1. <https://www.naukri.com/code360/problems/level-order-traversal_796002>

You have been given a Binary Tree of integers. You are supposed to return the level order traversal of the given tree.

The level order traversal will be {1,2,3,4,5,6,7}

vector<int> getLevelOrder(BinaryTreeNode<int> \*root)

{  vector<int>v;

queue<BinaryTreeNode<int>\*>q;

   if(root!=NULL){

q.push(root);

   }

    while(!q.empty()){

        BinaryTreeNode<int>\* f=q.front();

        q.pop();

        v.push\_back(f->val);

        if(f->left!=NULL){

            q.push(f->left);

        }

        if(f->right!=NULL){

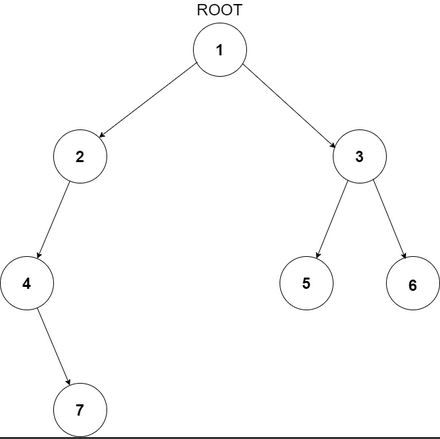
            q.push(f->right);

        }

    }

    return v;

}

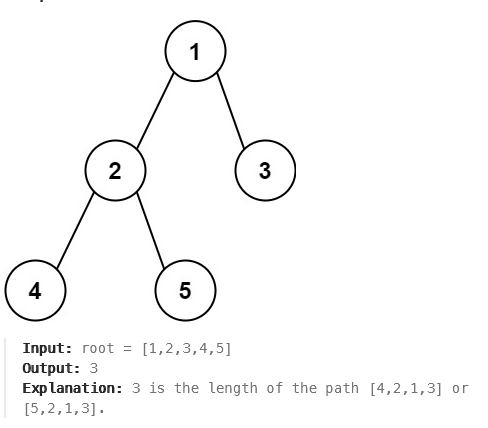


1. <https://leetcode.com/problems/diameter-of-binary-tree/description/>

Given the root of a binary tree, return the length of the ***diameter*** of the tree.

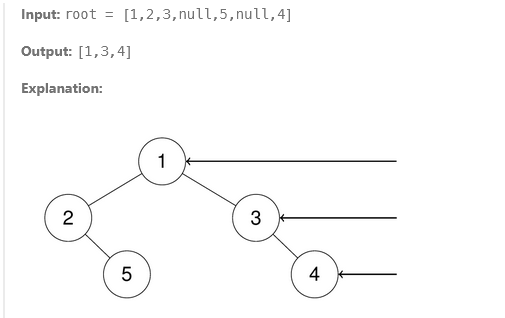
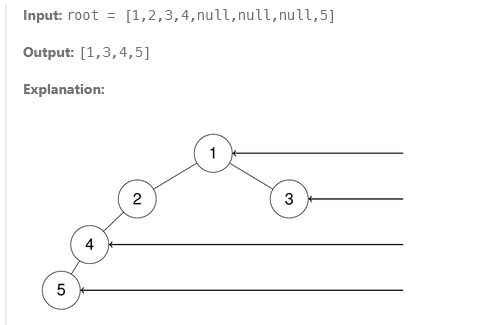
The **diameter** of a binary tree is the **length** of the longest path between any two nodes in a tree. This path may or may not pass through the root.

The **length** of a path between two nodes is represented by the number of edges between them.



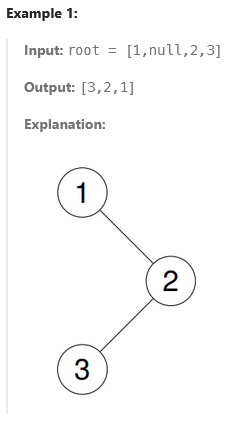
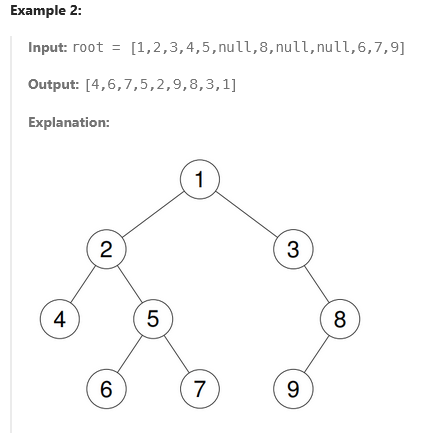
1. <https://leetcode.com/problems/binary-tree-right-side-view/description/>

Given the root of a binary tree, imagine yourself standing on the **right side** of it, return the values of the nodes you can see ordered from top to bottom.

1. <https://leetcode.com/problems/binary-tree-postorder-traversal/description/>

Given the root of a binary tree, return the postorder traversal of its nodes' values

class Solution {

public:

void post(TreeNode \*root,vector<int>& v){

if(root==NULL){

return ;

}

post(root->left,v);

post(root->right,v);

v.push\_back(root->val);

}

vector<int> postorderTraversal(TreeNode\* root) {

vector<int>v;

post(root, v);

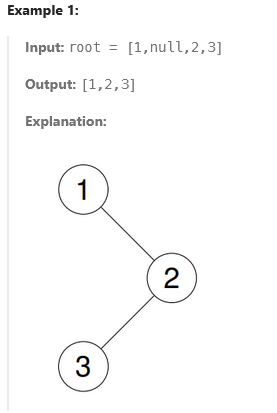
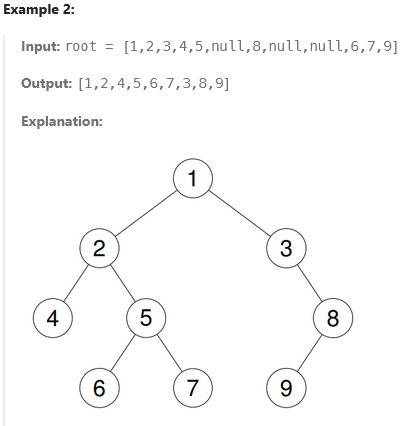
return v;

}

};

1. <https://leetcode.com/problems/binary-tree-preorder-traversal/description/>

Given the root of a binary tree, return the preorder traversal of its nodes' values.

class Solution {

public:

void pre(TreeNode \* root,vector<int>&v){//& na dile v er change ta change hobe na

if(root==NULL){

return;

}

v.push\_back(root->val);

pre(root->left,v);

pre(root->right,v);

}

vector<int> preorderTraversal(TreeNode\* root) {

vector<int>v;

pre(root,v);

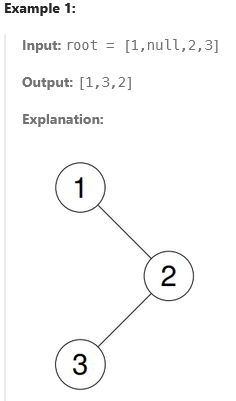
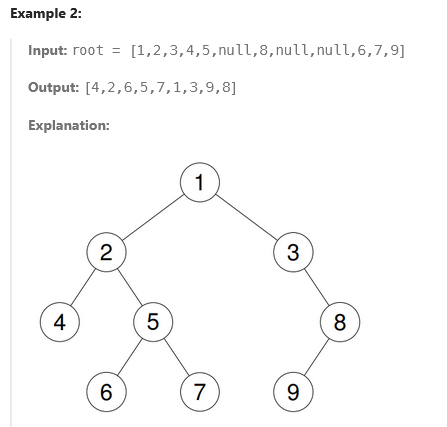
return v;

}

};

1. <https://leetcode.com/problems/binary-tree-inorder-traversal/description/>

Given the root of a binary tree, return the inorder traversal of its nodes' values.

class Solution {

public:

void in(TreeNode \*root,vector<int>& v){

if(root==NULL){

return ;

}

in(root->left,v);

v.push\_back(root->val);

in(root->right,v);

}

vector<int> inorderTraversal(TreeNode\* root) {

vector<int>v;

in(root, v);

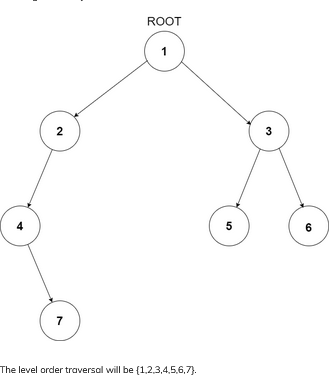
return v;

}

};

1. <https://www.naukri.com/code360/problems/level-order-traversal_796002>

You have been given a Binary Tree of integers. You are supposed to return the level order traversal of the given tree.



vector<int> getLevelOrder(BinaryTreeNode<int> \*root)

{  vector<int>v;

queue<BinaryTreeNode<int>\*>q;

   if(root!=NULL){

q.push(root);

   }

    while(!q.empty()){

        BinaryTreeNode<int>\* f=q.front();

        q.pop();

        v.push\_back(f->val);

        if(f->left!=NULL){

            q.push(f->left);

        }

        if(f->right!=NULL){

            q.push(f->right);

        }

    }

    return v;

}

1. <https://www.naukri.com/code360/problems/count-leaf-nodes_893055>

You are given a Binary tree. You have to count and return the number of leaf nodes present in it.

A binary tree is a tree data structure in which each node has at most two children, which are referred to as the left child and the right child

A node is a leaf node if both left and right child nodes of it are NULL.

int noOfLeafNodes(BinaryTreeNode<int> \*root){

    if(root==NULL){

        return 0;

    }

    if(root->left==NULL && root->right==NULL){

        return 1;

    }

    int l=noOfLeafNodes(root->left);

    int r=noOfLeafNodes(root->right);

    return l+r;

}

##### Sample Input 1:

3

1 -1 -1

1 2 3 -1 -1 -1 4 -1 -1

1 2 -1 -1 3 -1 4 -1 -1

##### Sample Output 1:

1

2

1

9. <https://www.naukri.com/code360/problems/left-sum_920380>

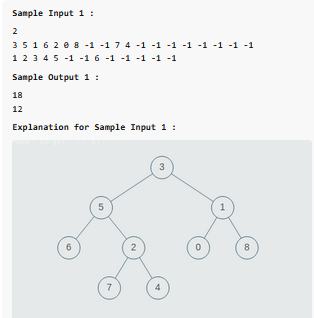
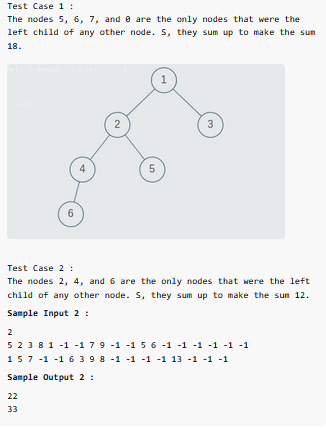
Given a binary tree having ‘N’ number of nodes. Your task is to find the sum of all left nodes present in the input binary tree. That is, you need to take the sum of all nodes which are the left child of some node.

Note :

1. A binary tree is a tree in which each node can have at most two children.

2. The given tree will be non-empty i.e the number of non-NULL nodes will always be greater than or equal to 1.

3. Multiple nodes in the tree can have the same values, all values in the tree will be positive.

long long leftSum(BinaryTreeNode<int> \*root)

{

    if(root==NULL){

        return 0;

    }

  else if(root->left){

        return root->left->data +leftSum(root->left)+leftSum(root->right);

    }

    else{

        return leftSum(root->right);

    }

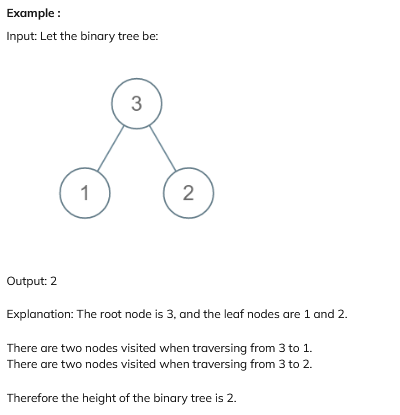
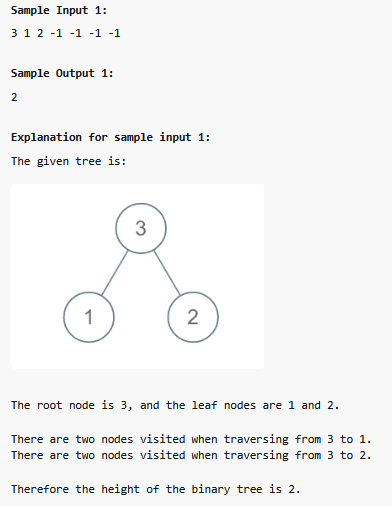
}

10.<https://www.naukri.com/code360/problems/tree-height_4609628>

The height of a tree is equal to the number of nodes on the longest path from the root to a leaf.

You are given an arbitrary binary tree consisting of ***'n'*** nodes where each node is associated with a certain value.

Find out the height of the tree.

int heightOfBinaryTree(TreeNode<int> \*root)

{

    if(root==NULL){

        return 0;

    }

    if(root->left==NULL && root->right==NULL){

        return 1;

    }

    return max(heightOfBinaryTree(root->left),heightOfBinaryTree(root->right))+1;

}

11. <https://www.naukri.com/code360/problems/code-find-a-node_5682>

For a given Binary Tree of type integer and a number X, find whether a node exists in the tree with data X or not

bool isNodePresent(BinaryTreeNode<int> \*root, int x) {

    queue<BinaryTreeNode<int> \*>q;

    if(root==NULL){

        return false;

    }

    q.push(root);

    if(root->data==x){

        return true;

    }

    while(!q.empty()){

        BinaryTreeNode<int> \* f=q.front();

        q.pop();

        if(f->data==x){

            return true;

        }

        if(f->left){

            q.push(f->left);

        }

        if(f->right){

            q.push(f->right);

        }

    }

    return false;

}

##### Sample Input 1:

8 3 10 1 6 -1 14 -1 -1 4 7 13 -1 -1

-1 -1 -1 -1 -1

7

##### Sample Output 1:

true

##### Explanation For Output 1:

Clearly, we can see that 7 is present in

the tree. So, the output will be true.

##### Sample Input 2:

2 3 4 -1 -1 -1 -1

10

##### Sample Output 2:

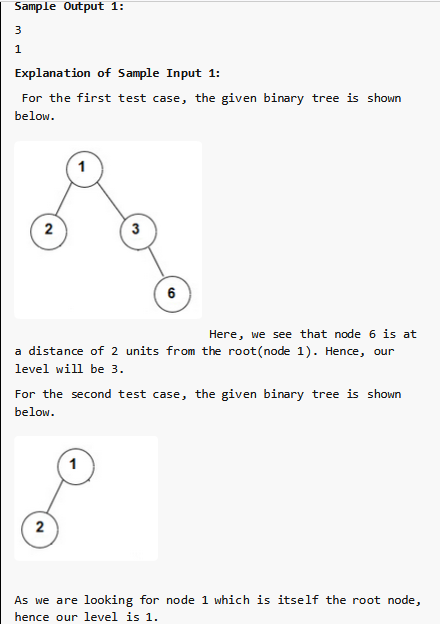
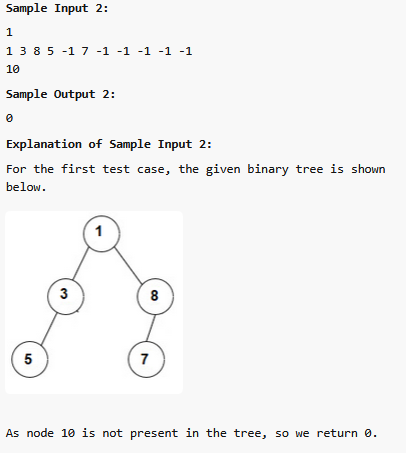
False

12. <https://www.naukri.com/code360/problems/node-level_920383>

Ninja has been given a binary tree having N nodes and an integer M, he needs to find the level of node M. He finds it difficult to solve and asks for your help.

Note:

Consider root to be at level 1. It is guaranteed that all the nodes in the binary tree have distinct values.

int nodeLevel(TreeNode<int>\* root, int searchedValue)

{

    if(root==NULL){

        return 0;

    }

    int c=0;

    int n;

    queue<TreeNode<int>\*>q;

    q.push(root);

    while(!q.empty()){

        c++;

        n=q.size();

        for(int i=0;i<n;i++){

            TreeNode<int>\* f=q.front();

            if(f->val==searchedValue){

                return c;

            }

            q.pop();

            if(f->left){

                q.push(f->left);

            }

            if(f->right){

                q.push(f->right);

            }

        }

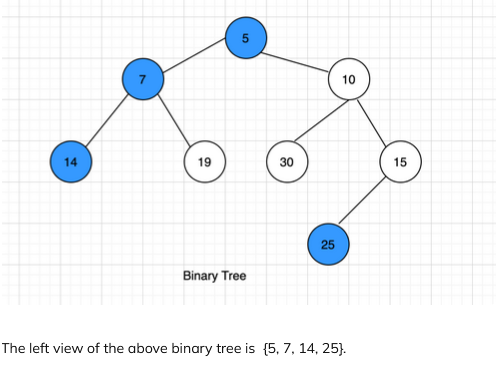
    }

    return 0;

}

13. <https://www.naukri.com/code360/problems/left-view-of-a-binary-tree_920519>

You have been given a binary tree of integers. You are supposed to find the left view of the binary tree. The left view of a binary tree is the set of all nodes that are visible when the binary tree is viewed from the left side



vector<int> getLeftView(TreeNode<int> \*root)

{

   queue<TreeNode<int> \*>q;

   vector<int>v;

   if(root==NULL){

       return v;

   }

   int c=0;

   int n;

   q.push(root);

   while(!q.empty()){

       c++;

       n=q.size();

       for(int i=0;i<n;i++){

           TreeNode<int> \* f=q.front();

           if(i==0){

               v.push\_back(f->data);

           }

           q.pop();

           if(f->left){

               q.push(f->left);

           }

           if(f->right){

               q.push(f->right);

           }

       }

   }

   return v;

}

##### Sample Input 1:

2

3 4 -1 -1 -1

2 8 7 -1 5 -1 -1 1 -1 -1 -1

##### Sample Output 1:

3 4

2 8 5 1

##### Explanation of Sample Input 1:

For the first test case, node 3 and node 4 are

visible when the binary tree is viewed from the left.

For the second test case, nodes 2, 8, 5, 1 are visible

when the binary tree is viewed from the left.

##### Sample Input 2:

2

1 3 5 12 -1 8 -1 -1 -1 -1 -1

9 6 3 -1 -1 4 8 -1 -1 -1 -1

##### Sample Output 2:

1 3 12

9 6 4

##### Explanation of Sample Input 2:

For the first test case, nodes 1, 3, 12 are visible when

the binary tree is viewed from the left.

For the second test case, nodes 9, 6, 4 are

visible when the binary tree is viewed from the left.

14. <https://www.naukri.com/code360/problems/diameter-of-the-binary-tree_920552>

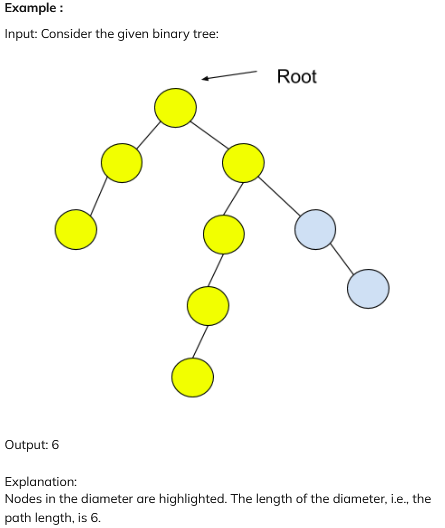
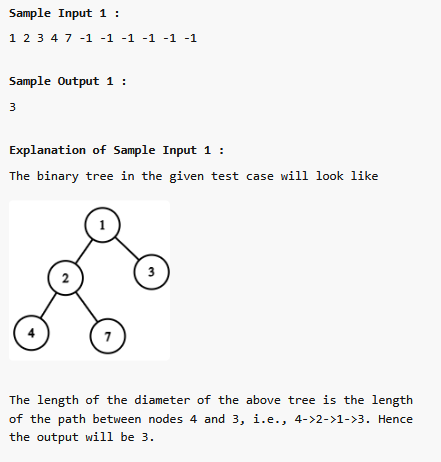
You are given a Binary Tree.

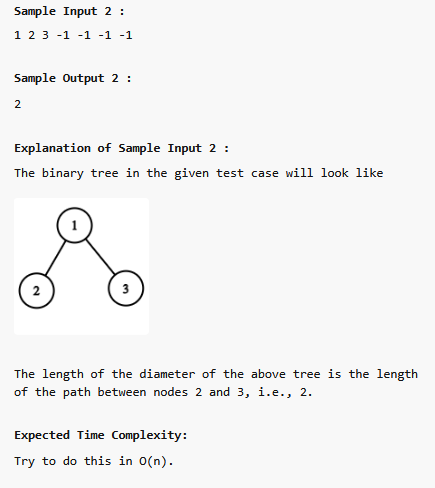
Return the length of the diameter of the tree.

**Note :**

The diameter of a binary tree is the length of the longest path between any two end nodes in a tree.

The number of edges between two nodes represents the length of the path between them



15. <https://www.naukri.com/code360/problems/special-binary-tree_920502?leftPanelTabValue=PROBLEM>

You are given an arbitrary binary tree. A binary tree is called special if every node of this tree has either zero or two children. You have to determine if the given binary tree is special or not.

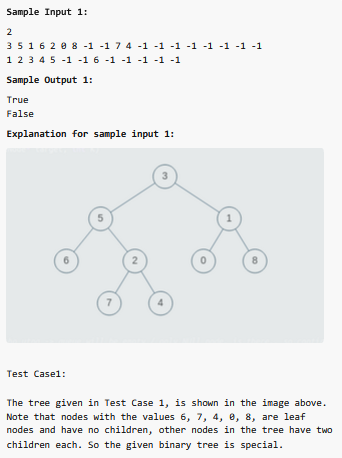
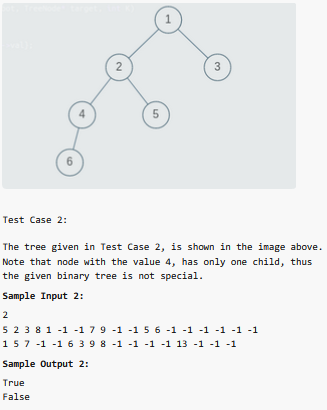
If the given binary tree is special, return True. Else, return False to the given function.

