < arr.size();

    }

};

/\*\*

 \* Your NestedIterator object will be instantiated and called as such:

 \* NestedIterator i(nestedList);

 \* while (i.hasNext()) cout << i.next();

 \*/

**T.C: - O(N)**

**S.C: - O(N)**

**N = count of total integer in nestedlist**