Stacks

1. **Stack**

**Send Feedback**

A stack is a ?

1. FIFO(First In First Out)
2. LIFO(Last In Last Out) answer
3. Ordered Array
4. Linear Tree
5. **Use Recursion**

**Send Feedback**

Which data structure is used for implementing recursion?

1. Queue
2. Stack answer
3. Array
4. List
5. **Code : Stack Using LL**

**Send Feedback**

#### Implement a Stack Data Structure specifically to store integer data using a Singly Linked List.

#### The data members should be private.

#### You need to implement the following public functions :

##### 1. Constructor:

#### It initialises the data members as required.

##### 2. push(data) :

#### This function should take one argument of type integer. It pushes the element into the stack and returns nothing.

##### 3. pop() :

#### It pops the element from the top of the stack and in turn, returns the element being popped or deleted. In case the stack is empty, it returns -1.

##### 4. top :

#### It returns the element being kept at the top of the stack. In case the stack is empty, it returns -1.

##### 5. size() :

#### It returns the size of the stack at any given instance of time.

##### 6. isEmpty() :

#### It returns a boolean value indicating whether the stack is empty or not.

##### Operations Performed on the Stack:

Query-1(Denoted by an integer 1): Pushes an integer data to the stack.

Query-2(Denoted by an integer 2): Pops the data kept at the top of the stack and returns it to the caller.

Query-3(Denoted by an integer 3): Fetches and returns the data being kept at the top of the stack but doesn't remove it, unlike the pop function.

Query-4(Denoted by an integer 4): Returns the current size of the stack.

Query-5(Denoted by an integer 5): Returns a boolean value denoting whether the stack is empty or not.

##### Input Format:

The first line contains an integer 'q' which denotes the number of queries to be run against each operation in the stack.

Then the test cases follow.

Every 'q' lines represent an operation that needs to be performed.

For the push operation, the input line will contain two integers separated by a single space, representing the type of the operation in integer and the integer data being pushed into the stack.

For the rest of the operations on the stack, the input line will contain only one integer value, representing the query being performed on the stack.

##### Output Format:

For Query-1, you do not need to return anything.

For Query-2, prints the data being popped from the stack.

For Query-3, prints the data kept on the top of the stack.

For Query-4, prints the current size of the stack.

For Query-5, prints 'true' or 'false'(without quotes).

Output for every query will be printed in a separate line.

##### Note:

You are not required to print anything explicitly. It has already been taken care of. Just implement the function.

##### Constraints:

1 <= q <= 10^5

1 <= x <= 5

-2^31 <= data <= 2^31 - 1 and data != -1

Where 'q' is the total number of queries being performed on the stack, 'x' is the range for every query and data represents the integer pushed into the stack.

Time Limit: 1 second

##### Sample Input 1:

6

1 13

1 47

4

5

2

3

##### Sample Output 1:

2

false

47

13

##### Sample Input 2:

4

5

2

1 10

5

##### Sample Output 2:

true

-1

false

##### Explanation of Sample Input 2:

There are 4 queries in total.

The first one is Query-5: It tells whether the stack is empty or not. Since the stack is empty at this point, the output is 'true'.

The second one is Query-2: It pops the data from the stack. Since at this point in time, no data exist in the stack hence, it prints -1.

The third one is Query-1: It pushes the specified data 10 into the stack and since the function doesn't return anything, nothing is printed.

The fourth one is Query-5: It tells whether the stack is empty at this point or not. Since the stack has one element and hence it is not empty, false is printed.

1. **Stack Implementation**

**Send Feedback**

Which of the following operation take worst case i.e. linear time (O(n)) in the array implementation of stack?

1. Push
2. Pop
3. isEmpty
4. None answer
5. **Balanced Paranthesis**

**Send Feedback**

#### For a given a string expression containing only round brackets or parentheses, check if they are balanced or not. Brackets are said to be balanced if the bracket which opens last, closes first.

##### Example:

Expression: (()())

Since all the opening brackets have their corresponding closing brackets, we say it is balanced and hence the output will be, 'true'.

#### You need to return a boolean value indicating whether the expression is balanced or not.

##### Note:

The input expression will not contain spaces in between.

##### Input Format:

The first and the only line of input contains a string expression without any spaces in between.

##### Output Format:

The only line of output prints 'true' or 'false'.

##### Note:

You don't have to print anything explicitly. It has been taken care of. Just implement the function.

##### Constraints:

1 <= N <= 10^7

Where N is the length of the expression.

Time Limit: 1sec

##### Sample Input 1 :

(()()())

##### Sample Output 1 :

true

##### Sample Input 2 :

()()(()

##### Sample Output 2 :

false

##### Explanation to Sample Input 2:

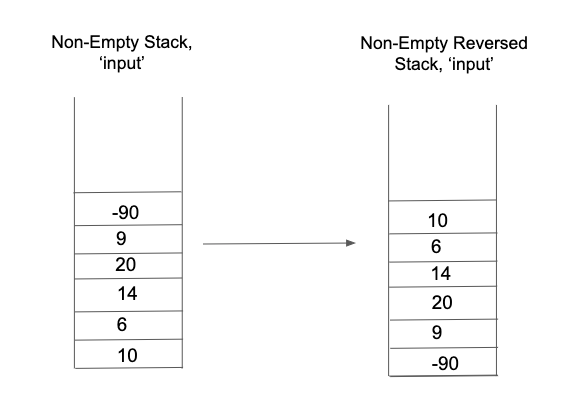
The initial two pairs of brackets are balanced. But when you see, the opening bracket at the fourth index doesn't have its corresponding closing bracket which makes it imbalanced and in turn, making the whole expression imbalanced. Hence the output prints 'false'.