Note:

1. A binary tree is a tree in which each node can have at most two children.

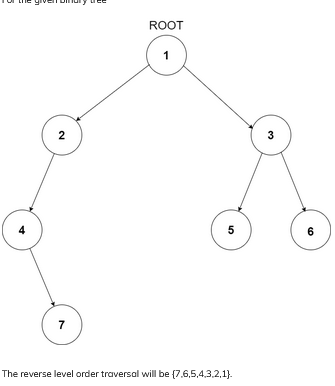
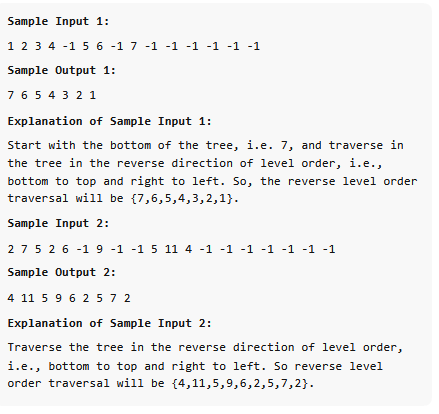
2. The given tree will be non-empty i.e the number of non-NULL nodes will always be greater than or equal to 1.

3. Multiple nodes in the tree can have the same values, all values in the tree will be positive.

16. <https://www.naukri.com/code360/problems/reverse-level-order-traversal_764339>

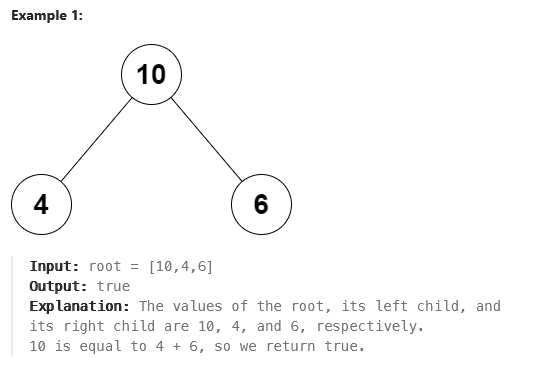
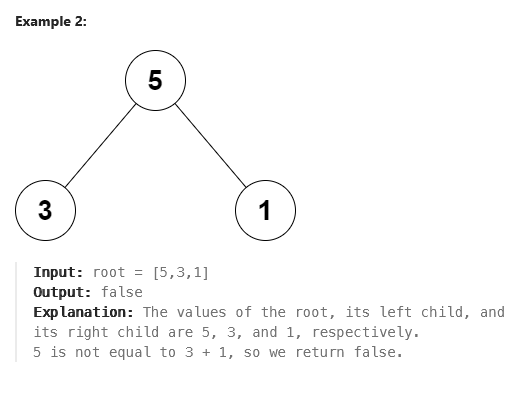
You have been given a Binary Tree of integers. You are supposed to return the reverse of the level order traversal

17. <https://leetcode.com/problems/root-equals-sum-of-children/description/>

You are given the root of a **binary tree** that consists of exactly 3 nodes: the root, its left child, and its right child.

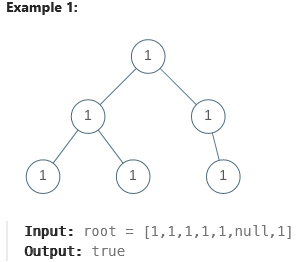
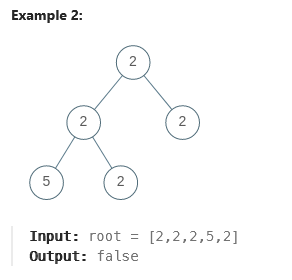
Return true if the value of the root is equal to the ***sum*** of the values of its two children, or false otherwise

18. <https://leetcode.com/problems/univalued-binary-tree/description/>

A binary tree is **uni-valued** if every node in the tree has the same value.

Given the root of a binary tree, return true if the given tree is ***uni-valued***, or false otherwise.

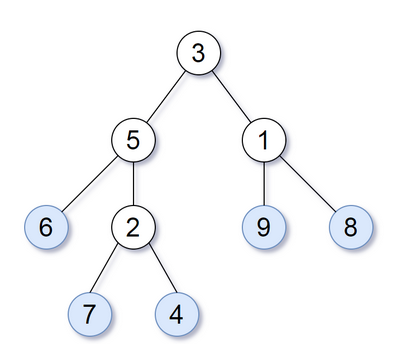
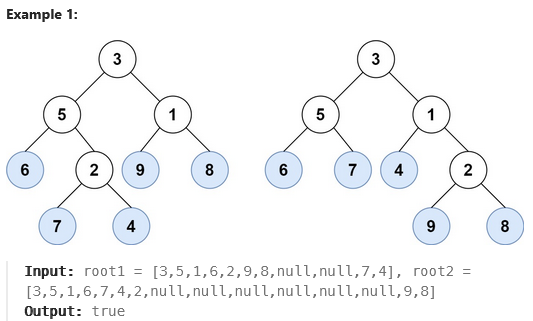
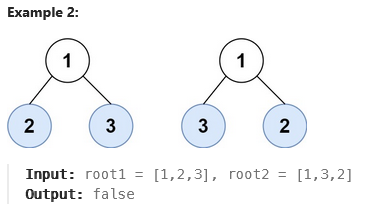
19. <https://leetcode.com/problems/leaf-similar-trees/description/>

Consider all the leaves of a binary tree, from left to right order, the values of those leaves form a **leaf value sequence**

For example, in the given tree above, the leaf value sequence is (6, 7, 4, 9, 8).

Two binary trees are considered leaf-similar if their leaf value sequence is the same.

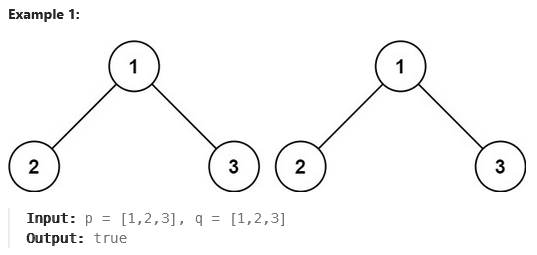
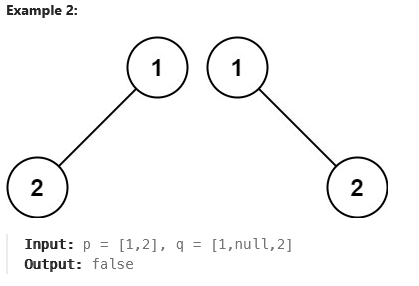
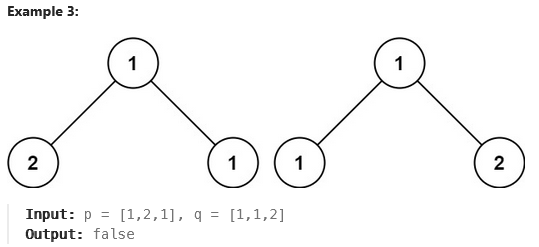
Return true if and only if the two given trees with head nodes root1 and root2 are leaf-similar.

20. <https://leetcode.com/problems/same-tree/description/>

Given the roots of two binary trees p and q, write a function to check if they are the same or not.

Two binary trees are considered the same if they are structurally identical, and the nodes have the same value.

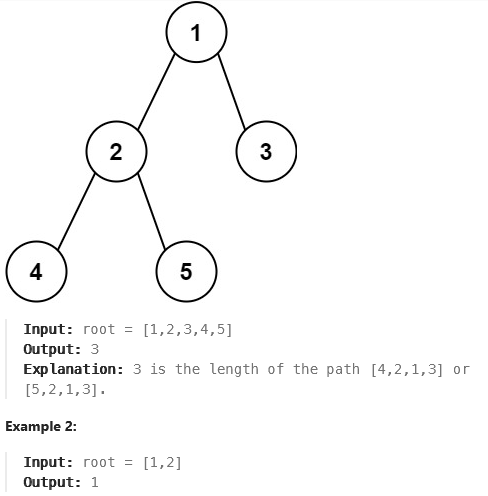
  

21. <https://leetcode.com/problems/diameter-of-binary-tree/description/>

Given the root of a binary tree, return the length of the ***diameter*** of the tree.

The **diameter** of a binary tree is the **length** of the longest path between any two nodes in a tree. This path may or may not pass through the root.

The **length** of a path between two nodes is represented by the number of edges between them.



22. <https://leetcode.com/problems/remove-all-adjacent-duplicates-in-string/description/>

You are given a string s consisting of lowercase English letters. A **duplicate removal** consists of choosing two **adjacent** and **equal** letters and removing them.

We repeatedly make **duplicate removals** on s until we no longer can.

Return the final string after all such duplicate removals have been made. It can be proven that the answer is **unique**.

**Example 1:**

**Input:** s = "abbaca"

**Output:** "ca"

**Explanation:**

For example, in "abbaca" we could remove "bb" since the letters are adjacent and equal, and this is the only possible move. The result of this move is that the string is "aaca", of which only "aa" is possible, so the final string is "ca".

**Example 2:**

**Input:** s = "azxxzy"

**Output:** "ay"

class Solution {

public:

string removeDuplicates(string s) {

stack<char>stk;

for(auto x:s){

if( !stk.empty() && x==stk.top() ){

stk.pop();

}

else{

stk.push(x);

}

}

string ans;

while(!stk.empty()) {

ans += stk.top();

stk.pop();

}

reverse(ans.begin(), ans.end());

return ans;

}

};

23. <https://leetcode.com/problems/implement-stack-using-queues/description/>

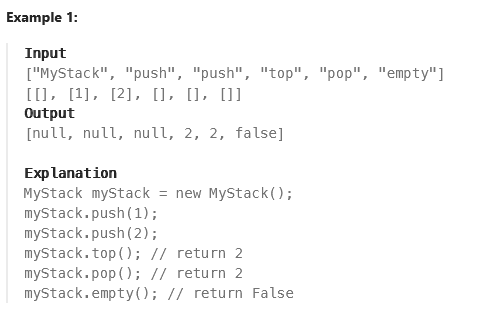
Implement a last-in-first-out (LIFO) stack using only two queues. The implemented stack should support all the functions of a normal stack (push, top, pop, and empty).

Implement the MyStack class:

* void push(int x) Pushes element x to the top of the stack.
* int pop() Removes the element on the top of the stack and returns it.
* int top() Returns the element on the top of the stack.
* boolean empty() Returns true if the stack is empty, false otherwise.

**Notes:**

* You must use **only** standard operations of a queue, which means that only push to back, peek/pop from front, size and is empty operations are valid.
* Depending on your language, the queue may not be supported natively. You may simulate a queue using a list or deque (double-ended queue) as long as you use only a queue's standard operations



class MyStack {

public:

queue<int>q;

MyStack() {

}

void push(int x) {

q.push(x);

}

int pop() {

queue<int>q2;

int val;

while(!q.empty()){

val=q.front();

q.pop();

if(q.empty()){

break;

}

q2.push(val); }

q=q2;

return val;

}

int top() {

return q.back();

}

bool empty() {

return q.empty();

}

};

\*/

View less







0/5

24. <https://leetcode.com/problems/implement-queue-using-stacks/>

Implement a first in first out (FIFO) queue using only two stacks. The implemented queue should support all the functions of a normal queue (push, peek, pop, and empty).

Implement the MyQueue class:

* void push(int x) Pushes element x to the back of the queue.
* int pop() Removes the element from the front of the queue and returns it.
* int peek() Returns the element at the front of the queue.
* boolean empty() Returns true if the queue is empty, false otherwise.

**Notes:**

* You must use **only** standard operations of a stack, which means only push to top, peek/pop from top, size, and is empty operations are valid.
* Depending on your language, the stack may not be supported natively. You may simulate a stack using a list or deque (double-ended queue) as long as you use only a stack's standard operations.

**Example 1:**

**Input**

["MyQueue", "push", "push", "peek", "pop", "empty"]

[[], [1], [2], [], [], []]

**Output**

[null, null, null, 1, 1, false]

**Explanation**

MyQueue myQueue = new MyQueue();

myQueue.push(1); // queue is: [1]

myQueue.push(2); // queue is: [1, 2] (leftmost is front of the queue)

myQueue.peek(); // return 1

myQueue.pop(); // return 1, queue is [2]

myQueue.empty(); // return false

class MyQueue {

public:

stack<int>s;

MyQueue() {

}

void push(int x) {

s.push(x);

}

int pop() {

stack<int>s1;

int v;

while(!s.empty()){

v=s.top();

s.pop();

if(s.empty()){

break;

}

s1.push(v);

}

while(!s1.empty()){

s.push(s1.top());

s1.pop();

}

return v;

}

int peek() {

stack<int>s1;

int v;

while(!s.empty()){

v=s.top();

s.pop();

if(s.empty()){

s1.push(v);

break;

}

s1.push(v);

}

while(!s1.empty()){

s.push(s1.top());

s1.pop();

}

return v;

}

bool empty() {

return s.empty();

}

};

25. <https://leetcode.com/problems/valid-parentheses/description/>

Given a string s containing just the characters '(', ')', '{', '}', '[' and ']', determine if the input string is valid.

An input string is valid if:

1. Open brackets must be closed by the same type of brackets.
2. Open brackets must be closed in the correct order.
3. Every close bracket has a corresponding open bracket of the same type.

class Solution {

public:

bool isValid(string s) {

stack<char>st;

int sz=s.size();

for(char c:s){

if(c=='(' || c=='{' || c=='['){

st.push(c);

}

else{

if (st.empty()) {

return false;

}

char topp=st.top();

st.pop();

if ((c == ')' && topp != '(') ||

(c == '}' && topp != '{') ||

(c == ']' && topp != '[') ){//!= dite hobe ex: "]" for this it will be invalid

return false;

}

}

}

if(st.empty()){

return true;

}

else{

return false;

}

}

};

**Example 1:**

**Input:** s = "()"

**Output:** true

**Example 2:**

**Input:** s = "()[]{}"

**Output:** true

**Example 3:**

**Input:** s = "(]"

**Output:** false

**Example 4:**

**Input:** s = "([])"

**Output:** true

26. <https://leetcode.com/problems/backspace-string-compare/description/>

Given two strings s and t, return true if they are equal when both are typed into empty text editors. '#' means a backspace character.

Note that after backspacing an empty text, the text will continue empty.

**Example 1:**

**Input:** s = "ab#c", t = "ad#c"

**Output:** true

**Explanation:** Both s and t become "ac".

**Example 2:**

**Input:** s = "ab##", t = "c#d#"

**Output:** true

**Explanation:** Both s and t become "".

**Example 3:**

**Input:** s = "a#c", t = "b"

**Output:** false

**Explanation:** s becomes "c" while t becomes "b".

class Solution {

public:

bool backspaceCompare(string s, string t) {

stack<char>s1,s2;

for(auto x:s){

if(x!='#'){

s1.push(x);

}

else{

if(!s1.empty())

s1.pop();

}

}

for(auto x:t){

if(x!='#'){

s2.push(x);

}

else{

if(!s2.empty())

s2.pop();

}

}

return s1==s2;

}

};

27. <https://www.naukri.com/code360/problems/insert-an-element-at-its-bottom-in-a-given-stack_1171166>

You are given a stack/deque of integers 'MY-STACK' and an integer ‘X’. Your task is to insert ‘X’ to the bottom of ‘MY-STACK’ and return the updated stack/deque.

**Note :**

If ‘MY\_STACK’ = [7, 1, 4, 5], then the first element represents the element at the bottom of the stack and the last element represents the element at the top of the stack.

**For Example :**

Let ‘MY\_STACK’ = [7, 1, 4, 5] and ‘X’ = 9. So, ‘MY\_STACK’ after insertion becomes [9, 7, 1, 4, 5].

#include <bits/stdc++.h>

stack<int> pushAtBottom(stack<int>& myStack, int x)

{

    stack<int>s2;

    while(!myStack.empty()){

       int v=myStack.top();

       myStack.pop();

       s2.push(v);

    }

    myStack.push(x);

    while(!s2.empty()){

       int v=s2.top();

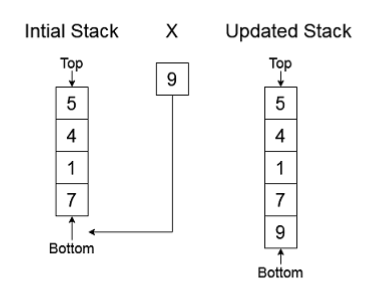
       s2.pop();

       myStack.push(v);

    }

    return myStack;

}



28. <https://www.naukri.com/code360/problems/maximum-equal-stack-sum_1062571>

Given three filled stacks namely ‘stack1’ ‘stack2’ and ‘stack3’ of positive numbers, the task is to find the possible equal maximum sum of the stacks with the removal of top elements allowed.

For example, let the stacks be:

We can see that currently,

the sum of stack 1 is: 8+5+3 = 16

the sum of stack 2 is: 2+2+4+9+6 = 23

the sum of stack 3 is: 2+1+2+3 = 8

So they are not equal.

However, if we pop {8} from stack 1, {6,9} from stack 2 and nothing from stack 3,

We get the sum as :

Stack 1: 16-8=8

Stack 2: 23-15=8

Stack 3: 8-0=8

We can see that now the sum of all three stacks are equal which is 8 and it is the highest possible, hence we return 8.

**Note:**

1. Do not print anything, just return an integer which is the maximum possible sum for the three stacks.

2.It is guaranteed that the elements in the stack are positive integers.

3.It can be proved that a non-negative integer answer always exists

#include <bits/stdc++.h>

int sum(stack<int>s){//\* na deya te just s er akta copy ekhane asbe

 int sm=0;

 while(!s.empty()){

     sm+=s.top();

     s.pop();

 }

 return sm;

}

int maxSum(stack<int> &stk1, stack<int> &stk2, stack<int> &stk3) {

    int s1=sum(stk1);

    int s2=sum(stk2);

    int s3=sum(stk3);

    while(!stk1.empty() && !stk2.empty() && !stk3.empty()){

        if(s1==s2 && s2==s3){

            return s1;

        }

        else if(s1<=s2 && s3<=s2){

            int v=stk2.top();

            s2=s2-v;

            stk2.pop();

        }

        else if(s2<=s3 && s1<=s3){

             int v=stk3.top();

            s3=s3-v;

            stk3.pop();

        }

        else{

             int v=stk1.top();

            s1=s1-v;

            stk1.pop();

        }

    }

    return 0;

}

29. <https://www.naukri.com/code360/problems/reversing-a-queue_982934>

You are given a queue of 'N' elements. Your task is to reverse the order of elements present in the queue.

You can only use the standard operations of the QUEUE STL.

1. enqueue(x): Add an item x to the rear of the queue.

2. dequeue(): Removes an item from the front of the queue.

3. size(): Returns the number of elements in the queue.

4. front(): Finds front element.

5. empty(): Checks whether the queue is empty or not

##### Sample Input 1:

2

1

9

5

10 6 8 12 3

##### Sample Output 1:

#include <bits/stdc++.h>

queue<int> reverseQueue(queue<int> q)

{

   stack<int>s;

   while(!q.empty()){

       int v=q.front();

       q.pop();

       s.push(v);

   }

  queue<int>q2;

  while(!s.empty()){

     int v=s.top();

       s.pop();

       q2.push(v);

  }

  return q2;

}

9

3 12 8 6 10

##### Explanation 1:

For the first test case, the queue

has only one element, i.e. 9.

So, even after reversing, our queue remains the same.

For the second test case, the queue has elements

in the order: 10, 6, 8, 12, 3. Reversing the queue

changes the order of elements to - 3, 12, 8, 6, 10.

##### Sample Input 2:

2

2

99 89

6

1 2 3 4 5 6

##### Sample Output 2:

89 99

6 5 4 3 2 1

30. <https://leetcode.com/problems/min-stack/description/>

Design a stack that supports push, pop, top, and retrieving the minimum element in constant time.

Implement the MinStack class:

* MinStack() initializes the stack object.
* void push(int val) pushes the element val onto the stack.
* void pop() removes the element on the top of the stack.
* int top() gets the top element of the stack.
* int getMin() retrieves the minimum element in the stack.

You must implement a solution with O(1) time complexity for each function.

**Example 1:**

**Input**

["MinStack","push","push","push","getMin","pop","top","getMin"]

[[],[-2],[0],[-3],[],[],[],[]]

**Output**

[null,null,null,null,-3,null,0,-2]

**Explanation**

MinStack minStack = new MinStack();

minStack.push(-2);

minStack.push(0);

minStack.push(-3);

minStack.getMin(); // return -3

minStack.pop();

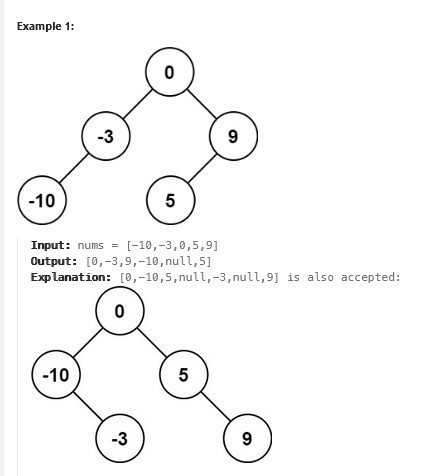
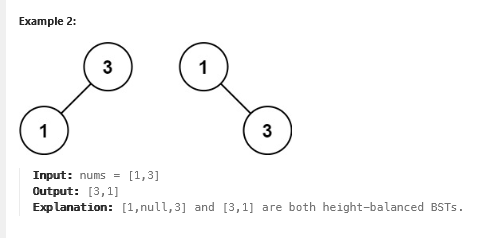
minStack.top(); // return 0

minStack.getMin(); // return -2

31. <https://leetcode.com/problems/convert-sorted-array-to-binary-search-tree/description/>

Given an integer array nums where the elements are sorted in **ascending order**, convert *it to a* ***height-balanced***

*binary search tree*.

class Solution {

public:

TreeNode\* sortedArrayToBST(vector<int>& nums) {

int l=nums.size()-1;

if(nums.empty()){

return NULL;

}

int mid=l/2;

TreeNode\* root=new TreeNode(nums[mid]);

vector<int>v1(nums.begin(),nums.begin()+mid); //\*\*\*\*\*carefully observe it

vector<int>v2(nums.begin()+mid+1,nums.end());

TreeNode \* leftt=sortedArrayToBST(v1);

TreeNode \* rightt=sortedArrayToBST(v2);

root->right=rightt;

root->left=leftt;

return root;

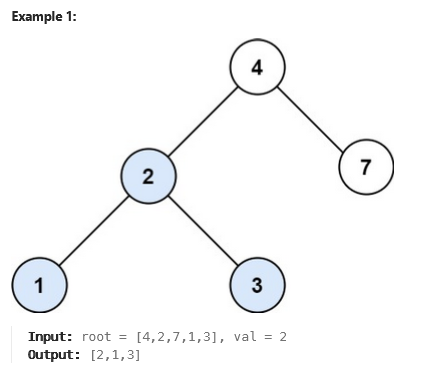
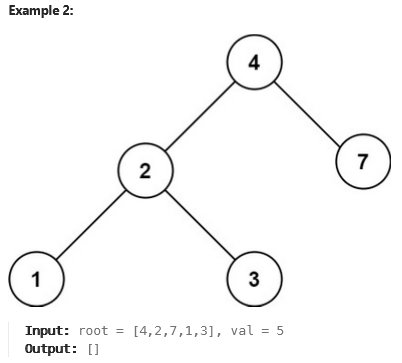
}

};

32. <https://leetcode.com/problems/search-in-a-binary-search-tree/description/>

You are given the root of a binary search tree (BST) and an integer val.