1. **Reverse Stack**

**Send Feedback**

#### You have been given two stacks that can store integers as the data. Out of the two given stacks, one is populated and the other one is empty. You are required to write a function that reverses the populated stack using the one which is empty.

##### For Example:



##### Input Format :

The first line of input contains an integer N, denoting the total number of elements in the stack.

The second line of input contains N integers separated by a single space, representing the order in which the elements are pushed into the stack.

##### Output Format:

The only line of output prints the order in which the stack elements are popped, all of them separated by a single space.

##### Note:

You are not required to print the expected output explicitly, it has already been taken care of. Just make the changes in the input stack itself.

##### Constraints:

1 <= N <= 10^3

-2^31 <= data <= 2^31 - 1

Time Limit: 1sec

##### Sample Input 1:

6

1 2 3 4 5 10

##### Note:

Here, 10 is at the top of the stack.

##### Sample Output 1:

1 2 3 4 5 10

##### Note:

Here, 1 is at the top of the stack.

##### Sample Input 2:

5

2 8 15 1 10

##### Sample Output 2:

2 8 15 1 10

Assignment

1. **Check redundant brackets**

**Send Feedback**

#### For a given expression in the form of a string, find if there exist any redundant brackets or not. It is given that the expression contains only rounded brackets or parenthesis and the input expression will always be balanced.

#### A pair of the bracket is said to be redundant when a sub-expression is surrounded by unnecessary or needless brackets.

##### Example:

Expression: (a+b)+c

Since there are no needless brackets, hence, the output must be 'false'.

Expression: ((a+b))

The expression can be reduced to (a+b). Hence the expression has redundant brackets and the output will be 'true'.

##### Note:

You will not get a partial score for this problem. You will get marks only if all the test cases are passed.

##### Input Format :

The first and the only line of input contains a string expression, without any spaces in between.

##### Output Format :

The first and the only line of output will print either 'true' or 'false'(without the quotes) denoting whether the input expression contains redundant brackets or not.

##### Note:

You are not required to print the expected result. It has already been taken care of.

##### Constraints:

0 <= N <= 10^6

Where N is the length of the expression.

Time Limit: 1 second

##### Sample Input 1:

a+(b)+c

##### Sample Output 1:

true

##### Explanation:

The expression can be reduced to a+b+c. Hence, the brackets are redundant.

##### Sample Input 2:

(a+b)

##### Sample Output 2:

false

1. **Stock Span**

**Send Feedback**

#### Afzal has been working with an organization called 'Money Traders' for the past few years. The organization is into the money trading business. His manager assigned him a task. For a given array/list of stock's prices for N days, find the stock's span for each day.

#### The span of the stock's price today is defined as the maximum number of consecutive days(starting from today and going backwards) for which the price of the stock was less than today's price.

#### For example, if the price of a stock over a period of 7 days are [100, 80, 60, 70, 60, 75, 85], then the stock spans will be [1, 1, 1, 2, 1, 4, 6].

##### Explanation:

On the sixth day when the price of the stock was 75, the span came out to be 4, because the last 4 prices(including the current price of 75) were less than the current or the sixth day's price.

Similarly, we can deduce the remaining results.

#### Afzal has to return an array/list of spans corresponding to each day's stock's price. Help him to achieve the task.

##### Input Format:

The first line of input contains an integer N, denoting the total number of days.

The second line of input contains the stock prices of each day. A single space will separate them.

##### Output Format:

The only line of output will print the span for each day's stock price. A single space will separate them.

##### Note:

You are not required to print the expected output explicitly. It has already been taken care of.

##### Constraints:

0 <= N <= 10^7

1 <= X <= 10^9

Where X denotes the stock's price for a day.

Time Limit: 1 second

##### Sample Input 1:

4

10 10 10 10

##### Sample Output 1:

1 1 1 1

##### Sample Input 2:

8

60 70 80 100 90 75 80 120

##### Sample Output 2:

1 2 3 4 1 1 2 8

1. **Minimum bracket Reversal**

**Send Feedback**

#### For a given expression in the form of a string, find the minimum number of brackets that can be reversed in order to make the expression balanced. The expression will only contain curly brackets.

#### If the expression can't be balanced, return -1.

##### Example:

Expression: {{{{

If we reverse the second and the fourth opening brackets, the whole expression will get balanced. Since we have to reverse two brackets to make the expression balanced, the expected output will be 2.

Expression: {{{

In this example, even if we reverse the last opening bracket, we would be left with the first opening bracket and hence will not be able to make the expression balanced and the output will be -1.

##### Input Format :

The first and the only line of input contains a string expression, without any spaces in between.

##### Output Format :

The only line of output will print the number of reversals required to balance the whole expression. Prints -1, otherwise.

##### Note:

You don't have to print anything. It has already been taken care of.

##### Constraints:

0 <= N <= 10^6

Where N is the length of the expression.

Time Limit: 1sec

##### Sample Input 1:

{{{

##### Sample Output 1:

-1

##### Sample Input 2:

{{{{}}

##### Sample Output 2:

1