Find the node in the BST that the node's value equals val and return the subtree rooted with that node. If such a node does not exist, return null.

class Solution {

public:

TreeNode\* searchBST(TreeNode\* root, int val) {

queue<TreeNode\*>q;

q.push(root);

while(!q.empty()){

TreeNode \*f=q.front();

q.pop();

if(f->val==val){

return f;

}

if(f->left){

q.push(f->left);

}

if(f->right){

q.push(f->right);

}

}

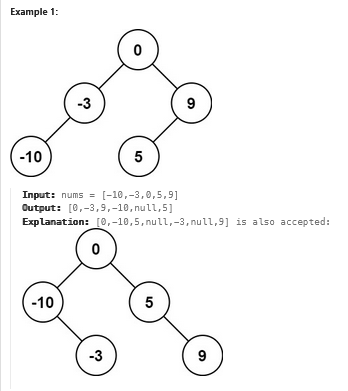
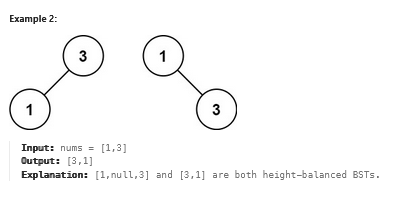
return NULL;

}

};

33. <https://leetcode.com/problems/convert-sorted-array-to-binary-search-tree/description/>

Given an integer array nums where the elements are sorted in **ascending order**, convert *it to a*  ***height-balanced*** *binary search tree*.

class Solution {

public:

TreeNode\* sortedArrayToBST(vector<int>& nums) {

int l=nums.size()-1;

if(nums.empty()){

return NULL;

}

int mid=l/2;

TreeNode\* root=new TreeNode(nums[mid]);

vector<int>v1(nums.begin(),nums.begin()+mid); //\*\*\*\*\*carefully observe it

vector<int>v2(nums.begin()+mid+1,nums.end());

TreeNode \* leftt=sortedArrayToBST(v1);

TreeNode \* rightt=sortedArrayToBST(v2);

root->right=rightt;

root->left=leftt;

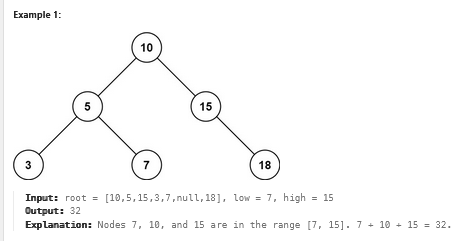
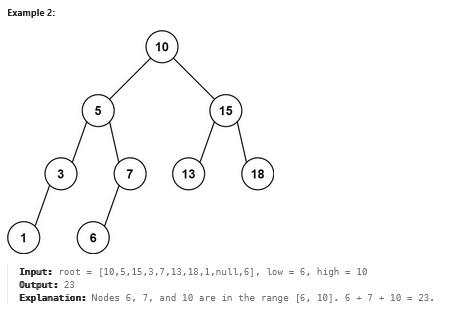
return root;

}

};

34. <https://leetcode.com/problems/range-sum-of-bst/description/>

Given the root node of a binary search tree and two integers low and high, return the sum of values of all nodes with a value in the ***inclusive*** range [low, high].

class Solution {

public:

    int rangeSumBST(TreeNode\* root, int low, int high) {

        queue<TreeNode \*>q;

        int sum=0;

        q.push(root);

        while(!q.empty()){

            TreeNode\* f=q.front();

            q.pop();

            if(f->val>=low && f->val<=high)

            sum+=f->val;

            if(f->left!=NULL){

                q.push(f->left);

            }

            if(f->right!=NULL){

                q.push(f->right);

            }

        }

        return sum;

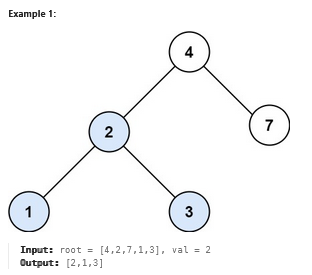
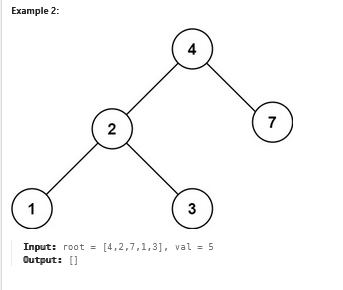
    }

};

35. <https://leetcode.com/problems/search-in-a-binary-search-tree/description/>

You are given the root of a binary search tree (BST) and an integer val.

Find the node in the BST that the node's value equals val and return the subtree rooted with that node. If such a node does not exist, return null.

class Solution {

public:

    TreeNode\* searchBST(TreeNode\* root, int val) {

        queue<TreeNode\*>q;

        q.push(root);

        while(!q.empty()){

            TreeNode \*f=q.front();

            q.pop();

            if(f->val==val){

                return f;

            }

            if(f->left){

                q.push(f->left);

            }

            if(f->right){

                q.push(f->right);

            }

        }

     return NULL;

    }

};

36. <https://leetcode.com/problems/maximum-product-of-two-elements-in-an-array/description/>

Given the array of integers nums, you will choose two different indices i and j of that array. *Return the maximum value of* (nums[i]-1)\*(nums[j]-1).

**Example 1:**

**Input:** nums = [3,4,5,2]

**Output:** 12

**Explanation:** If you choose the indices i=1 and j=2 (indexed from 0), you will get the maximum value, that is, (nums[1]-1)\*(nums[2]-1) = (4-1)\*(5-1) = 3\*4 = 12.

**Example 2:**

**Input:** nums = [1,5,4,5]

**Output:** 16

**Explanation:** Choosing the indices i=1 and j=3 (indexed from 0), you will get the maximum value of (5-1)\*(5-1) = 16.

**Example 3:**

**Input:** nums = [3,7]

**Output:** 12

class Solution {

public:

int maxProduct(vector<int>& nums) {

priority\_queue<int>q;

for(auto i:nums){

q.push(i);

}

int m=q.top()-1;

q.pop();

m=m\*(q.top()-1);

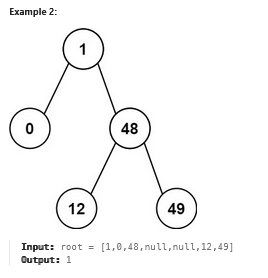
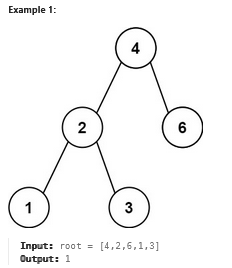
return m;

}

};

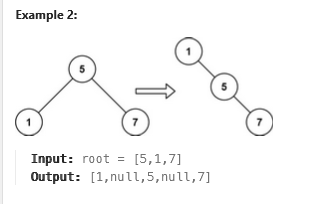
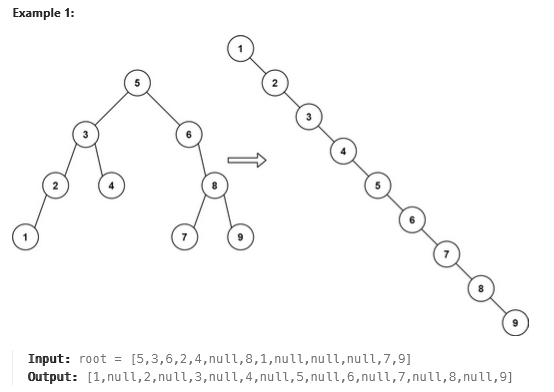
37. <https://leetcode.com/problems/minimum-absolute-difference-in-bst/description/>

Given the root of a Binary Search Tree (BST), return the minimum absolute difference between the values of any two different nodes in the tree.



38. <https://leetcode.com/problems/increasing-order-search-tree/description/>

Given the root of a binary search tree, rearrange the tree in **in-order** so that the leftmost node in the tree is now the root of the tree, and every node has no left child and only one right child



class Solution {

public:

void inorder(TreeNode\* root,vector<int>&v){

if(root==NULL)return;

inorder(root->left,v);

v.push\_back(root->val);

inorder(root->right,v);

}

TreeNode\* increasingBST(TreeNode\* root) {

vector<int>v;

TreeNode \*rt;

if(root==NULL){

return NULL;

}

inorder(root,v);

rt=new TreeNode(v[0]);

TreeNode \*tmp=rt;

for(int i=1;i<v.size();i++){

TreeNode\* t=new TreeNode(v[i]);

tmp->right=t;

tmp=tmp->right;

}

return rt;

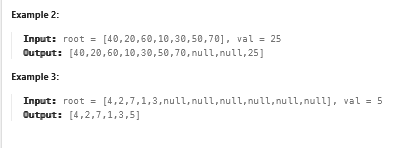
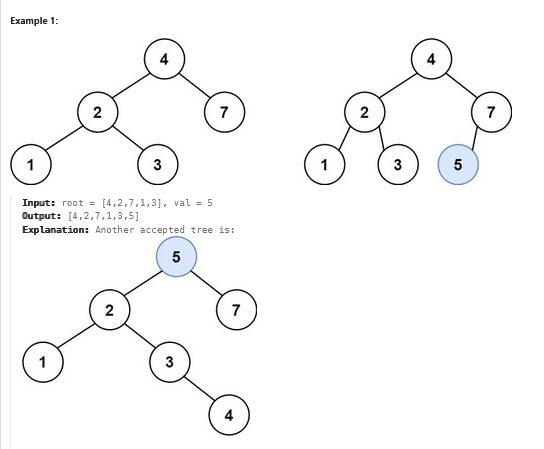
}

};

39. <https://leetcode.com/problems/insert-into-a-binary-search-tree/description/>

You are given the root node of a binary search tree (BST) and a value to insert into the tree. Return the root node of the BST after the insertion. It is **guaranteed** that the new value does not exist in the original BST.

**Notice** that there may exist multiple valid ways for the insertion, as long as the tree remains a BST after insertion. You can return **any of them**



40. <https://www.naukri.com/code360/problems/fourth-largest-element-in-the-array_1792782>

Given an array 'ARR' of 'N' integers. Find the fourth highest element present in the give 'ARR'. Return the minimum value of an integer which is -2147483648 if there it does not exists.

Detailed explanation ( Input/output format, Notes, Images )

**Constraints :**

1 <= N < 10^6

-10^6 <= ARR[i] <= 10^6

Where 'ARR[i]' denotes the 'ARR' element at 'ith' position.

Time Limit: 1 sec

**Sample Input 1:**

5

3 5 1 3 1

**Sample Output 1:**

1

**Explanation Of Sample Input 1:**

5 is the largest element, 3 is the second-largest element, again we have a 3 so it's the third largest and 1 is the fourth-largest, hence the answer 1.

**Sample Input 2:**

4

9 9 9 9

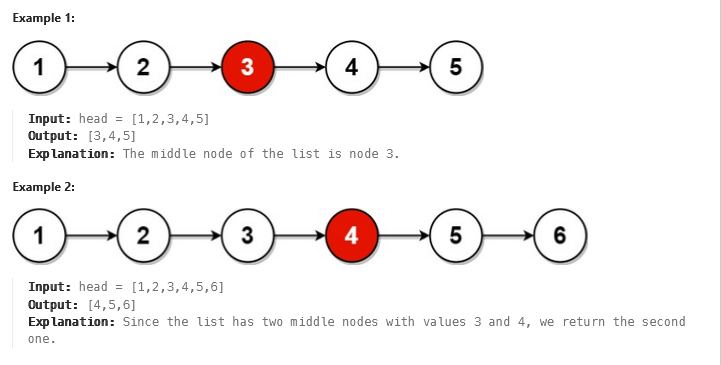
**Sample Output 2:**

9

41. <https://leetcode.com/problems/middle-of-the-linked-list/description/>

Given the head of a singly linked list, return the middle node of the linked list.

If there are two middle nodes, return **the second middle** node.



class Solution {

public:

ListNode\* middleNode(ListNode\* head) {

int i=0;

ListNode \*tmp=head;

while(tmp!=NULL){

i++;

tmp=tmp->next;

}

tmp=head;

if(i%2==0){

i=i/2;

for(int x=0;x<i;x++){

tmp=tmp->next;

}

return tmp;

}

else{

i=i/2;

for(int x=0;x<i;x++){

tmp=tmp->next;

}

return tmp;

}

}

};

42. <https://leetcode.com/problems/design-linked-list/description/>

Design your implementation of the linked list. You can choose to use a singly or doubly linked list.  
A node in a singly linked list should have two attributes: val and next. val is the value of the current node, and next is a pointer/reference to the next node.  
If you want to use the doubly linked list, you will need one more attribute prev to indicate the previous node in the linked list. Assume all nodes in the linked list are **0-indexed**.

Implement the MyLinkedList class:

* MyLinkedList() Initializes the MyLinkedList object.
* int get(int index) Get the value of the indexth node in the linked list. If the index is invalid, return -1.
* void addAtHead(int val) Add a node of value val before the first element of the linked list. After the insertion, the new node will be the first node of the linked list.
* void addAtTail(int val) Append a node of value val as the last element of the linked list.
* void addAtIndex(int index, int val) Add a node of value val before the indexth node in the linked list. If index equals the length of the linked list, the node will be appended to the end of the linked list. If index is greater than the length, the node **will not be inserted**.
* void deleteAtIndex(int index) Delete the indexth node in the linked list, if the index is valid.

**Example 1:**

**Input**

["MyLinkedList", "addAtHead", "addAtTail", "addAtIndex", "get", "deleteAtIndex", "get"]

[[], [1], [3], [1, 2], [1], [1], [1]]

**Output**

[null, null, null, null, 2, null, 3]

**Explanation**

MyLinkedList myLinkedList = new MyLinkedList();

myLinkedList.addAtHead(1);

myLinkedList.addAtTail(3);

myLinkedList.addAtIndex(1, 2); // linked list becomes 1->2->3

myLinkedList.get(1); // return 2

myLinkedList.deleteAtIndex(1); // now the linked list is 1->3

myLinkedList.get(1); // return 3

class Node {

public:

int val;

Node\* next;

Node(int val) {

this->val = val;

next = NULL;

}

};

class MyLinkedList {

public:

int size = 0;

Node\* head = NULL;

Node\* tail = NULL;

MyLinkedList() {}

int get(int index) {

if(index >= size)

return -1;

Node\* temp = head;

for(int i=0; i<index; ++i)

temp = temp->next;

return temp->val;

}

void addAtHead(int val) {

Node\* nnode = new Node(val);

nnode->next = head;

head = nnode;

if(size==0)

tail = nnode;

size++;

}

void addAtTail(int val) {

if(size ==0) {

addAtHead(val);

return;

}

Node\* nnode = new Node(val);

tail->next = nnode;

tail = nnode;

size++;

}

void addAtIndex(int index, int val) {

if(index>size)

return;

else if(index == size)

addAtTail(val);

else if(index==0)

addAtHead(val);

else {

Node\* temp =head;

Node\* nnode = new Node(val);

for(int i=0; i<index-1; ++i) {

temp = temp->next;

}

nnode->next = temp->next;

temp->next = nnode;

size++;

}

}

void deleteAtIndex(int index) {

if(index>=size)

return;

else if(index == size-1) {

if(index==0){

head==NULL;

tail==NULL;

}

else {

Node\* temp = head;

while(temp->next!=tail) {

temp= temp->next;

}

tail = temp;

temp->next = NULL;

}

}

else if(index==0) {

Node\* temp = head;

head = head->next;

temp =NULL;

}

else {

Node\* temp = head;

for(int i=0; i<index-1;++i) {

temp = temp->next;

}

Node\* t = temp->next;

temp->next = temp->next->next;

t->next = NULL;

}

size--;

}

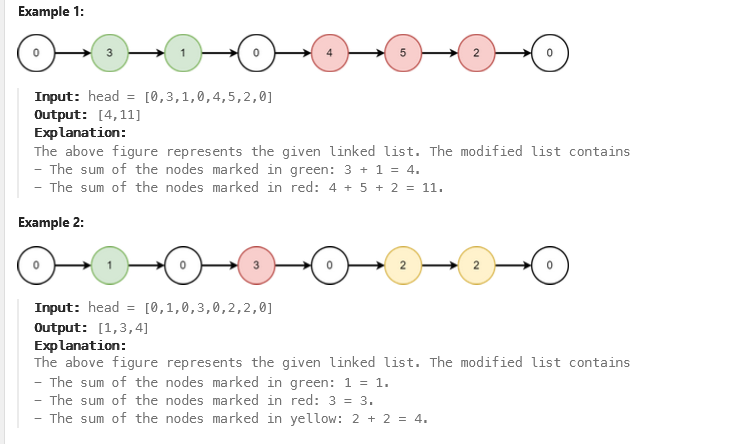
};

43. <https://leetcode.com/problems/merge-nodes-in-between-zeros/description/>

You are given the head of a linked list, which contains a series of integers **separated** by 0's. The **beginning** and **end** of the linked list will have Node.val == 0.

For **every** two consecutive 0's, **merge** all the nodes lying in between them into a single node whose value is the **sum** of all the merged nodes. The modified list should not contain any 0's.

Return the head of the modified linked list



class Solution {

public:

ListNode\* mergeNodes(ListNode\* head) {

ListNode \*d=new ListNode(-1);

ListNode \*h=d;

ListNode \*head2=d;

int t=0;

ListNode \*tmp=head->next;

while(tmp){

if(tmp->val==0){

ListNode \*tt=new ListNode(t);

h->next=tt;

h=h->next;

t=0;

}

else{

t=t+tmp->val;

}

tmp=tmp->next;

}

head2=head2->next;

return head2;

}

};

44. <https://www.naukri.com/code360/problems/queue-using-array-or-singly-linked-list_2099908>

Implement a Queue Data Structure specifically to store integer data using a Singly Linked List or an array.

You need to implement the following public functions :

1. Constructor: It initializes the data members as required.

2. enqueue(data): This function should take one argument of type integer. It enqueues the element into the queue.

3. dequeue(): It dequeues/removes the element from the front of the queue and in turn, returns the element being dequeued or removed. In case the queue is empty, it returns -1.

4. front(): It returns the element being kept at the front of the queue. In case the queue is empty, it returns -1.

5. isEmpty(): It returns a boolean value indicating whether the queue is empty or not.

**Operations Performed on the Queue :**

Query-1(Denoted by an integer 1): Enqueues integer data to the queue.

Query-2(Denoted by an integer 2): Dequeues the data kept at the front of the queue and returns it to the caller, return -1 if no element is present in the queue.

Query-3(Denoted by an integer 3): Fetches and returns the data being kept at the front of the queue but doesn't remove it, unlike the dequeue function, return -1 if no element is present in the queue.

Query-4(Denoted by an integer 4): Returns a boolean value denoting whether the queue is empty or not.

##### Sample Input 1 :

1

7

1 17

1 23

1 11

2

2

2

2

##### Sample Output 1 :

17

23

11

-1

##### Explanation for Sample Output 1 :

The first three queries are of enQueue, so we will push 17, 23, and 11 into the queue.

The next four queries are of deQueue, so we will starting removing elements from the queue, so the first element will be 17, then 23, and then 11. And after the third dequeue query, the queue is now empty so for the fourth query, we will return -1.

##### Sample Input 2 :

1

3

2

1 10

3

##### Sample Output 2 :

-1

10

#include <bits/stdc++.h>

class Node{

    public:

    int val;

    Node \*next;

    Node (int val){

        this->val=val;

        this->next=NULL;

    }

};

class Queue {

public:

    int size;

    Node \*start;

    Node \* end;

    Queue() {

      start=NULL;

      end=NULL;

      size=0;

    }

    /\*----------------- Public Functions of Queue -----------------\*/

    bool isEmpty() {

       if(start==NULL && end==NULL) return true;

       return false;

    }

    void enqueue(int data) {

         if(start==NULL){

             start=new Node(data);

             end=start;

         }

         else{

             end->next=new Node(data);

             end=end->next;

         }

         size++;

    }

    int dequeue() {

        if (isEmpty()) return -1;  // Return -1 if the queue is empty

        int value = start->val;   // Store the value to return

        Node \*temp = start;       // Temporary node to delete

        start = start->next;      // Move `start` to the next node

        if (start == NULL) {      // If the queue is now empty, reset `end`

            end = NULL;

        }

        delete temp;              // Free the memory of the dequeued node

        size--;

        return value;

    }

    int front() {

        if(isEmpty())return -1;

        return start->val;

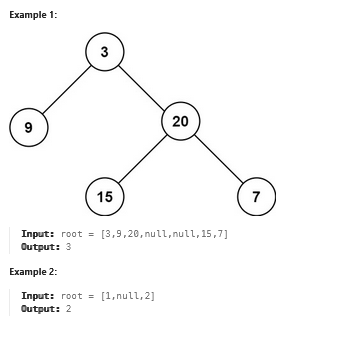
    }

};

45. <https://leetcode.com/problems/maximum-depth-of-binary-tree/description/>

Given the root of a binary tree, return its maximum depth.

A binary tree's **maximum depth** is the number of nodes along the longest path from the root node down to the farthest leaf node.



class Solution {

public:

int maxDepth(TreeNode\* root) {

if(root==NULL){

return 0;

}

if(root->left==NULL && root->right==NULL){

return 1;

}

int l=maxDepth(root->left);

int r=maxDepth(root->right);

return max(l,r)+1;

}

};

46. <https://leetcode.com/problems/left-and-right-sum-differences/description/>

Given a **0-indexed** integer array nums, find a **0-indexed** integer array answer where:

* answer.length == nums.length.
* answer[i] = |leftSum[i] - rightSum[i]|.

Where:

* leftSum[i] is the sum of elements to the left of the index i in the array nums. If there is no such element, leftSum[i] = 0.
* rightSum[i] is the sum of elements to the right of the index i in the array nums. If there is no such element, rightSum[i] = 0.

Return *the array* answer.

**Example 1:**

**Input:** nums = [10,4,8,3

**Output:** [15,1,11,22]

**Explanation:** The array leftSum is [0,10,14,22] and the array rightSum is [15,11,3,0].

The array answer is [|0 - 15|,|10 - 11|,|14 - 3|,|22 - 0|] = [15,1,11,22].

**Example 2:**

**Input:** nums = [1]

**Output:** [0]

**Explanation:** The array leftSum is [0] and the array rightSum is [0].

The array answer is [|0 - 0|] = [0].

class Solution {

public:

vector<int> leftRightDifference(vector<int>& nums) {

int n = nums.size();

vector<int>Lsum(n,0);

vector<int>Rsum(n,0);

vector<int>ans(n,0);

for(int i = 1; i<n; i++){

Lsum[i] = Lsum[i-1] + nums[i-1];

}

for(int i = n-2; i >=0;i--){

Rsum[i]=Rsum[i+1]+nums[i+1];

}

for(int i = 0 ; i<n;i++){

ans[i] = abs(Lsum[i]-Rsum[i]);

}

return ans;

}

};

47. <https://leetcode.com/problems/find-pivot-index/description/>

Given an array of integers nums, calculate the **pivot index** of this array.

The **pivot index** is the index where the sum of all the numbers **strictly** to the left of the index is equal to the sum of all the numbers **strictly** to the index's right.

If the index is on the left edge of the array, then the left sum is 0 because there are no elements to the left. This also applies to the right edge of the array.

Return the ***leftmost pivot index***. If no such index exists, return -1.

**Example 1:**

**Input:** nums = [1,7,3,6,5,6]

**Output:** 3

**Explanation:**

The pivot index is 3.

Left sum = nums[0] + nums[1] + nums[2] = 1 + 7 + 3 = 11

Right sum = nums[4] + nums[5] = 5 + 6 = 11

**Example 2:**

**Input:** nums = [1,2,3]

**Output:** -1

**Explanation:**

There is no index that satisfies the conditions in the problem statement.

**Example 3:**

**Input:** nums = [2,1,-1]

**Output:** 0

**Explanation:**

The pivot index is 0.

Left sum = 0 (no elements to the left of index 0)

Right sum = nums[1] + nums[2] = 1 + -1 = 0

class Solution {

public:

int pivotIndex(vector<int>& v) {

int n=v.size();

for(int i=1;i<v.size();i++){

v[i]=v[i-1]+v[i];

}

for(int i=0;i<n;i++){

int pre,post;

if(i==0){

pre=0;

post=v[n-1]-v[0];

}

else{

pre=v[i-1];

post=v[n-1]-v[i];

// cout<<pre<<"-->"<<post<<endl;

}

if(pre==post){

return i;

}

}

return -1;

}

};

48. <https://codeforces.com/group/MWSDmqGsZm/contest/223205/problem/L>

Given two arrays *A* and *B* of size *N*. Print a new array *C* that holds the concatenation of array *B* followed by array *A*

.

Note: Solve this problem using function.

Input

First line will contain a number *N*

(1≤*N*≤103)

1. #include <bits/stdc++.h>
2. using namespace std;
3. int main(){
5. int n;
6. cin>>n;
7. vector <int> v(2\*n);
8. vector<int>v1(n),v2(n);
9. for(int i=0;i<n;i++){
10. cin>>v2[i];
11. }
12. for(int i=0;i<n;i++){
13. cin>>v1[i];
14. }
15. v1.insert(v1.begin()+n,v2.begin(),v2.end());
16. for(int s:v1){
17. cout<<s<<" ";
18. }
19. return 0;
20. }

.

Second line will contain *N*

numbers (1≤*Ai*≤105) array *A*

elements.

Third line will contain *N*

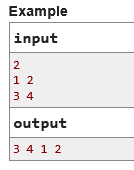
numbers (1≤*Bi*≤105) array *B*

elements.

Output

Print array *C*

elements separated by space.



49. <https://codeforces.com/group/MWSDmqGsZm/contest/219774/problem/C>

Given a number *N* and an array *A* of *N* numbers. Print the array after doing the following operations:

* Replace every positive number by 1.
* Replace every negative number by 2.

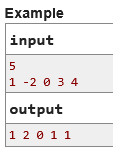
Input

First line contains a number *N* (2 ≤ *N* ≤ 1000) number of elements.

Second line contains *N* numbers (-105  ≤  *Ai*  ≤  105).

Output

Print the array after the replacement and it's values separated by space

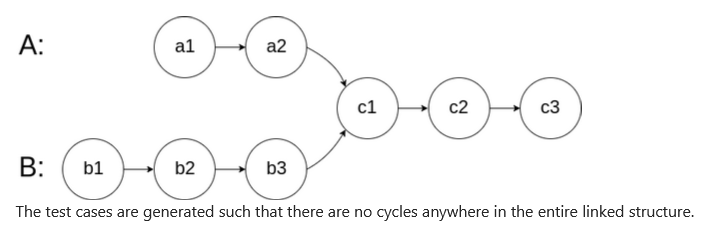


1. #include<bits/stdc++.h>
2. using namespace std;
4. int main() {
5. long long int n,sum=0;
6. cin >>n;
7. long long int a[n];
8. for(int i=0;i<n;i++){
9. cin>>a[i];
10. if(a[i]<0)
11. a[i]=2;
12. else if(a[i]==0)
13. a[i]=0;
14. else{
15. a[i]=1;
16. }
17. }
18. for(int i=0;i<n;i++){
19. cout<<a[i]<<" ";
21. }
22. return 0;
23. }

50. <https://leetcode.com/problems/intersection-of-two-linked-lists/description/>

Given the heads of two singly linked-lists headA and headB, return the node at which the two lists intersect. If the two linked lists have no intersection at all, return null.

For example, the following two linked lists begin to intersect at node c1



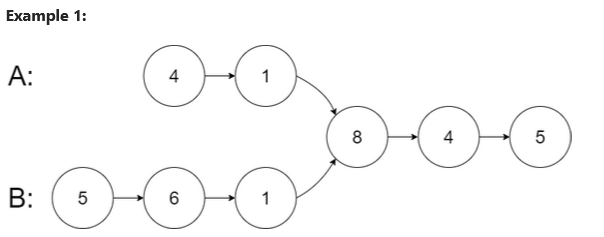
**Note** that the linked lists must **retain their original structure** after the function returns.

**Custom Judge:**

The inputs to the **judge** are given as follows (your program is **not** given these inputs):

* intersectVal - The value of the node where the intersection occurs. This is 0 if there is no intersected node.
* listA - The first linked list.
* listB - The second linked list.
* skipA - The number of nodes to skip ahead in listA (starting from the head) to get to the intersected node.
* skipB - The number of nodes to skip ahead in listB (starting from the head) to get to the intersected node.

The judge will then create the linked structure based on these inputs and pass the two heads, headA and headB to your program. If you correctly return the intersected node, then your solution will be **accepted**.



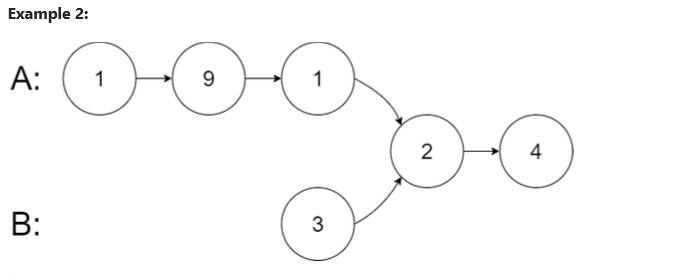
**Input:** intersectVal = 8, listA = [4,1,8,4,5], listB = [5,6,1,8,4,5], skipA = 2, skipB = 3

**Output:** Intersected at '8'

**Explanation:** The intersected node's value is 8 (note that this must not be 0 if the two lists intersect).

From the head of A, it reads as [4,1,8,4,5]. From the head of B, it reads as [5,6,1,8,4,5]. There are 2 nodes before the intersected node in A; There are 3 nodes before the intersected node in B.

- Note that the intersected node's value is not 1 because the nodes with value 1 in A and B (2nd node in A and 3rd node in B) are different node references. In other words, they point to two different locations in memory, while the nodes with value 8 in A and B (3rd node in A and 4th node in B) point to the same location in memory.

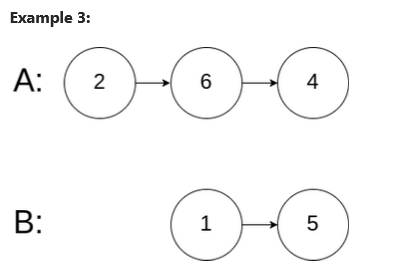


**Input:** intersectVal = 2, listA = [1,9,1,2,4], listB = [3,2,4], skipA = 3, skipB = 1

**Output:** Intersected at '2'

**Explanation:** The intersected node's value is 2 (note that this must not be 0 if the two lists intersect).

From the head of A, it reads as [1,9,1,2,4]. From the head of B, it reads as [3,2,4]. There are 3 nodes before the intersected node in A; There are 1 node before the intersected node in B.



**Input:** intersectVal = 0, listA = [2,6,4], listB = [1,5], skipA = 3, skipB = 2

**Output:** No intersection

**Explanation:** From the head of A, it reads as [2,6,4]. From the head of B, it reads as [1,5]. Since the two lists do not intersect, intersectVal must be 0, while skipA and skipB can be arbitrary values.

Explanation: The two lists do not intersect, so return null.

Could you write a solution that runs in O(m + n) time and use only O(1) memory?

class Solution {

public:

ListNode \*getIntersectionNode(ListNode \*headA, ListNode \*headB) {

ListNode \*tmp1 = headA;

ListNode \*tmp2 = headB;

int l1 = 0;

int l2 = 0;

// Calculate the length of list A

while (tmp1 != NULL) {

l1++;

tmp1 = tmp1->next;

}

// Calculate the length of list B

while (tmp2 != NULL) {

l2++;

tmp2 = tmp2->next;

}

// Reset pointers to the start of each list

tmp1 = headA;

tmp2 = headB;

// Align the pointers by skipping the difference in length

if (l1 > l2) {

for (int i = 0; i < (l1 - l2); i++) { // Corrected here: l1 > l2, skip (l1 - l2)

tmp1 = tmp1->next;

}

} else {

for (int i = 0; i < (l2 - l1); i++) { // Corrected here: l2 > l1, skip (l2 - l1)

tmp2 = tmp2->next;

}

}

// Traverse both lists together to find the intersection

while (tmp1 != tmp2) {

tmp1 = tmp1->next;

tmp2 = tmp2->next;

}

return tmp1; // Either the intersection node or NULL if no intersection

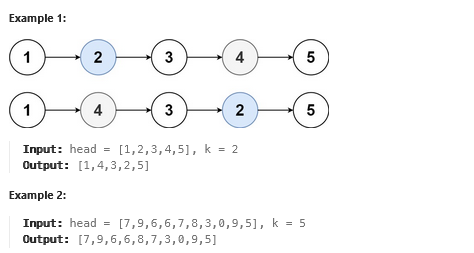
}

};

51. <https://leetcode.com/problems/swapping-nodes-in-a-linked-list/description/>

You are given the head of a linked list, and an integer k.

Return the head of the linked list after ***swapping*** the values of the kth node from the beginning and the kth node from the end (the list is ***1-indexed***).



class Solution {

public:

    ListNode\* swapNodes(ListNode\* head, int k) {

        ListNode \* tmp=head;

        int l=0;

        while(tmp!=NULL){

          l++;

          tmp=tmp->next;

        }

        //\*\*\*\*\*\*\*

        int k1 = l - k + 1;

        ListNode \* t1=head;

        ListNode \* t2=head;

        for(int i=1;i<k;i++){

            t1=t1->next;

        }

        for(int i=1;i<k1;i++){

            t2=t2->next;

        }

        swap(t1->val,t2->val);

        return head;

    }

};

52. <https://codeforces.com/group/MWSDmqGsZm/contest/219774/problem/F>

Given a number *N* and an array *A* of *N* numbers. Print the array in a reversed order.

Note:

\*Don't use built-in-functions.

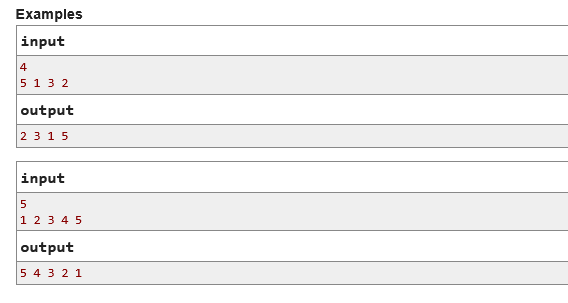
Input

First line contains a number *N* (1 ≤ *N* ≤ 103) number of elements.

Second line contains *N* numbers (0 ≤ *Ai* ≤ 109).

Output

Print the array in a reversed order.



1. #include <bits/stdc++.h>
2. using namespace std;
3. int main(){
5. int n;
6. cin>>n;
7. vector<int>v(n);
8. for(int i=0;i<n;i++){
9. cin>>v[i];
10. }
11. for(int i=0;i<n/2;i++){
12. swap(v[i],v[n-i-1]);
13. }
14. for(int x:v){
15. cout<<x<<" ";
16. }
18. return 0;
19. }

53. <https://codeforces.com/group/MWSDmqGsZm/contest/219774/problem/C>

Given a number *N* and an array *A* of *N* numbers. Print the array after doing the following operations:

* Replace every positive number by 1.
* Replace every negative number by 2.

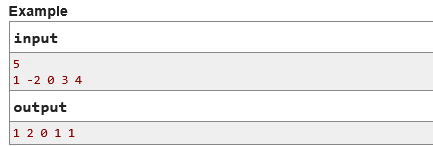
Input

First line contains a number *N* (2 ≤ *N* ≤ 1000) number of elements.

Second line contains *N* numbers (-105  ≤  *Ai*  ≤  105).

Output

Print the array after the replacement and it's values separated by space



1. #include<bits/stdc++.h>
2. using namespace std;
4. int main() {
5. long long int n,sum=0;
6. cin >>n;
7. long long int a[n];
8. for(int i=0;i<n;i++){
9. cin>>a[i];
10. if(a[i]<0)
11. a[i]=2;
12. else if(a[i]==0)
13. a[i]=0;
14. else{
15. a[i]=1;
16. }
17. }
18. for(int i=0;i<n;i++){
19. cout<<a[i]<<" ";
21. }
22. return 0;
23. }

54. <https://codeforces.com/group/MWSDmqGsZm/contest/329103/problem/D>

You are given an array *a* of *n* integers, count the number of element *ai* in the array such that *ai*+1 is also exists in the array *a*

If there're duplicates in *a*

count them separately.

Input

The first line contains an integer *n*(1≤*n*≤103)

the number of elements in the array *a*

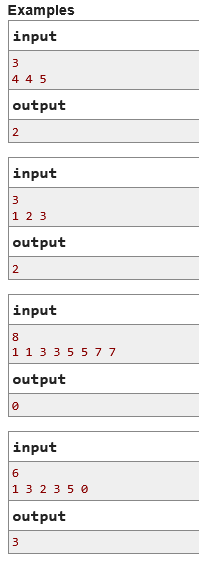
The second line contains *n*

integers *ai*(0≤*Xi*≤103) the elements of the array *a*

.

Output

output the number of elements as descriped above.



1. #include <bits/stdc++.h>
2. using namespace std;
3. int main(){
5. int n;
6. cin>>n;
7. vector<int>v(n);
8. for(int i=0;i<n;i++){
9. cin>>v[i];
10. }
11. int c=0;
12. for(int x:v){
13. auto it=find(v.begin(),v.end(),x+1);
14. if(it!=v.end()){
15. c++;
16. }
17. }
18. cout<<c<<endl;
19. return 0;
20. }

55. <https://codeforces.com/group/MWSDmqGsZm/contest/219856/problem/J>

Given a string *S*. Determine how many times does each letter occurred in *S*.

Input

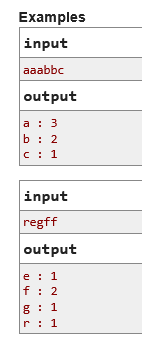
Only one line contains the string *S* (1 ≤ |*S*| ≤ 107) where |S| is the length of the string and it consists of only lowercase English letters.

Output

For each character that appears in *S*, print a single line that contains the following format: "*X* : *Y*" where *X* is the letter and *Y* is the number of times that letter *X* occurred in *S*.

1. #include <bits/stdc++.h>
2. using namespace std;
3. int main(){
4. string s;
5. cin>>s;
6. int n=s.size();
7. sort(s.begin(),s.end());
8. int c=1;
9. for(int i=1;i<s.size();i++){
10. if(s[i]!=s[i-1]){
11. cout<<s[i-1]<<" : "<<c<<endl;
12. c=1;
13. }
14. else{
15. c++;
16. }
17. }
18. cout<<s[n-1]<<" : "<<c<<endl;
19. return 0;
20. }

Note: you must print letters in ascending order.



56. <https://codeforces.com/group/MWSDmqGsZm/contest/219774/problem/Y>

Given 2 numbers *N* and *Q*, an array *A* of *N* number and *Q* number of pairs *L*, *R*. For each query *Q* print a single line that contains the summation of all numbers from index *L* to index *R*

Input

First line contains two numbers *N*

, *Q* (1≤*N*,*Q*≤105) where *N* is number of elements in *A* and *Q*

is number of query pairs.

Second line contains *N*

numbers(1≤*Ai*≤109)

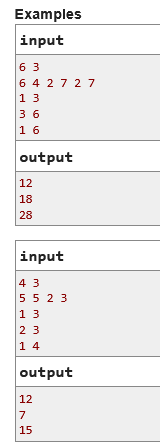
Next *Q*

lines contains *L*,*R* (1≤*L*≤*R*≤*N*)

Output

For each query *Q*

print a single line that contains the summation of all numbers from index *L* to index *R*.



1. #include <bits/stdc++.h>
2. using namespace std;
3. int main(){
4. long long int n,q;
5. cin>>n>>q;
6. vector<long long int>v(n+1);
8. for(int i=1;i<=n;i++){
9. cin>>v[i];
10. }
11. vector<long long int>p(n+1);
12. p[1]=v[1];
13. for(int i=2;i<=n;i++){
14. p[i]=p[i-1]+v[i];
15. }
16. while(q--){
17. long long int l,r,sum=0;
18. cin>>l>>r;
19. if(l==1){
20. sum=p[r];
21. cout<<sum<<endl;
22. }
23. else{
24. sum=p[r]-p[l-1];
25. cout<<sum<<endl;
26. }
27. }
28. return 0;
29. }

57. <https://codeforces.com/group/MWSDmqGsZm/contest/219774/problem/Z>

Given 2 numbers *N* and *Q*, array A of N numbers and Q queries each one contains a number *X*

.

For each query print a single line that contains "found" if the number *X*

exists in array *A*

otherwise, print "not found".

Input

First line contains two numbers *N*

, *Q* (1≤*N*,*Q*≤105)

.

Second line contains *N*

numbers (1≤*Ai*≤109)

.

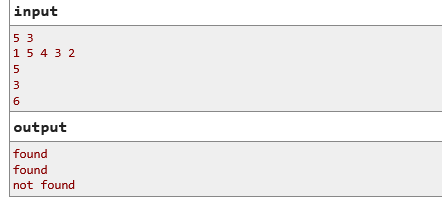
Next *Q*

lines contains *X* (1≤*X*≤109)

.

Output

Print the answer for each query in a single line



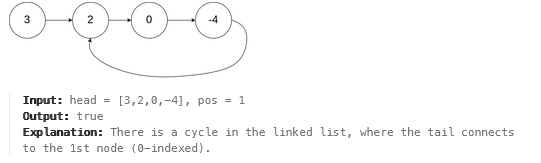
1. #include<bits/stdc++.h>
2. using namespace std;
3. int main(){
5. long long n,q;
6. cin>>n>>q;
7. vector<long long int>v(n);
8. for(int i=0;i<n;i++){
9. cin>>v[i];
10. }
11. sort(v.begin(),v.end());
12. for(int i=0;i<q;i++){
13. int x,f=0;
14. cin>>x;
15. int l=0,r=n-1;
16. while(l<=r){
17. int mid=(l+r)/2;
18. if(v[mid]==x){
19. f=1;
20. break;
21. }
22. else if(v[mid]>x){
23. r=mid-1;
24. }
25. else{
26. l=mid+1;
27. }
28. }
29. if(f==1){
30. cout<<"found"<<endl;
31. }
32. else{
33. cout<<"not found"<<endl;
34. }
36. }
37. return 0;
38. }

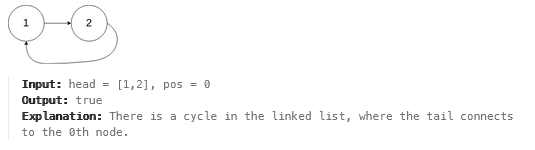
58. <https://leetcode.com/problems/linked-list-cycle/description/>

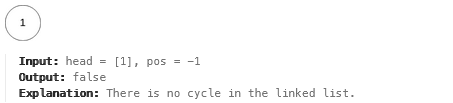
Given head, the head of a linked list, determine if the linked list has a cycle in it.

There is a cycle in a linked list if there is some node in the list that can be reached again by continuously following the next pointer. Internally, pos is used to denote the index of the node that tail's next pointer is connected to. **Note that pos is not passed as a parameter**.

Return true if there is a cycle in the linked list. Otherwise, return false







class Solution {

public:

    bool hasCycle(ListNode \*head) {

        if (head == NULL || head->next == NULL) {  // Check if the list is empty or has only one node

            return false; // A single node or empty list cannot have a cycle

        }

        ListNode \*fast = head;

        ListNode \*slow = head;

        // Use the two-pointer (fast and slow) technique to detect a cycle

        while (fast != NULL && fast->next != NULL) {

            slow = slow->next;             // Move slow pointer one step

            fast = fast->next->next;       // Move fast pointer two steps

            if (fast == slow) {            // If the fast pointer meets the slow pointer, there's a cycle

                return true;

            }

        }

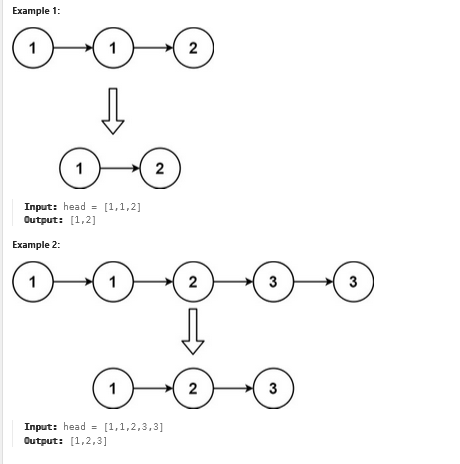
        return false; // If the loop ends, there's no cycle

    }

};

59. <https://leetcode.com/problems/remove-duplicates-from-sorted-list/description/>

Given the head of a sorted linked list, delete all duplicates such that each element appears only once. Return the linked list ***sorted*** as well.



class Solution {

public:

    ListNode\* deleteDuplicates(ListNode\* head) {

        if (head == NULL || head->next == NULL) {

            return head;

        }

        ListNode \*tmp=head;

            while (tmp != NULL && tmp->next != NULL) {

            if (tmp->val == tmp->next->val) {

                // Duplicate detected; remove the next node

                ListNode\* dlt = tmp->next;

                tmp->next = tmp->next->next; // Update the link to skip the duplicate

                delete dlt;                  // Free memory of the removed node

            } else {

                // Move to the next node if no duplicate

                tmp = tmp->next;

            }

        }

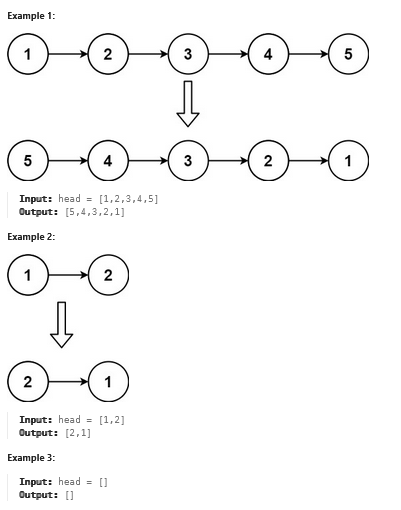
        return head;

        }

};

60. <https://leetcode.com/problems/reverse-linked-list/description/>

Given the head of a singly linked list, reverse the list, and return the reversed list



class Solution {

public:

    ListNode\* reverseList(ListNode\* head) {

        if(head==NULL || head->next==NULL){

            return head;

        }

        ListNode \*pre=head;

         ListNode \*curr=head->next;

          pre->next=NULL;

          while(curr!=NULL){

             ListNode \*nx=curr->next;

             curr->next=pre;

             pre=curr;

             curr=nx;

          }

          return pre;

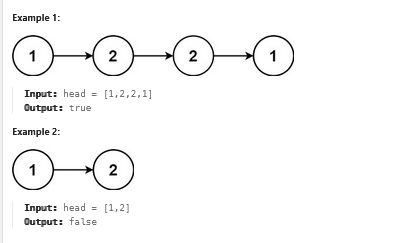
    }

};

61. <https://leetcode.com/problems/palindrome-linked-list/description/>

Given the head of a singly linked list, return true *if it is a*

*Palindrome* *or* false *otherwise*.



class Solution {

public:

    bool isPalindrome(ListNode\* head) {

        if(head==NULL || head->next==NULL){

            return true;

        }

        ListNode \*list2=new ListNode(head->val);

        ListNode \*head2=list2;

        ListNode \*t=head->next;

        while(t){

           ListNode \*tmp=new ListNode(t->val);

           list2->next=tmp;

           //\*\*\*\*\*\*\*\*\*\*

           list2=list2->next;

           t=t->next;

        }

        ListNode \*tm=head2;

        while(tm){

            tm=tm->next;

        }

        int l=1;

        ListNode \*pre=head;

        ListNode \*cur=head->next;

        pre->next=NULL;

        while(cur){

            l++;

            ListNode \*nx=cur->next;

            cur->next=pre;

            pre=cur;

            cur=nx;

        }

        head=pre;

        l=l/2;

        ListNode \*tmp1=head;

         ListNode \*tmp2=head2;

      for(int i=0;i<l;i++){

         cout<<tmp1->val<<".."<<tmp2->val<<endl;

        if(tmp1->val!=tmp2->val){

            return false;

        }

        tmp1=tmp1->next;

        tmp2=tmp2->next;

      }

      return true;

    }

};

62. <https://leetcode.com/problems/delete-node-in-a-linked-list/description/>

There is a singly-linked list head and we want to delete a node node in it.

You are given the node to be deleted node. You will **not be given access** to the first node of head.

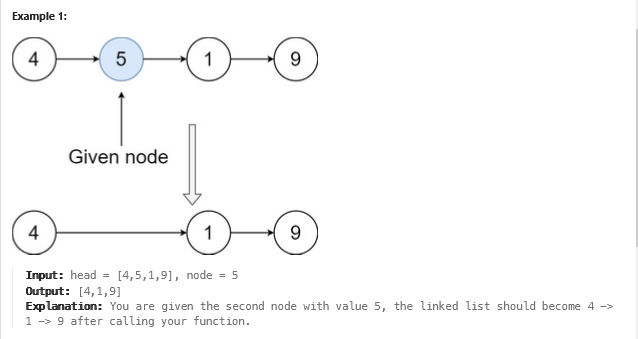
All the values of the linked list are **unique**, and it is guaranteed that the given node node is not the last node in the linked list.

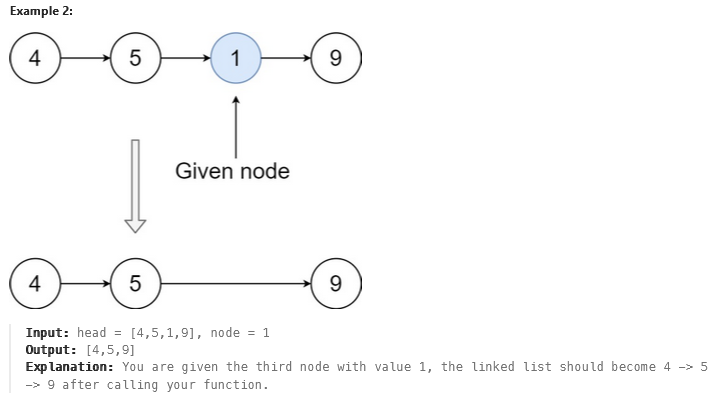
Delete the given node. Note that by deleting the node, we do not mean removing it from memory. We mean:

* The value of the given node should not exist in the linked list.
* The number of nodes in the linked list should decrease by one.
* All the values before node should be in the same order.
* All the values after node should be in the same order.

**Custom testing:**

* For the input, you should provide the entire linked list head and the node to be given node. node should not be the last node of the list and should be an actual node in the list.
* We will build the linked list and pass the node to your function.
* The output will be the entire list after calling your function.





class Solution {

public:

void deleteNode(ListNode\* node) {

ListNode \*tmp=node->next;

node->val=node->next->val;

node->next=node->next->next;

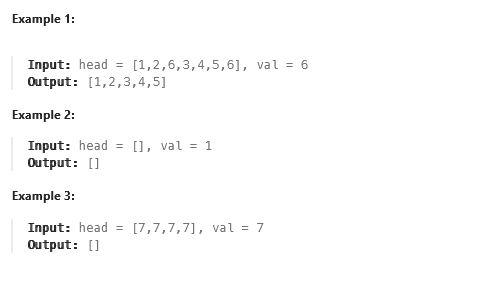
delete tmp;

}

};

63. <https://leetcode.com/problems/remove-linked-list-elements/description/>

Given the head of a linked list and an integer val, remove all the nodes of the linked list that has Node.val == val, and return the new head.



class Solution {

public:

    ListNode\* removeElements(ListNode\* &head, int val) {

        ListNode \*curr=head;

        ListNode \*d=new ListNode(-1);

        d->next=head;

        ListNode \*pre=d;

        while(curr){

            ListNode \*nx=curr->next;

            if(curr->val==val){

                ListNode \*tmp=curr;

                pre->next=nx;

                curr=nx;

                delete tmp;

            }

            else{

                pre=curr;

                curr=nx;

            }

        }

        ListNode \* newhead=d->next;

        delete d;

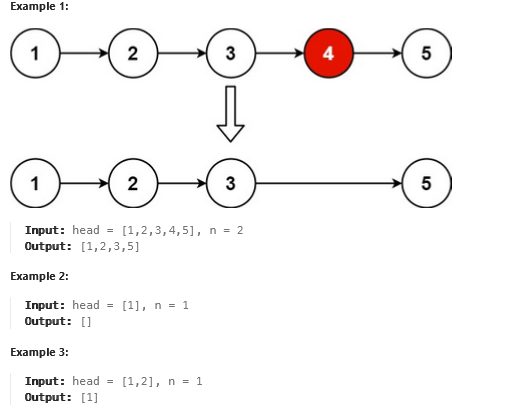
        return newhead;

    }

};

64. <https://leetcode.com/problems/remove-nth-node-from-end-of-list/>

Given the head of a linked list, remove the nth node from the end of the list and return its head.



class Solution {

public:

    ListNode\* removeNthFromEnd(ListNode\* head, int n) {

      ListNode \*tmp=head;

      int l=0;

      while(tmp){

        l++;

        tmp=tmp->next;

      }

      if(head->next==NULL){

        head=NULL;

        return head;

      }

      if(l==n){

        ListNode \*a=head;

        head=head->next;

        delete a;

        return head;

      }

      l=l-n;

      tmp=head;

      for(int i=1;i<l;i++){

        tmp=tmp->next;

      }

     ListNode \*t=tmp->next;

     tmp->next=tmp->next->next;

     delete t;

     return head;

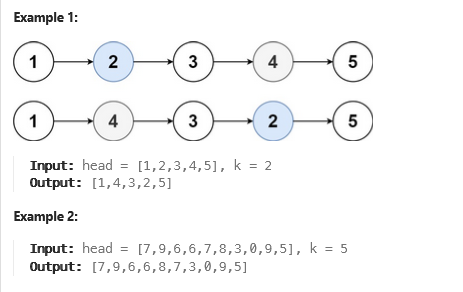
    }

};

65. <https://leetcode.com/problems/swapping-nodes-in-a-linked-list/description/>

You are given the head of a linked list, and an integer k.

Return the head of the linked list after ***swapping*** the values of the kth node from the beginning and the kth node from the end (the list is ***1-indexed***)



class Solution {

public:

    ListNode\* swapNodes(ListNode\* head, int k) {

        ListNode \* tmp=head;

        int l=0;

        while(tmp!=NULL){

          l++;

          tmp=tmp->next;

        }

        //\*\*\*\*\*\*\*

        int k1 = l - k + 1;

        ListNode \* t1=head;

        ListNode \* t2=head;

        for(int i=1;i<k;i++){

            t1=t1->next;

        }

        for(int i=1;i<k1;i++){

            t2=t2->next;

        }

        swap(t1->val,t2->val);

        return head;

    }

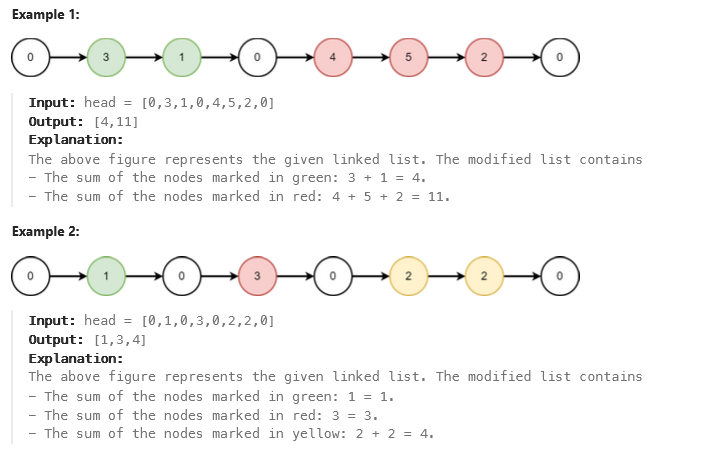
};

66. <https://leetcode.com/problems/merge-nodes-in-between-zeros/description/>

You are given the head of a linked list, which contains a series of integers **separated** by 0's. The **beginning** and **end** of the linked list will have Node.val == 0.

For **every** two consecutive 0's, **merge** all the nodes lying in between them into a single node whose value is the **sum** of all the merged nodes. The modified list should not contain any 0's.

Return the head of the modified linked list.

****

class Solution {

public:

    ListNode\* mergeNodes(ListNode\* head) {

       ListNode \*d=new ListNode(-1);

       ListNode \*h=d;

       ListNode \*head2=d;

       int t=0;

       ListNode \*tmp=head->next;

       while(tmp){

        if(tmp->val==0){

            ListNode \*tt=new ListNode(t);

            h->next=tt;

            h=h->next;

            t=0;

        }

        else{

             t=t+tmp->val;

        }

        tmp=tmp->next;

       }

       head2=head2->next;

       return head2;

    }

};

**67.** [**https://www.hackerrank.com/contests/assignment-01-a-basic-data-structure-a-batch-06/challenges/duplicate-12**](https://www.hackerrank.com/contests/assignment-01-a-basic-data-structure-a-batch-06/challenges/duplicate-12)

You will be given an array **A** of size **N**. Print "**YES**" if there is any duplicate value in the array, "**NO**" otherwise.

**Input Format**

* First line will contain **N**.
* Second line will contain the array **A**.

**Constraints**

1. 1 <= **N** <= 100000
2. 0 <= **A[i]** <= 10^9; Where 0 <= **i** < N

**Output Format**

* Output "**YES**" or "**NO**" without the quotation marks according to the problem statement.

**Sample Input 0**

5

1 2 3 4 5

**Sample Output 0**

NO

**Sample Input 1**

6

2 1 3 5 2 1

**Sample Output 1**

YES

#include<bits/stdc++.h>

using namespace std;

int main(){

long long int n;

cin>>n;

vector<long long int>v(n);

for(int i=0;i<n;i++){

cin>>v[i];

}

sort(v.begin(),v.end());

for(int i=1;i<n;i++){

if(v[i]==v[i-1]){

cout<<"YES"<<endl;

return 0;

}

}

cout<<"NO"<<endl;

return 0;

}

68. <https://www.hackerrank.com/contests/assignment-01-a-basic-data-structure-a-batch-06/challenges/get-prefix-sum>

You will be given an integer array **A** of size **N**. You need to print the prefix sum array of the array **A** in reverse order.

**Input Format**

* First line will contain **N**.
* Next line of contain the array **A**.

**Constraints**

1. 1 <= **N** <= 10^5
2. 1 <= **A[i]** <= 10^9; Where 0 <= **i** < N

**Output Format**

* Output the prefix sum array in reverse order.

**Sample Input 0**

5

2 4 1 5 3

**Sample Output 0**

15 12 7 6 2

**Explanation 0**

The prefix sum of the given array is: 2 6 7 12 15.

The reverse order is: 15 12 7 6 2.

**Sample Input 1**

3

1000000000 1000000000 1000000000

**Sample Output 1**

3000000000 2000000000 1000000000

#include<bits/stdc++.h>

using namespace std;

int main(){

long long int n;

cin>>n;

vector<long long int>v(n);

for(int i=0;i<n;i++){

cin>>v[i];

}

for(int i=1;i<n;i++){

v[i]=v[i]+v[i-1];

}

reverse(v.begin(),v.end());

for(int i=0;i<n;i++){

cout<<v[i]<<" ";

}

return 0;

}

69. <https://www.hackerrank.com/contests/assignment-01-a-basic-data-structure-a-batch-06/challenges/sorted-2-2>

You will given an array **A** of size **N**. You need to tell if the array is already sorted or not. If the array is sorted in **ascending** order print "**YES**", otherwise print "**NO**".

**Input Format**

* First line will contain **T**, the number of test cases.
* The first line of each test case will contain **N**.
* The second line of each test case will contain the array **A**.

**Constraints**

1. 1 <= **T** <= 1000
2. 1 <= **N** <= 1000
3. 0 <= **A[i]** <= 1000; Where 0 <= **i** < N

**Output Format**

* Output "**YES**" or "**NO**" without the quotation marks according to the problem statement.

**Sample Input 0**

3

5

2 4 6 7 10

8

1 100 101 120 120 121 1000 1000

4

100 1 102 12

**Sample Output 0**

YES

YES

NO

#include<bits/stdc++.h>

using namespace std;

int main(){

long long int n;

cin>>n;

for(int i=0;i<n;i++){

long long int x;

cin>>x;

vector<long long int>v(x);

for(int j=0;j<x;j++){

cin>>v[j];

}

long long int f=0;

for(int j=1;j<x;j++){

if(v[j]<v[j-1]){

f=1;

//cout<<"...."<<v[j-1]<<".."<<v[j];

break;

}

}

if(f==0)

cout<<"YES"<<endl;

else

cout<<"NO"<<endl;

}

return 0;

}

70. <https://www.hackerrank.com/contests/assignment-01-a-basic-data-structure-a-batch-06/challenges/insert-the-vector>

You will given an integer array **A** of size **N** and another array **B** of size **M**. Also you will be given an index **X**. You need to insert the whole array **B** to the index **X** of array **A**.

**Input Format**

* First line will contain **N**.
* Second line will contain array **A**.
* Third line will contain **M**.
* Fourth line will contain array **B**.
* The last line will contain **X**.

**Constraints**

1. 1 <= **N, M** <= 10^3
2. 1 <= **A[i], B[j]** <= 10^3; Where 0 <= **i** < N and 0 <= **j** < M
3. 0 <= **X** <= N

**Output Format**

* Output the final array **A**.

**Sample Input 0**

5

2 3 4 5 6

3

10 20 30

3

**Sample Output 0**

2 3 4 10 20 30 5 6

**Sample Input 1**

5

2 3 4 5 6

3

10 20 30

0

**Sample Output 1**

10 20 30 2 3 4 5 6

**Sample Input 2**

4

3 4 5 6

3

10 20 30

4

**Sample Output 2**

3 4 5 6 10 20 30

#include<bits/stdc++.h>

using namespace std;

int main(){

long long int n,m,x;

cin>>n;

vector<long long int>v1(n);

for(int i=0;i<n;i++){

cin>>v1[i];

}

cin>>m;

vector<long long int>v2(m);

for(int i=0;i<m;i++){

cin>>v2[i];

}

cin>>x;

v1.insert(v1.begin()+x,v2.begin(),v2.end());

for(int i=0;i<(m+n);i++){

cout<<v1[i]<<" ";

}

return 0;

}

71. <https://www.hackerrank.com/contests/assignment-01-a-basic-data-structure-a-batch-06/challenges/printing-x>

You will be given an positive **odd** integer **N**, you need to print the pattern for it. See sample input and output for understanding the pattern.

**Input Format**

* Input will contain only **N**.

**Constraints**

1. 1 <= **N** <= 20 and N is odd.

**Output Format**

* Output the pattern.

**Sample Input 0**

#include<bits/stdc++.h>

using namespace std;

int main(){

int n;

cin>>n;

for(int i=0;i<n/2;i++){

for(int j=0;j<i;j++){

cout<<" ";

}

cout<<"\\";

int c=i\*2+2;

c=n-c;

for(int j=0;j<c;j++){

cout<<" ";

}

cout<<"/";

cout<<endl; }

for(int i=0;i<n/2;i++){

cout<<" ";

}

cout<<"X";

cout<<endl;;

for(int i=n/2-1;i>=0;i--){

for(int j=0;j<i;j++){

cout<<" ";

}

cout<<"/";

int c=i\*2+2;

c=n-c;

for(int j=0;j<c;j++){

cout<<" ";

}

cout<<"\\";

cout<<endl;

}

return 0;

}

5

**Sample Output 0**

\ /

\ /

X

/ \

/ \

**Sample Input 1**

7

**Sample Output 1**

\ /

\ /

\ /

X

/ \

/ \

/ \

**Sample Input 2**

3

**Sample Output 2**

\ /

X

/ \

**Sample Input 3**

1

**Sample Output 3**

X

72. <https://www.hackerrank.com/contests/assignment-02-a-basic-data-structures-a-batch-06/challenges/get-difference>

You need to take a singly linked list of integer value as input and print the difference between the maximum and minimum value of the singly linked list.

**Note**: You must use singly linked list to solve this problem, otherwise you will not get marks.

**Input Format**

* Input will contain the values of the singly linked list, and will terminate with -1.

**Constraints**

1. 1 <= **N** <= 10^5; Here N is the maximum number of nodes of the linked list.
2. -10^9 <= **V** <= 10^9; Here V is the value of each node.

**Output Format**

* Output the difference between the maximum and minimum value.

**Sample Input 0**

2 4 1 5 3 6 -1

**Sample Output 0**

5

**Sample Input 1**

2 -1

**Sample Output 1**

0

#include <bits/stdc++.h>

using namespace std;

class Node{

public:

long long int val;

Node \* next;

Node(long long int val){

this->val=val;

this->next=NULL;

}

};

void insert(Node \* &head,Node \* &tail,long long int val){

Node \* newnode=new Node(val);

if(head==NULL){

tail=newnode;

head=newnode;

return;

}

tail->next=newnode;

tail=tail->next;

}

int main(){

Node \*head=NULL;

Node \*tail=NULL;

long long int val;

long long int mi=INT16\_MAX,mx=INT16\_MIN;

while(true){

cin>>val;

if(val==-1){

break;

}

mx=max(mx,val);

mi=min(mi,val);

insert(head,tail,val);

}

cout<<mx-mi<<endl;

return 0;

}

73. <https://www.hackerrank.com/contests/assignment-02-a-basic-data-structures-a-batch-06/challenges/search-12>

You need to take a singly linked list of integer values as input. Afterward, you will be given an integer value **X**. Your task is to determine whether **X** is present in the linked list or not. If it is present, print its first index from the left side; otherwise, print -1. Assume that the linked list's index starts with 0.

**Note**: You must use a singly linked list; otherwise, you will not receive marks.

**Input Format**

* First line will contain **T**, the number of test cases.
* First line of each test case will contain the values of the singly linked list, and will terminate with -1.
* Second line of each test case will contain **X**.

**Constraints**

1. 1 <= **T** <= 100
2. 1 <= **N** <= 10^5; Here N is the maximum number of nodes of the linked list.
3. -10^9 <= **V** <= 10^9; Here V is the value of each node.
4. -10^9 <= **X** <= 10^9

**Output Format**

* Output the index of **X** in the linked list.

**Sample Input 0**

4

1 2 3 4 5 -1

3

1 2 3 -1

5

1 -1

1

10 20 -1

20

**Sample Output 0**

2

-1

0

1

#include <bits/stdc++.h>

using namespace std;

class Node {

public:

long long int val;

Node\* next;

Node(long long int val) {

this->val = val;

this->next = NULL;

}

};

void insert(Node\*& head, Node\*& tail, long long int val) {

Node\* newnode = new Node(val);

if (head == NULL) {

tail = newnode;

head = newnode;

return;

}

tail->next = newnode;

tail = tail->next;

}

int main() {

int n;

cin >> n;

if (n <= 0) {

return 0;

}

for (int i = 0; i < n; i++) {

Node\* head = NULL;

Node\* tail = NULL;

long long int val;

while (true) {

cin >> val;

if (val == -1) {

break;

}

insert(head, tail, val);

}

long long int x;

cin >> x;

int index = 0, found = 0;

for (Node\* current = head; current != NULL; current = current->next) {

if (current->val == x) {

cout << index << endl;

found = 1;

break;

}

index++;

}

if (!found) {

cout << -1 << endl;

}

Node\* temp;

while (head != NULL) {

temp = head;

head = head->next;

delete temp;

}

}

return 0;

}

74. <https://www.hackerrank.com/contests/assignment-02-a-basic-data-structures-a-batch-06/challenges/same-to-same-1>

You will be given two singly linked list of integer values as input. You need to check if all the elements of both list are same which means both list are same. If they are same print "**YES**" otherwise print "**NO**".

**Note**: You must use singly linked list, otherwise you will not get marks.

**Input Format**

* First line will contain the values of the first singly linked list, and will terminate with -1.
* Second line will contain the values of the second singly linked list, and will terminate with -1.

**Constraints**

1. 1 <= **N1, N2** <= 1000; Here N1 and N2 is the maximum number of nodes of the first and second linked list.
2. 0 <= **V** <= 1000; Here V is the value of each node.

**Output Format**

* Output "**YES**" or "**NO**".

**Sample Input 0**

10 20 30 40 -1

10 20 30 40 -1

**Sample Output 0**

YES

**Sample Input 1**

10 20 30 40 -1

10 20 30 -1

**Sample Output 1**

NO

**Sample Input 2**

10 20 30 40 -1

40 30 20 10 -1

**Sample Output 2**

NO

#include <bits/stdc++.h>

using namespace std;

class Node{

public:

long long int val;

Node \* next;

Node(long long int val){

this->val=val;

this->next=NULL;

}

};

void insert(Node \* &head,Node \* &tail,long long int val){

Node \* newnode=new Node(val);

if(head==NULL){

tail=newnode;

head=newnode;

return;

}

tail->next=newnode;

tail=tail->next;

}

int main(){

Node \*head=NULL;

Node \*tail=NULL;

Node \*head1=NULL;

Node \*tail1=NULL;

long long int val;

int c1=0,c2=0;

while(true){

cin>>val;

c1++;

if(val==-1){

break;

}

insert(head,tail,val);

}

while(true){

cin>>val;

c2++;

if(val==-1){

break;

}

insert(head1,tail1,val);

}

if(c1!=c2){

cout<<"NO"<<endl;

return 0;

}

Node \*tmp=head;

Node \*tmp1=head1;

while(tmp!=NULL || tmp1!=NULL){

if(tmp->val!=tmp1->val){

cout<<"NO"<<endl;

return 0;

}

tmp=tmp->next;

tmp1=tmp1->next;

}

cout<<"YES"<<endl;

return 0;

}

75. <https://www.hackerrank.com/contests/assignment-02-a-basic-data-structures-a-batch-06/challenges/queries-6-1>

You have a singly linked list which is **empty** initially. Then you will be given **Q** queries. In each query you will be given two values **X** and **V**.

* If **X** is **0** that means you will insert the value **V** to the head of the linked list.
* If **X** is **1** then you will insert the value **V** to the tail of the linked list.
* If **X** is **2** then you will delete the value **Vth** index of the linked list. Assume that index starts from 0. If the index is invalid, then you shouldn't perform the deletion.
* After each query you need to print the linked list.

**Note**: You must use singly linked list, otherwise you will not get marks.

**Input Format**

* First line will contain **Q**.
* Next **Q** lines will contain **X** and **V**.

**Constraints**

1. 1 <= **Q** <= 1000;
2. 0 <= **X** <= 2;
3. 0 <= **V** <= 10^9

**Output Format**

* For each query ouput the updated linked list.

**Sample Input 0**

4

0 10

1 20

1 30

0 40

**Sample Output 0**

10

10 20

10 20 30

40 10 20 30

**Sample Input 1**

11

0 10

2 5

1 20

1 30

0 40

2 0

0 50

2 2

1 60

2 3

2 3

**Sample Output 1**

10

10

10 20

10 20 30

40 10 20 30

10 20 30

50 10 20 30

50 10 30

50 10 30 60

50 10 30

50 10 30

**Sample Input 2**

10

1 4

2 1

0 9

0 10

2 2

1 5

2 0

2 1

2 5

2 2

**Sample Output 2**

4

4

9 4

10 9 4

10 9

10 9 5

9 5

9

9

9

#include <bits/stdc++.h>

using namespace std;

class Node {

public:

long long int val;

Node\* next;

Node(long long int val) {

this->val = val;

this->next = NULL;

}

};

void insert\_tail(Node\*& head, Node\*& tail, long long int val) {

Node\* newnode = new Node(val);

if (tail == NULL) {

head = tail = newnode;

} else {

tail->next = newnode;

tail = newnode;

}

}

void insert\_head(Node\*& head, Node\*& tail, long long int val) {

Node\* newnode = new Node(val);

newnode->next = head;

head = newnode;

if (tail == NULL) {

tail = newnode;

}

}

void delete\_node(Node\*& head, Node\*& tail, long long int pos) {

if (head == NULL || pos < 0) {

return;

}

if (pos == 0) {

Node\* toDelete = head;

head = head->next;

if (head == NULL) {

tail = NULL;

}

delete toDelete;

return;

}

Node\* temp = head;

for (long long int i = 1; i < pos; i++) {

if (temp->next == NULL) {

return;

}

temp = temp->next;

}

if (temp->next == NULL) {

return;

}

Node\* toDelete = temp->next;

temp->next = toDelete->next;

if (toDelete == tail) {

tail = temp;

}

delete toDelete;

}

void print\_list(Node\* head) {

Node\* temp = head;

while (temp != NULL) {

cout << temp->val << " ";

temp = temp->next;

}

cout << endl;

}

int main() {

Node\* head = NULL;

Node\* tail = NULL;

int n;

cin >> n;

for (int i = 0; i < n; i++) {

long long int x, v;

cin >> x >> v;

if (x == 0) {

insert\_head(head, tail, v);

} else if (x == 1) {

insert\_tail(head, tail, v);

} else if (x == 2) {

delete\_node(head, tail, v);

}

print\_list(head);

}

return 0;

}

76. <https://www.hackerrank.com/contests/assignment-02-a-basic-data-structures-a-batch-06/challenges/remove-duplicate-9-1>

You will be given a singly linked list of integer values as input. You need to remove duplicate values from the linked list and finally print the linked list.

The process is, for each node **N**, traverse from that node and delete all nodes where the values are same with **N**.

**Note**: You must use singly linked list, otherwise you will not get marks.

**Input Format**

* First line will contain the values of the singly linked list, and will terminate with -1.

**Constraints**

1. 1 <= **N** <= 1000; Here N is the maximum number of nodes of the linked list.
2. 0 <= **V** <= 1000; Here V is the value of each node.

**Output Format**

* Output the final linked list where there will be no duplicate values.

**Sample Input 0**

1 2 3 4 5 -1

**Sample Output 0**

1 2 3 4 5

**Sample Input 1**

1 2 4 2 3 5 1 4 5 2 6 1 -1

**Sample Output 1**

1 2 4 3 5 6

**Sample Input 2**

5 5 1 1 2 4 2 4 1 3 5 0 -1

**Sample Output 2**

5 1 2 4 3 0

**Sample Input 3**

10 10 10 20 20 20 10 20 -1

**Sample Output 3**

10 20

#include <bits/stdc++.h>

using namespace std;

class Node {

public:

long long int val;

Node\* next;

Node(long long int val) {

this->val = val;

this->next = NULL;

}

};

void insert\_tail(Node\*& head, Node\*& tail, long long int val) {

Node\* newnode = new Node(val);

if (head == NULL) {

head = tail = newnode;

} else {

tail->next = newnode;

tail = newnode;

}

}

void print\_list(Node\* head) {

Node\* temp = head;

while (temp != NULL) {

cout << temp->val << " ";

temp = temp->next;

}

cout << endl;

}

void remove\_duplicates(Node\*& head) {

if (head == NULL) return;

unordered\_set<long long int> seen;

Node\* current = head;

Node\* prev = NULL;

while (current != NULL) {

if (seen.find(current->val) != seen.end()) {

prev->next = current->next;

delete current;

current = prev->next;

} else {

seen.insert(current->val);

prev = current;

current = current->next;

}

}

}

int main() {

Node\* head = NULL;

Node\* tail = NULL;

long long int val;

while (true) {

cin >> val;

if (val == -1) {

break;

}

insert\_tail(head, tail, val);

}

remove\_duplicates(head);

print\_list(head);

return 0;

}

77. <https://www.hackerrank.com/contests/mid-term-exam-a-basic-data-structures-a-batch-06/challenges/remove-duplicate-again>

You will be given a linked list of integer values as input. You need to remove duplicate values from the linked list and finally print the linked list in **ascending** order.

**Note**: You need to solve this using **STL List**, otherwise you will not get marks.

**Input Format**

* First line will contain the values of the linked list, and will terminate with -1.

**Constraints**

1. 1 <= **N** <= 1000; Here N is the maximum number of nodes of the linked list.
2. 0 <= **V** <= 1000; Here V is the value of each node.

**Output Format**

* Output the final linked list where there will be no duplicate values.

**Sample Input 0**

1 2 3 4 5 -1

**Sample Output 0**

1 2 3 4 5

**Sample Input 1**

1 2 4 2 3 5 1 4 5 2 6 1 -1

**Sample Output 1**

1 2 3 4 5 6

**Sample Input 2**

5 5 1 1 2 4 2 4 1 3 5 0 -1

**Sample Output 2**

0 1 2 3 4 5

**Sample Input 3**

10 10 10 20 20 20 10 20 -1

**Sample Output 3**

10 20

#include<bits/stdc++.h>

using namespace std;

int main(){

list<int>l;

int val;

while(true){

cin>>val;

if(val==-1){

break;

}

l.push\_back(val);

}

l.sort();

l.unique();

for(auto x:l){

cout<<x<<" ";

}

cout<<endl;

return 0;

}

78. <https://www.hackerrank.com/contests/mid-term-exam-a-basic-data-structures-a-batch-06/challenges/palindrome-26-2>

You need to take a doubly linked list of integer value as input. You need to tell if the doubly linked list forms a palindrome or not.

**Note**: You need to solve this using **Doubly Linked List**, otherwise you will not get marks.

**Input Format**

* Input will contain the values of the doubly linked list, and will terminate with -1.

**Constraints**

1. 1 <= **N** <= 10^6; Here N is the maximum number of nodes of the linked list.
2. 0 <= **V** <= 1000; Here V is the value of each node.

**Output Format**

* Output "**YES**" if it forms a palindrom otherwise output "**NO**".

**Sample Input 0**

1 2 3 2 1 -1

**Sample Output 0**

YES

**Sample Input 1**

1 2 2 1 -1

**Sample Output 1**

YES

**Sample Input 2**

1 -1

**Sample Output 2**

YES

**Sample Input 3**

1 2 3 1 -1

**Sample Output 3**

NO

#include<bits/stdc++.h>

using namespace std;

class Node{

public:

Node \*next;

int val;

Node \*pre;

Node(int val){

this->val=val;

this->next=NULL;

this->pre=NULL;

}

};

void insert\_at\_tail(Node \*&head,Node \*&tail,int val){

Node \*newnode=new Node(val);

if(head==NULL){

head=newnode;

tail=newnode;

return;

}

tail->next=newnode;

newnode->pre=tail;

tail=newnode;

}

int main(){

Node \* head=NULL;

Node \*tail=NULL;

int val;

while(true){

cin>>val;

if(val==-1){

break;

}

insert\_at\_tail(head,tail,val);

}

Node \*tmp1=head;

Node \*tmp2=tail;

while(tmp1!=tmp2 && tmp1!=tmp2->next){

//cout<<"tmp1 & tmp2 : "<<tmp1->val<<" "<<tmp2->val<<endl;

if(tmp1->val!=tmp2->val){

cout<<"NO"<<endl;

return 0;

}

tmp1=tmp1->next;

tmp2=tmp2->pre;

}

cout<<"YES"<<endl;

return 0;

}

79. <https://www.hackerrank.com/contests/mid-term-exam-a-basic-data-structures-a-batch-06/challenges/browser-history-1>

You are given a doubly linked list of unique string values. These strings refer to web **addresses** without any spaces. You will be given Q queries. In each query you will be given some commands. Type of commands are -

1. **visit address** - You need to go to that address from where you are in that list and print that **address** if it is in the list. Otherwise print "**Not Available**".
2. **next** - You need to go to the next address from where you are in that list and print that **address** if it is in the list. Otherwise print "**Not Available**".
3. **prev** - You need to go to the previous address from where you are in that list and print that **address** if it is in the list. Otherwise print "**Not Available**".

One more thing, if the address isn't available make sure you don't move from your current position. You are at the head initially.

**Note**: You can use **Singly/Doubly Linked List** or **STL List** to solve this problem.

**Input Format**

* First line will contain the values of the doubly linked list, and will terminate with the string "**end**".
* Second line will contain **Q**.
* Next Q lines will contain the commands. It is guranteed that you will get "**visit address**" command at first which will contain a valid address. It will not contain valid address everytime!

**Constraints**

1. 1 <= **N** <= 1000; Here **N** is the maximum number of nodes of the linked list.
2. 1 <= **Q** <= 1000;
3. 1 <= |Address| <= 100; Here |Address| is the length of the string address.

**Output Format**

* For each query output as asked.

**Sample Input 0**

facebook google phitron youtube twitter end

12

visit phitron

prev

prev

prev

prev

next

visit twitter

next

next

prev

visit django

prev

**Sample Output 0**

phitron

google

facebook

Not Available

Not Available

google

twitter

Not Available

Not Available

youtube

Not Available

phitron

**Sample Input 1**

a b c d e f g h i j k l m n o

p q r s t u v w x y z end

7

visit s

next

visit zz

next

visit z

next

prev

**Sample Output 1**

s

t

Not Available

u

z

Not Available

y

#include<bits/stdc++.h>

using namespace std;

class Node{

public:

Node \*next;

string val;

Node \*pre;

Node(string val){

this->val=val;

this->next=NULL;

this->pre=NULL;

}

};

void insert\_at\_tail(Node \*&head,Node \*&tail,string val){

Node \*newnode=new Node(val);

if(head==NULL){

head=newnode;

tail=newnode;

return;

}

tail->next=newnode;

newnode->pre=tail;

tail=newnode;

}

int main(){

Node \* head=NULL;

Node \*tail=NULL;

string val;

while(true){

cin>>val;

if(val=="end"){

break;

}

insert\_at\_tail(head,tail,val);

}

int n;

cin>>n;

Node \*tmp=head;

for(int i=0;i<n;i++){

string s;

cin>>s;

Node \*t=tmp;

if(s=="visit")

{ tmp=head;

string s1;

cin>>s1;

int f=0;

while(tmp){

if(tmp->val==s1){

f=1;

break;

}

else{

tmp=tmp->next;

}

}

if(f==1){

cout<<s1<<endl;

}

else{

cout<<"Not Available"<<endl;

tmp=t;

}

}

else if(s=="prev"){

if(tmp->pre==NULL){

cout<<"Not Available"<<endl;

}

else{

tmp=tmp->pre;

cout<<tmp->val<<endl;

}

}

else{

if(tmp->next==NULL){

cout<<"Not Available"<<endl;

}

else{

tmp=tmp->next;

cout<<tmp->val<<endl;

}

}

}

return 0;

}

80. <https://www.hackerrank.com/contests/mid-term-exam-a-basic-data-structures-a-batch-06/challenges/queries-again-1>

You have a doubly linked list which is **empty** initially. Then you will be given **Q** queries. In each query you will be given two values **X** and **V**.

* You need to insert the value **V** at index **X**. Assume that index starts from 0.
* After that for each query you need to print the linked list from left to right and right to left.
* If the index is invalid, then print "Invalid".

**Note**: You must use **Doubly Linked List**, otherwise you will not get marks.

**Input Format**

* First line will contain **Q**.
* Next **Q** lines will contain **X** and **V**.

**Constraints**

1. 1 <= **Q** <= 1000;
2. 0 <= **X** <= 1000;
3. 0 <= **V** <= 1000

**Output Format**

* For each query print the linked list from left to right and right to left or print "Invalid" as asked.
* Print "**L ->** " before printing the linked list from left to right.
* Print "**R ->** " before printing the linked list from right to left.

**Sample Input 0**

5

1 10

0 10

1 20

3 30

2 30

**Sample Output 0**

Invalid

L -> 10

R -> 10

L -> 10 20

R -> 20 10

Invalid

L -> 10 20 30

R -> 30 20 10

**Sample Input 1**

10

0 10

1 20

0 30

1 40

6 50

0 60

4 70

4 80

2 90

1 100

**Sample Output 1**

L -> 10

R -> 10

L -> 10 20

R -> 20 10

L -> 30 10 20

R -> 20 10 30

L -> 30 40 10 20

R -> 20 10 40 30

Invalid

L -> 60 30 40 10 20

R -> 20 10 40 30 60

L -> 60 30 40 10 70 20

R -> 20 70 10 40 30 60

L -> 60 30 40 10 80 70 20

R -> 20 70 80 10 40 30 60

L -> 60 30 90 40 10 80 70 20

R -> 20 70 80 10 40 90 30 60

L -> 60 100 30 90 40 10 80 70 20

R -> 20 70 80 10 40 90 30 100 60

#include<bits/stdc++.h>

using namespace std;

class Node{

public:

Node \*next;

int val;

Node \*pre;

Node(int val){

this->val=val;

this->next=NULL;

this->pre=NULL;

}

};

void insert\_at\_tail(Node \*&head,Node \*&tail,int val){

Node \*newnode=new Node(val);

if(head==NULL){

head=newnode;

tail=newnode;

return;

}

tail->next=newnode;

newnode->pre=tail;

tail=newnode;

}

void insert\_at\_any(Node \*&head,int pos,int val){

Node \*newnode=new Node(val);

Node \*tmp=head;

for(int i=1;i<pos;i++){

tmp=tmp->next;

}

newnode->next=tmp->next;

tmp->next->pre=newnode;

tmp->next=newnode;

newnode->pre=tmp;

}

void insert\_at\_head(Node \*&head, Node \*&tail, int val) {

Node \*newnode = new Node(val);

if (head == NULL) {

head = newnode;

tail = newnode;

return;

}

newnode->next = head;

head->pre = newnode;

head = newnode;

}

int main(){

Node \* head=NULL;

Node \*tail=NULL;

int n;

cin>>n;

int t=0;

for(int i=0;i<n;i++){

int x,v;

cin>>x>>v;

if(x>t || x<0){

cout<<"Invalid"<<endl;

}

else

{if(x==0 && head==NULL){

insert\_at\_tail(head,tail,v);

t++;

}

else if(x==0){

insert\_at\_head(head,tail,v);

t++;

}

else if(x==t){

insert\_at\_tail(head,tail,v);

t++;

}

else{

insert\_at\_any(head,x,v);

t++;

}

Node \* tmp=head;

cout<<"L -> ";

while(tmp){

cout<<tmp->val<<" ";

tmp=tmp->next;

}

cout<<endl;

tmp=tail;

cout<<"R -> ";

while(tmp){

cout<<tmp->val<<" ";

tmp=tmp->pre;

}

cout<<endl;

}

}

return 0;

}

81. <https://www.hackerrank.com/contests/mid-term-exam-a-basic-data-structures-a-batch-06/challenges/remember-previous-queries>

You have a doubly linked list which is **empty** initially. Then you will be given **Q** queries. In each query you will be given two values **X** and **V**.

* If **X** is **0** that means you will insert the value **V** to the head of the linked list.
* If **X** is **1** then you will insert the value **V** to the tail of the linked list.
* If **X** is **2** then you will delete the value **Vth** index of the linked list. Assume that index starts from 0. If the index is invalid, then you shouldn't perform the deletion.
* After each query you need to print the linked list from both left to right and right to left.

**Note**: You must use **STL List**, otherwise you will not get marks.

**Input Format**

* First line will contain **Q**.
* Next **Q** lines will contain **X** and **V**.

**Constraints**

1. 1 <= **Q** <= 1000;
2. 0 <= **X** <= 2;
3. 0 <= **V** <= 10^9

**Output Format**

* For each query print the linked list from left to right and right to left.
* Print "**L ->** " before printing the linked list from left to right.
* Print "**R ->** " before printing the linked list from right to left.

**Sample Input 0**

4

0 10

1 20

1 30

0 40

**Sample Output 0**

L -> 10

R -> 10

#include<bits/stdc++.h>

using namespace std;

int main(){

list<int>l;

int n;

cin>>n;

for(int i=0;i<n;i++){

int x,y;

cin>>x>>y;

if(x==0){

l.push\_front(y);

}

else if(x==1){

l.push\_back(y);

}

else{

if(l.size()!=0 && l.size()>y)

l.erase(next(l.begin(),y));

}

cout<<"L ->";

for(auto i:l){

cout<<" "<<i;

}

cout<<endl;

l.reverse();

cout<<"R ->";

for(auto i:l){

cout<<" "<<i;

}

cout<<endl;

l.reverse();

}

return 0;

}

L -> 10 20

R -> 20 10

L -> 10 20 30

R -> 30 20 10

L -> 40 10 20 30

R -> 30 20 10 40

**Sample Input 1**

9

0 10

2 1

2 0

1 20

0 10

2 2

2 1

2 2

2 0

**Sample Output 1**

L -> 10

R -> 10

L -> 10

R -> 10

L ->

R ->

L -> 20

R -> 20

L -> 10 20

R -> 20 10

L -> 10 20

R -> 20 10

L -> 10

R -> 10

L -> 10

R -> 10

L ->

R ->

**Sample Input 2**

11

0 10

2 5

1 20

1 30

0 40

2 0

0 50

2 2

1 60

2 3

2 3

**Sample Output 2**

L -> 10

R -> 10

L -> 10

R -> 10

L -> 10 20

R -> 20 10

L -> 10 20 30

R -> 30 20 10

L -> 40 10 20 30

R -> 30 20 10 40

L -> 10 20 30

R -> 30 20 10

L -> 50 10 20 30

R -> 30 20 10 50

L -> 50 10 30

R -> 30 10 50

L -> 50 10 30 60

R -> 60 30 10 50

L -> 50 10 30

R -> 30 10 50

L -> 50 10 30

R -> 30 10 50

**Sample Input 3**

10

1 4

2 1

0 9

0 10

2 2

1 5

2 0

2 1

2 5

2 2

**Sample Output 3**

L -> 4

R -> 4

L -> 4

R -> 4

L -> 9 4

R -> 4 9

L -> 10 9 4

R -> 4 9 10

L -> 10 9

R -> 9 10

L -> 10 9 5

R -> 5 9 10

L -> 9 5

R -> 5 9

L -> 9

R -> 9

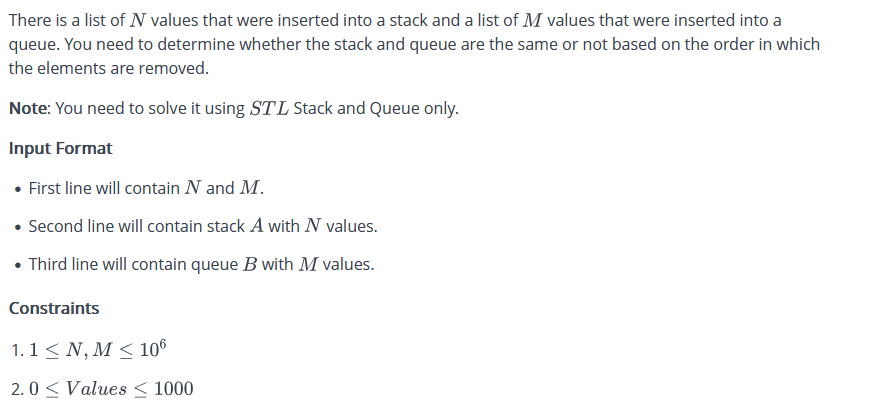
L -> 9

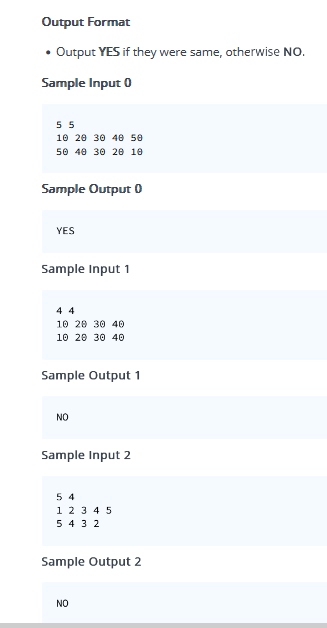
R -> 9

L -> 9

R -> 9

82. <https://www.hackerrank.com/contests/assignment-03-a-basic-data-structure-a-batch-06/challenges/same-or-not-ii>





#include <bits/stdc++.h>

using namespace std;

int main()

{

int m,n;

cin>>m>>n;

stack<int>s;

queue<int>q;

for(int i=0;i<m;i++){

int x;

cin>>x;

s.push(x);

}

for(int i=0;i<n;i++){

int x;

cin>>x;

q.push(x);

}

if(m!=n){

cout<<"NO"<<endl;

return 0;

}

for(int i=0;i<m;i++){

int x1=s.top();

int x2=q.front();

// cout<<"x1 & x2 are "<<x1<<" "<<x2<<endl;

s.pop();

q.pop();

if(x2!=x1){

cout<<"NO"<<endl;

return 0;

}

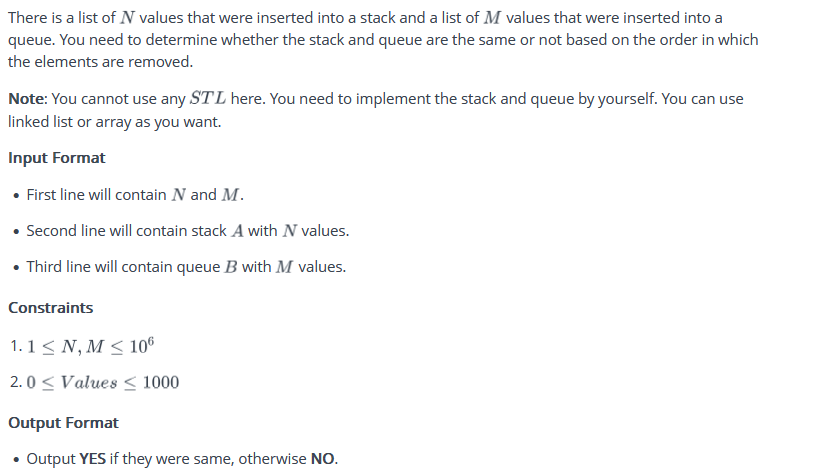
}

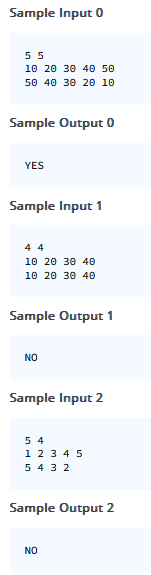
cout<<"YES"<<endl;

return 0;

}

83. <https://www.hackerrank.com/contests/assignment-03-a-basic-data-structure-a-batch-06/challenges/same-or-not-4>





#include <bits/stdc++.h>

using namespace std;

class Node{

public:

int val;

Node \* next;

Node \*pre;

Node(int val){

this->val=val;

this->next=NULL;

this->pre=NULL;

}

};

class que{

public:

Node \*head=NULL;

Node \*tail=NULL;

void push(int x){

Node \* newnode=new Node(x);

if(head==NULL){

head=newnode;

tail=newnode;

}

tail->next=newnode;

newnode->pre=tail;

tail=newnode;

}

void pop(){

if(head == NULL) return;

Node \*dlt = head;

head = head->next;

delete dlt;

if(head == NULL){

tail = NULL;

} else {

head->pre = NULL;

}

}

int front(){

return head->val;

}

};

class stk{

public:

Node \*head=NULL;

Node \*tail=NULL;

void push(int x){

Node \* newnode=new Node(x);

if(head==NULL){

head=newnode;

tail=newnode;

}

tail->next=newnode;

newnode->pre=tail;

tail=newnode;

}

void pop(){

if(tail == NULL) return;

Node \*dlt = tail;

tail = tail->pre;

delete dlt;

if(tail == NULL){

head = NULL;

} else {

tail->next = NULL;

}

}

int top(){

return tail->val;

}

};

int main()

{

int m,n;

cin>>m>>n;

stk s;

que q;

for(int i=0;i<m;i++){

int x;

cin>>x;

s.push(x);

}

for(int i=0;i<n;i++){

int x;

cin>>x;

q.push(x);

}

if(m!=n){

cout<<"NO"<<endl;

return 0;

}

for(int i=0;i<m;i++){

int x1=s.top();

int x2=q.front();

// cout<<"x1 & x2 are "<<x1<<" "<<x2<<endl;

s.pop();

q.pop();

if(x2!=x1){

cout<<"NO"<<endl;

return 0;

}

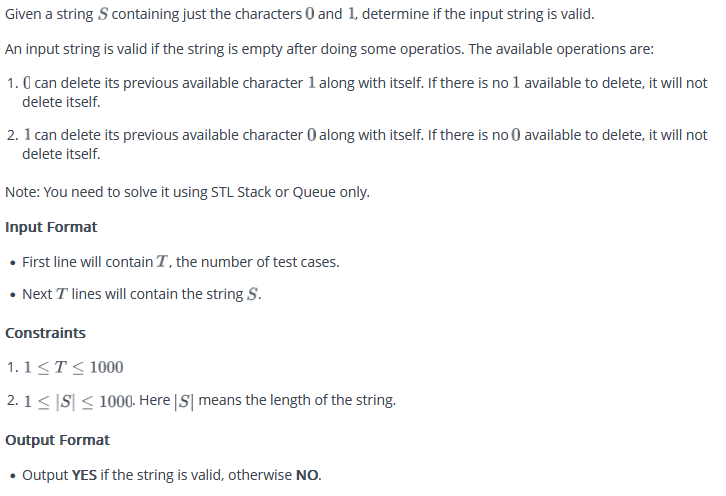
}

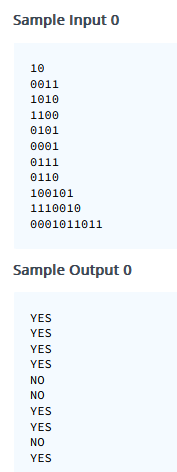
cout<<"YES"<<endl;

return 0;

}

85. <https://www.hackerrank.com/contests/assignment-03-a-basic-data-structure-a-batch-06/challenges/is-it-valid-1-1>





#include <bits/stdc++.h>

using namespace std;

int main()

{

int n;

cin>>n;

for(int j=0;j<n;j++){

string s;

cin>>s;

stack<char>st;

for(char i:s){

if(!st.empty() &&( (i=='0' && st.top()=='1') ||(i=='1' && st.top()=='0'))){

st.pop();

}

else{

st.push(i);

}

}

if(st.empty()){

cout<<"YES"<<endl;

}

else{

cout<<"NO"<<endl;

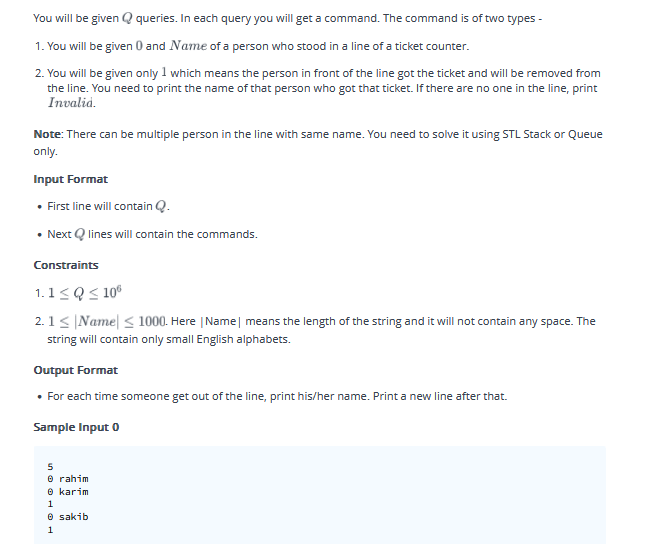
}

}

return 0;

}

86. <https://www.hackerrank.com/contests/assignment-03-a-basic-data-structure-a-batch-06/challenges/special-queries-1-1>



#include <bits/stdc++.h>

using namespace std;

int main()

{

int n;

cin>>n;

queue<string>s;

while(n--){

int x;

cin>>x;

if(x==0){

string y;

cin>>y;

s.push(y);

}

else{

if(!s.empty())

{

string v=s.front();

s.pop();

cout<<v<<endl;

}

else{

cout<<"Invalid"<<endl;

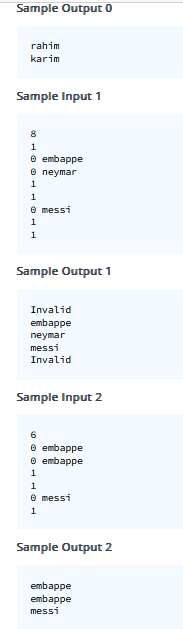
}

}

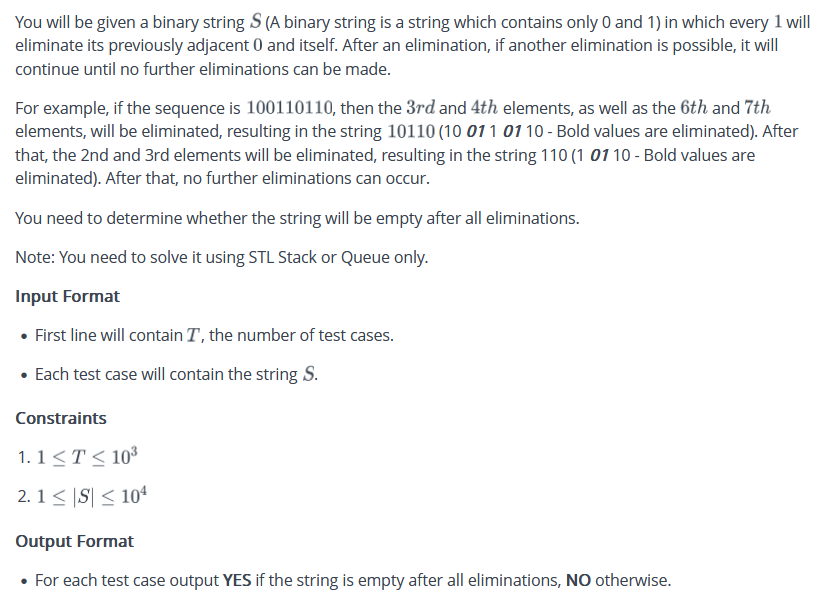
}

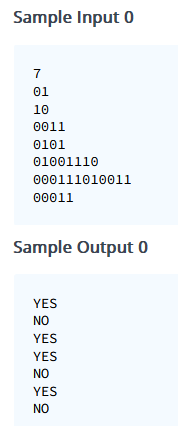
return 0;

}



87. <https://www.hackerrank.com/contests/assignment-03-a-basic-data-structure-a-batch-06/challenges/elimination-2-2>





#include <bits/stdc++.h>

using namespace std;

int main()

{

int n;

cin>>n;

while(n--){

string s;

cin>>s;

stack<char>st;

for(char i:s){

if(!st.empty() && (st.top()=='0' && i=='1' )){

st.pop();

}

else{

st.push(i);

}

}

if(st.empty())cout<<"YES"<<endl;

else{

cout<<"NO"<<endl;

}

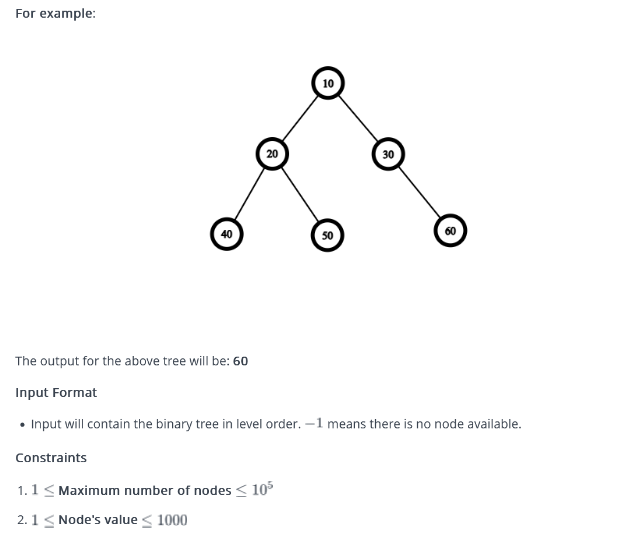
}

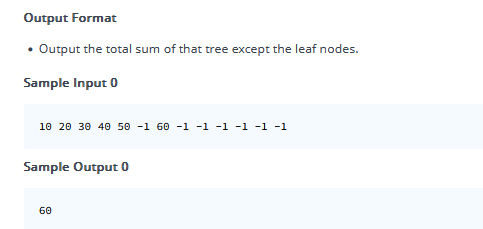
return 0;

}

88. <https://www.hackerrank.com/contests/assignment-04-a-basic-data-structure-a-batch-06/challenges/sum-without-leaf>

You will be given a binary tree as input in level order. You need to output the sum of all node's values in that tree except the leaf nodes





#include <bits/stdc++.h>

using namespace std;

class Node {

public:

int val;

Node \*left;

Node \*right;

Node(int val) {

this->val = val;

this->left = NULL;

this->right = NULL;

}

};

Node\* input\_tree() {

int val;

cin >> val;

if (val == -1) {

return NULL;

}

Node\* root = new Node(val);

queue<Node\*> q;

q.push(root);

while (!q.empty()) {

Node\* p = q.front();

q.pop();

int l, r;

cin >> l >> r;

Node\* myleft = (l == -1) ? NULL : new Node(l);

Node\* myright = (r == -1) ? NULL : new Node(r);

p->left = myleft;

p->right = myright;

if (p->left) q.push(p->left);

if (p->right) q.push(p->right);

}

return root;

}

int summ(Node\* root) {

if (root == NULL) {

return 0;

}

int sum = 0;

queue<Node\*> q;

q.push(root);

while (!q.empty()) {

Node\* f = q.front();

q.pop();

// Check if the node is not a leaf

if (f->left != NULL || f->right != NULL) {

sum += f->val;

}

if (f->left) q.push(f->left);

if (f->right) q.push(f->right);

}

return sum;

}

int main() {

Node\* root = input\_tree();

int sum = summ(root);

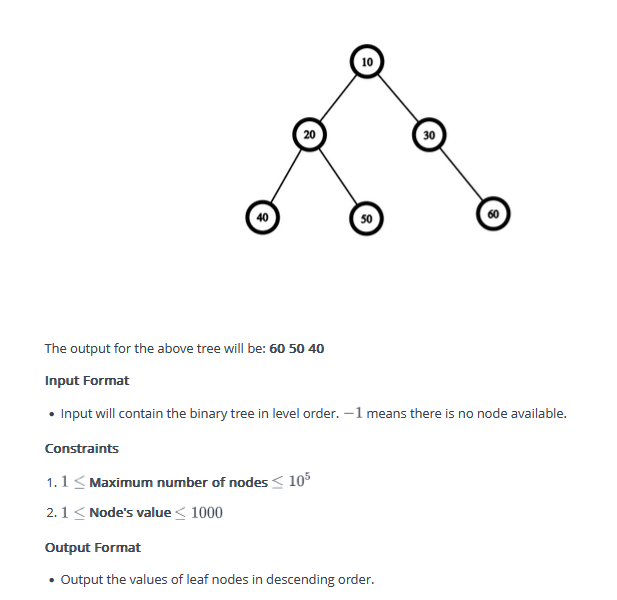
cout << sum << endl;

return 0;

}

89. <https://www.hackerrank.com/contests/assignment-04-a-basic-data-structure-a-batch-06/challenges/leaf-nodes-1>

You will be given a binary tree as input in level order. You need to print the values of leaf nodes in descending order.



**Sample Input 0**

10 20 30 40 50 -1 60 -1 -1 -1 -1 -1 -1

**Sample Output 0**

60 50 40

#include <bits/stdc++.h>

using namespace std;

class Node {

public:

int val;

Node \*left;

Node \*right;

Node(int val) {

this->val = val;

this->left = NULL;

this->right = NULL;

}

};

Node\* input\_tree() {

int val;

cin >> val;

if (val == -1) {

return NULL;

}

Node\* root = new Node(val);

queue<Node\*> q;

q.push(root);

while (!q.empty()) {

Node\* p = q.front();

q.pop();

int l, r;

cin >> l >> r;

Node\* myleft = (l == -1) ? NULL : new Node(l);

Node\* myright = (r == -1) ? NULL : new Node(r);

p->left = myleft;

p->right = myright;

if (p->left) q.push(p->left);

if (p->right) q.push(p->right);

}

return root;

}

vector<int> dec(Node\* root) {

vector<int>v;

if (root == NULL) {

return v;

}

queue<Node\*> q;

q.push(root);

while (!q.empty()) {

Node\* f = q.front();

q.pop();

if (f->left == NULL && f->right == NULL) {

v.push\_back(f->val);

}

if (f->left) q.push(f->left);

if (f->right) q.push(f->right);

}

return v;

}

int main() {

Node\* root = input\_tree();

vector<int>v;

v=dec(root);

sort(v.begin(),v.end(),greater<int>());

for(auto i:v){

cout<<i<<" ";

}

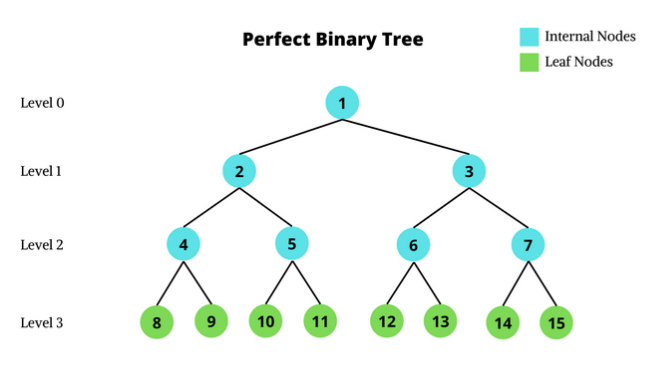
cout<<endl;

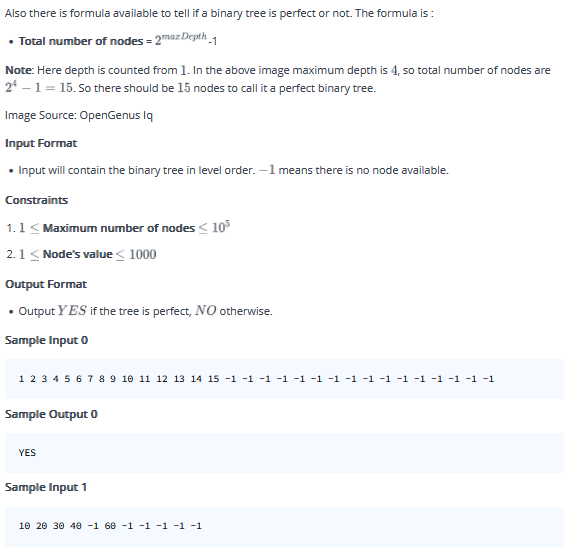
return 0;

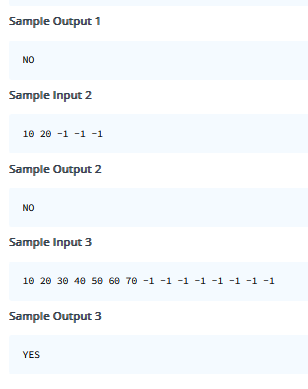
}

90. <https://www.hackerrank.com/contests/assignment-04-a-basic-data-structure-a-batch-06/challenges/perfect-binary-tree>

You will be given a binary tree as input in level order. You need to tell if the binary tree is perfect or not. A binary tree is called perfect if all leaf nodes are at the maximum depth of the tree, and the tree is completely filled with no gaps.**Here is an example of perfect binary tree**:







#include <bits/stdc++.h>

using namespace std;

class Node {

public:

int val;

Node \*left;

Node \*right;

Node(int val) {

this->val = val;

this->left = NULL;

this->right = NULL;

}

};

Node\* input\_tree() {

int val;

cin >> val;

if (val == -1) {

return NULL;

}

Node\* root = new Node(val);

queue<Node\*> q;

q.push(root);

while (!q.empty()) {

Node\* p = q.front();

q.pop();

int l, r;

cin >> l >> r;

Node\* myleft = (l == -1) ? NULL : new Node(l);

Node\* myright = (r == -1) ? NULL : new Node(r);

p->left = myleft;

p->right = myright;

if (p->left) q.push(p->left);

if (p->right) q.push(p->right);

}

return root;

}

void perfect(Node\* root) {

if (root == NULL) {

return ;

}

queue<Node\*> q;

q.push(root);

while (!q.empty()) {

Node\* f = q.front();

q.pop();

if ((f->left!=NULL && f->right== NULL) ||(f->left==NULL && f->right!= NULL)) {

cout<<"NO"<<endl;

return;

}

if (f->left) q.push(f->left);

if (f->right) q.push(f->right);

}

cout<<"YES"<<endl;

}

int main() {

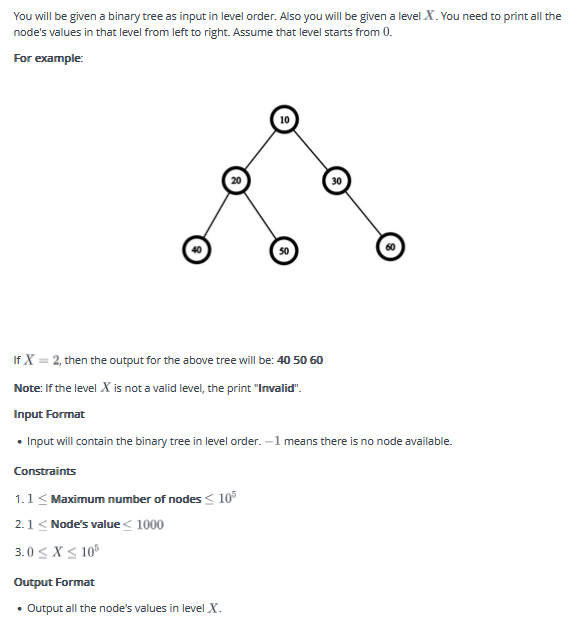
Node\* root = input\_tree();

perfect(root);

return 0;

}

91. <https://www.hackerrank.com/contests/assignment-04-a-basic-data-structure-a-batch-06/challenges/level-nodes-2>





#include <bits/stdc++.h>

using namespace std;

class Node {

public:

int val;

Node \*left;

Node \*right;

Node(int val) {

this->val = val;

this->left = NULL;

this->right = NULL;

}

};

Node\* input\_tree() {

int val;

cin >> val;

if (val == -1) {

return NULL;

}

Node\* root = new Node(val);

queue<Node\*> q;

q.push(root);

while (!q.empty()) {

Node\* p = q.front();

q.pop();

int l, r;

cin >> l >> r;

Node\* myleft = (l == -1) ? NULL : new Node(l);

Node\* myright = (r == -1) ? NULL : new Node(r);

p->left = myleft;

p->right = myright;

if (p->left) q.push(p->left);

if (p->right) q.push(p->right);

}

return root;

}

vector<int> lvl(Node\* root,int x) {

vector<int>v;

if (root == NULL) {

return v;

}

int n;

int c=-1;

queue<Node\*> q;

q.push(root);

while (!q.empty()) {

c++;

n=q.size();

for(int i=0;i<n;i++){

Node \*f=q.front();

q.pop();

if(c==x){

v.push\_back(f->val);

}

if(f->left)q.push(f->left);

if(f->right)q.push(f->right);

}

}

if(c<x){

vector<int>v1;

v1.push\_back(-1);

return v1 ;

}

return v;

}

int main() {

Node\* root = input\_tree();

vector<int>v;

int x;

cin>>x;

v=lvl(root,x);

for(auto i:v){

if(i==-1){

cout<<"Invalid"<<endl;

return 0;

}

cout<<i<<" ";

}

cout<<endl;

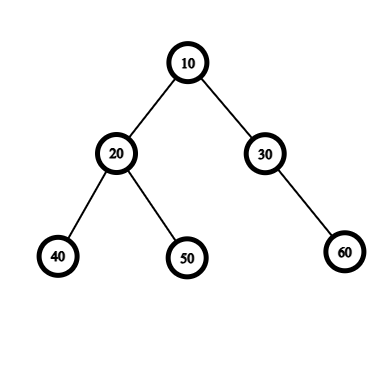
return 0;

}

92. <https://www.hackerrank.com/contests/assignment-04-a-basic-data-structure-a-batch-06/challenges/sum-without-leaf>

You will be given a binary tree as input in level order. You need to output the sum of all node's values in that tree except the leaf nodes.

**For example**:



The output for the above tree will be: **60**

**Input Format**

* Input will contain the binary tree in level order.
* means there is no node available.

**Constraints**

**Maximum number of nodes** **Node's value**

**Output Format**

* Output the total sum of that tree except the leaf nodes.

**Sample Input 0**

10 20 30 40 50 -1 60 -1 -1 -1 -1 -1 -1

**Sample Output 0**

60

#include <bits/stdc++.h>

using namespace std;

class Node {

public:

int val;

Node \*left;

Node \*right;

Node(int val) {

this->val = val;

this->left = NULL;

this->right = NULL;

}

};

Node\* input\_tree() {

int val;

cin >> val;

if (val == -1) {

return NULL;

}

Node\* root = new Node(val);

queue<Node\*> q;

q.push(root);

while (!q.empty()) {

Node\* p = q.front();

q.pop();

int l, r;

cin >> l >> r;

Node\* myleft = (l == -1) ? NULL : new Node(l);

Node\* myright = (r == -1) ? NULL : new Node(r);

p->left = myleft;

p->right = myright;

if (p->left) q.push(p->left);

if (p->right) q.push(p->right);

}

return root;

}

int summ(Node\* root) {

if (root == NULL) {

return 0;

}

int sum = 0;

queue<Node\*> q;

q.push(root);

while (!q.empty()) {

Node\* f = q.front();

q.pop();

// Check if the node is not a leaf

if (f->left != NULL || f->right != NULL) {

sum += f->val;

}

if (f->left) q.push(f->left);

if (f->right) q.push(f->right);

}

return sum;

}

int main() {

Node\* root = input\_tree();

int sum = summ(root);

cout << sum << endl;

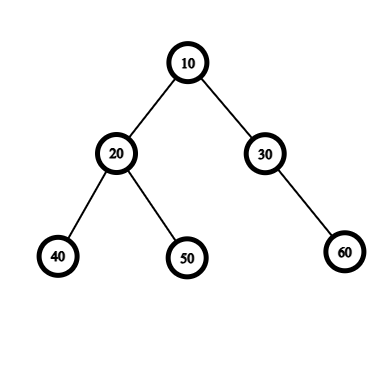
return 0;

}

93. <https://www.hackerrank.com/contests/assignment-04-a-basic-data-structure-a-batch-06/challenges/leaf-nodes-1>

You will be given a binary tree as input in level order. You need to print the values of leaf nodes in descending order.

**For example**:



The output for the above tree will be: **60 50 40**

**Input Format**

* Input will contain the binary tree in level order.
* means there is no node available.

**Constraints**

**Maximum number of nodes** **Node's value**

**Output Format**

* Output the values of leaf nodes in descending order.

**Sample Input 0**

10 20 30 40 50 -1 60 -1 -1 -1 -1 -1 -1

**Sample Output 0**

60 50 40

#include <bits/stdc++.h>

using namespace std;

class Node {

public:

int val;

Node \*left;

Node \*right;

Node(int val) {

this->val = val;

this->left = NULL;

this->right = NULL;

}

};

Node\* input\_tree() {

int val;

cin >> val;

if (val == -1) {

return NULL;

}

Node\* root = new Node(val);

queue<Node\*> q;

q.push(root);

while (!q.empty()) {

Node\* p = q.front();

q.pop();

int l, r;

cin >> l >> r;

Node\* myleft = (l == -1) ? NULL : new Node(l);

Node\* myright = (r == -1) ? NULL : new Node(r);

p->left = myleft;

p->right = myright;

if (p->left) q.push(p->left);

if (p->right) q.push(p->right);

}

return root;

}

vector<int> dec(Node\* root) {

vector<int>v;

if (root == NULL) {

return v;

}

queue<Node\*> q;

q.push(root);

while (!q.empty()) {

Node\* f = q.front();

q.pop();

if (f->left == NULL && f->right == NULL) {

v.push\_back(f->val);

}

if (f->left) q.push(f->left);

if (f->right) q.push(f->right);

}

return v;

}

int main() {

Node\* root = input\_tree();

vector<int>v;

v=dec(root);

sort(v.begin(),v.end(),greater<int>());

for(auto i:v){

cout<<i<<" ";

}

cout<<endl;

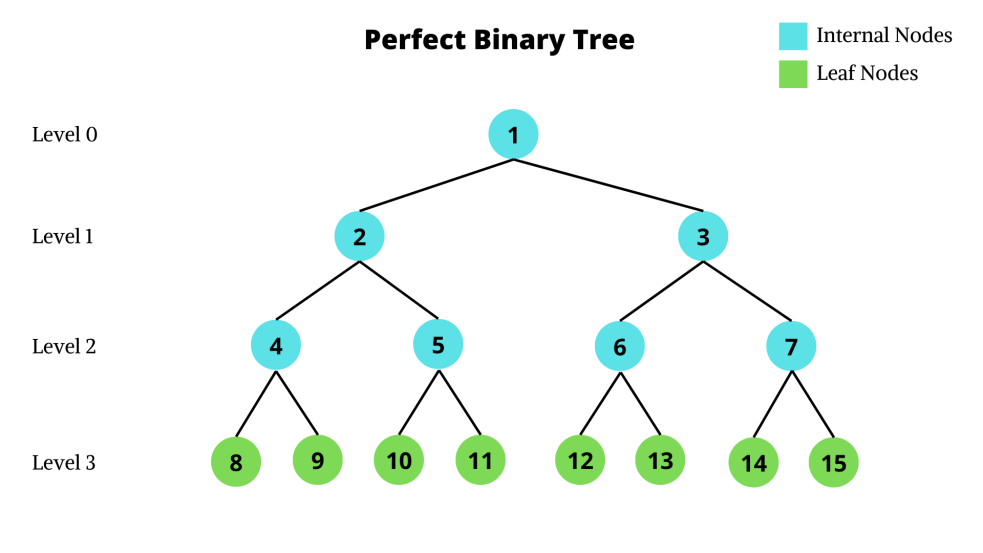
return 0;

}

94. <https://www.hackerrank.com/contests/assignment-04-a-basic-data-structure-a-batch-06/challenges/perfect-binary-tree>

You will be given a binary tree as input in level order. You need to tell if the binary tree is perfect or not. A binary tree is called perfect if all leaf nodes are at the maximum depth of the tree, and the tree is completely filled with no gaps.

**Here is an example of perfect binary tree**:



Also there is formula available to tell if a binary tree is perfect or not. The formula is :

* **Total number of nodes** =
* -1

**Note**: Here depth is counted from

. In the above image maximum depth is , so total number of nodes are . So there should be

nodes to call it a perfect binary tree.

Image Source: OpenGenus Iq

**Input Format**

* Input will contain the binary tree in level order.
* means there is no node available.

**Constraints**

**Maximum number of nodes** **Node's value**

**Output Format**

* Output

if the tree is perfect,

* otherwise.

**Sample Input 0**

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1

**Sample Output 0**

YES

**Sample Input 1**

10 20 30 40 -1 60 -1 -1 -1 -1 -1

**Sample Output 1**

NO

**Sample Input 2**

10 20 -1 -1 -1

**Sample Output 2**

NO

**Sample Input 3**

10 20 30 40 50 60 70 -1 -1 -1 -1 -1 -1 -1 -1

**Sample Output 3**

YES

#include <bits/stdc++.h>

using namespace std;

class Node {

public:

int val;

Node \*left;

Node \*right;

Node(int val) {

this->val = val;

this->left = NULL;

this->right = NULL;

}

};

Node\* input\_tree() {

int val;

cin >> val;

if (val == -1) {

return NULL;

}

Node\* root = new Node(val);

queue<Node\*> q;

q.push(root);

while (!q.empty()) {

Node\* p = q.front();

q.pop();

int l, r;

cin >> l >> r;

Node\* myleft = (l == -1) ? NULL : new Node(l);

Node\* myright = (r == -1) ? NULL : new Node(r);

p->left = myleft;

p->right = myright;

if (p->left) q.push(p->left);

if (p->right) q.push(p->right);

}

return root;

}

void perfect(Node\* root) {

if (root == NULL) {

return ;

}

queue<Node\*> q;

q.push(root);

while (!q.empty()) {

Node\* f = q.front();

q.pop();

if ((f->left!=NULL && f->right== NULL) ||(f->left==NULL && f->right!= NULL)) {

cout<<"NO"<<endl;

return;

}

if (f->left) q.push(f->left);

if (f->right) q.push(f->right);

}

cout<<"YES"<<endl;

}

int main() {

Node\* root = input\_tree();

perfect(root);

return 0;

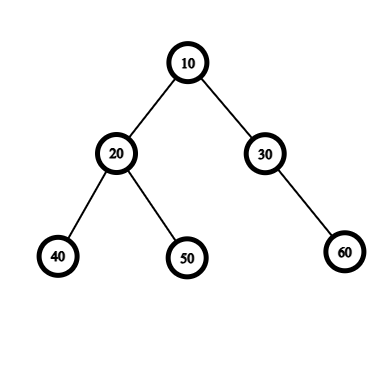
}

95. <https://www.hackerrank.com/contests/assignment-04-a-basic-data-structure-a-batch-06/challenges/level-nodes-2>

You will be given a binary tree as input in level order. Also you will be given a level . You need to print all the node's values in that level from left to right. Assume that level starts from

.

**For example**:



If

, then the output for the above tree will be: **40 50 60**

**Note**: If the level

is not a valid level, the print "**Invalid**".

**Input Format**

* Input will contain the binary tree in level order.
* means there is no node available.

**Constraints**

**Maximum number of nodes** **Node's value**

**Output Format**

* Output all the node's values in level
* .

**Sample Input 0**

10 20 30 40 50 -1 60 -1 -1 -1 -1 -1 -1

0

**Sample Output 0**

10

**Sample Input 1**

10 20 30 40 50 -1 60 -1 -1 -1 -1 -1 -1

1

**Sample Output 1**

20 30

**Sample Input 2**

10 20 30 40 50 -1 60 -1 -1 -1 -1 -1 -1

2

**Sample Output 2**

40 50 60

**Sample Input 3**

10 20 30 40 50 -1 60 -1 -1 -1 -1 -1 -1

3

**Sample Output 3**

Invalid

#include <bits/stdc++.h>

using namespace std;

class Node {

public:

int val;

Node \*left;

Node \*right;

Node(int val) {

this->val = val;

this->left = NULL;

this->right = NULL;

}

};

Node\* input\_tree() {

int val;

cin >> val;

if (val == -1) {

return NULL;

}

Node\* root = new Node(val);

queue<Node\*> q;

q.push(root);

while (!q.empty()) {

Node\* p = q.front();

q.pop();

int l, r;

cin >> l >> r;

Node\* myleft = (l == -1) ? NULL : new Node(l);

Node\* myright = (r == -1) ? NULL : new Node(r);

p->left = myleft;

p->right = myright;

if (p->left) q.push(p->left);

if (p->right) q.push(p->right);

}

return root;

}

vector<int> lvl(Node\* root,int x) {

vector<int>v;

if (root == NULL) {

return v;

}

int n;

int c=-1;

queue<Node\*> q;

q.push(root);

while (!q.empty()) {

c++;

n=q.size();

for(int i=0;i<n;i++){

Node \*f=q.front();

q.pop();

if(c==x){

v.push\_back(f->val);

}

if(f->left)q.push(f->left);

if(f->right)q.push(f->right);

}

}

if(c<x){

vector<int>v1;

v1.push\_back(-1);

return v1 ;

}

return v;

}

int main() {

Node\* root = input\_tree();

vector<int>v;

int x;

cin>>x;

v=lvl(root,x);

for(auto i:v){

if(i==-1){

cout<<"Invalid"<<endl;

return 0;

}

cout<<i<<" ";

}

cout<<endl;

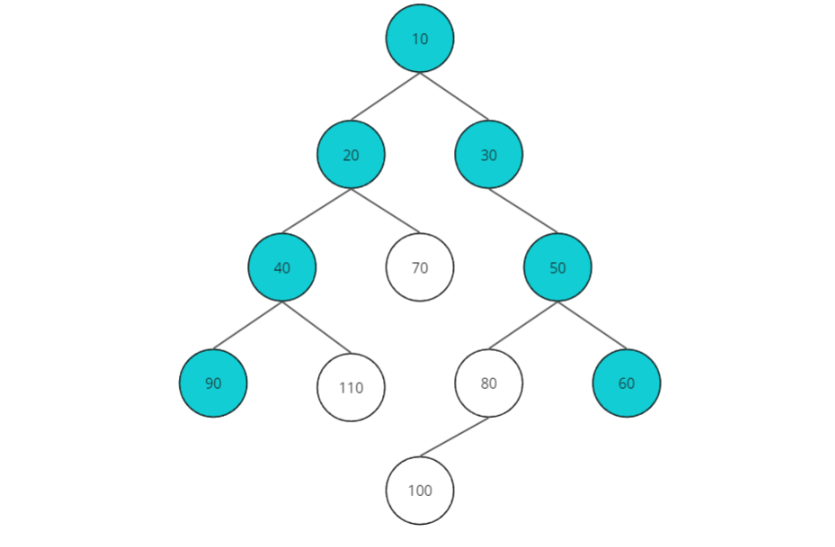
return 0;

}

96. <https://www.hackerrank.com/contests/assignment-04-a-basic-data-structure-a-batch-06/challenges/print-outer-tree>

You will be given a binary tree as input in level order. You need to print the outer side of the binary tree. See the sample input output for more clarifications. You need to print from the left most leaf node to right most leaf node.

**For example**:



The output for the above tree will be: **90 40 20 10 30 50 60**

**Input Format**

* Input will contain the binary tree in level order.
* means there is no node available.

**Constraints**

**Maximum number of nodes** **Node's value**

**Output Format**

* Output the left most leaf node to right most leaf node.

**Sample Input 0**

10

20 30

40 70 -1 50

90 110 -1 -1 80 60

-1 -1 -1 -1 100 -1 -1 -1

-1 -1

**Sample Output 0**

90 40 20 10 30 50 60

**Explanation 0**

This test case was explained in the question.

**Sample Input 1**

10

20 30

-1 40 70 50

60 90 -1 -1 80 -1

-1 -1 -1 -1 100 110

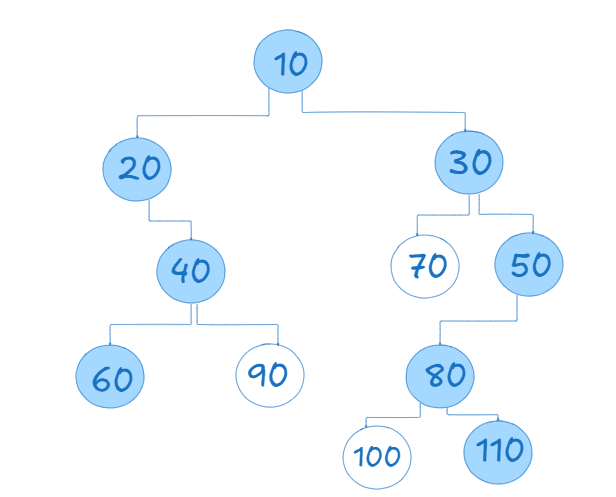
-1 -1 -1 -1

**Sample Output 1**

60 40 20 10 30 50 80 110

**Explanation 1**

Outer part of the binary tree -



**Sample Input 2**

10

-1 20

-1 50

80 -1

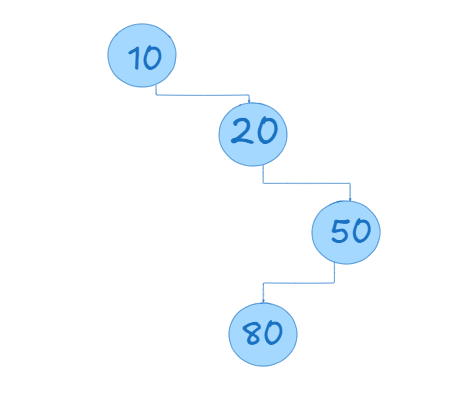
-1 -1

**Sample Output 2**

10 20 50 80

**Explanation 2**

Outer part of the binary tree -



**Sample Input 3**

10

20 -1

-1 50

80 -1

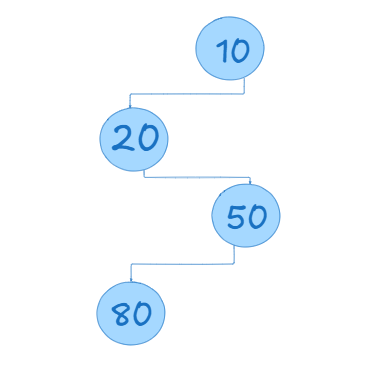
-1 -1

**Sample Output 3**

80 50 20 10

**Explanation 3**

Outer part of the binary tree -



#include <bits/stdc++.h>

using namespace std;

class TreeNode {

public:

int value;

TreeNode\* leftChild;

TreeNode\* rightChild;

TreeNode(int value) {

this->value = value;

this->leftChild = NULL;

this->rightChild = NULL;

}

};

TreeNode\* buildTree() {

int rootValue;

cin >> rootValue;

TreeNode\* root = (rootValue == -1) ? NULL : new TreeNode(rootValue);

queue<TreeNode\*> nodeQueue;

if (root)

nodeQueue.push(root);

while (!nodeQueue.empty()) {

TreeNode\* currentNode = nodeQueue.front();

nodeQueue.pop();

int leftValue, rightValue;

cin >> leftValue >> rightValue;

TreeNode\* leftChild = (leftValue == -1) ? NULL : new TreeNode(leftValue);

TreeNode\* rightChild = (rightValue == -1) ? NULL : new TreeNode(rightValue);

currentNode->leftChild = leftChild;

currentNode->rightChild = rightChild;

if (leftChild) nodeQueue.push(leftChild);

if (rightChild) nodeQueue.push(rightChild);

}

return root;

}

void traverseLeftBoundary(TreeNode\* node) {

if (node == NULL)

return;

if (node->leftChild) {

traverseLeftBoundary(node->leftChild);

cout << node->value << " ";

} else if (node->leftChild == NULL) {

traverseLeftBoundary(node->rightChild);

cout << node->value << " ";

}

}

void traverseRightBoundary(TreeNode\* node) {

if (node == NULL)

return;

if (node->rightChild) {

cout << node->value << " ";

traverseRightBoundary(node->rightChild);

} else if (node->rightChild == NULL) {

cout << node->value << " ";

traverseRightBoundary(node->leftChild);

}

}

int main() {

TreeNode\* root = buildTree();

if (root->leftChild == NULL) {

traverseRightBoundary(root);

} else if (root->rightChild == NULL) {

traverseLeftBoundary(root);

} else {

traverseLeftBoundary(root);

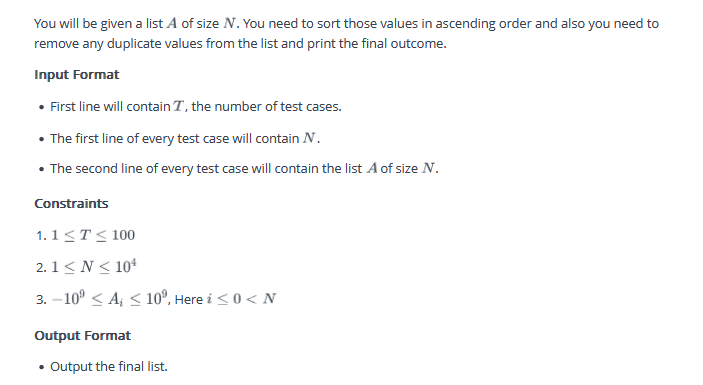
traverseRightBoundary(root->rightChild);

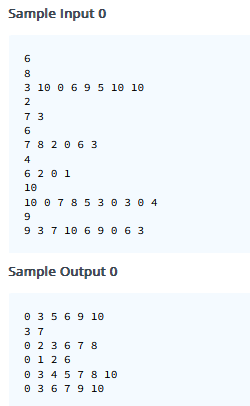
}

return 0;

}

97. <https://www.hackerrank.com/contests/final-exam-a-basic-data-structure-a-batch-06/challenges/sorted-3-1>





#include <bits/stdc++.h>

using namespace std;

int main() {

int n;

cin >> n;

for (int t = 0; t < n; t++) {

int m;

cin >> m;

set<int> unique\_nums;

for (int i = 0; i < m; i++) {

int val;

cin >> val;

unique\_nums.insert(val);

}

for (int val : unique\_nums) {

cout << val << " ";

}

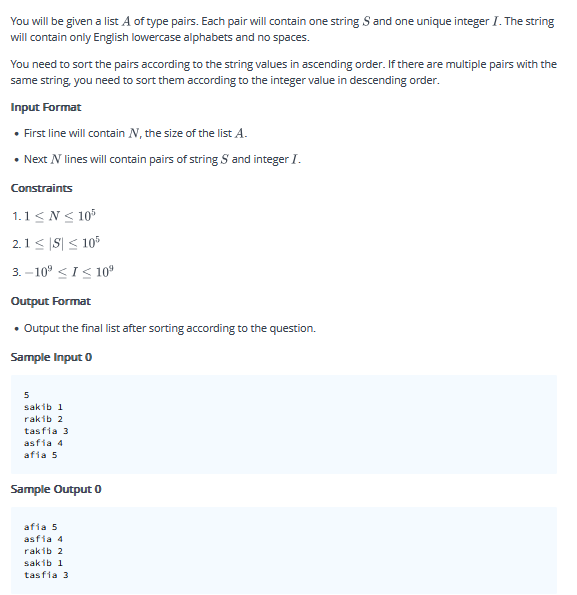
cout << endl;

}

return 0;

}

98. <https://www.hackerrank.com/contests/final-exam-a-basic-data-structure-a-batch-06/challenges/pairs-17>





#include <bits/stdc++.h>

using namespace std;

int main() {

int n;

cin >> n;

multimap<string, int> data;

for (int i = 0; i < n; i++) {

string s;

int num;

cin >> s >> num;

data.insert({s, num});

}

vector<pair<string, int>> sd(data.begin(), data.end());

sort(sd.begin(), sd.end(), [](const pair<string, int>& a, const pair<string, int>& b) {

if (a.first == b.first) {

return a.second > b.second;

}

return a.first < b.first;

});

for (const auto& p : sd) {

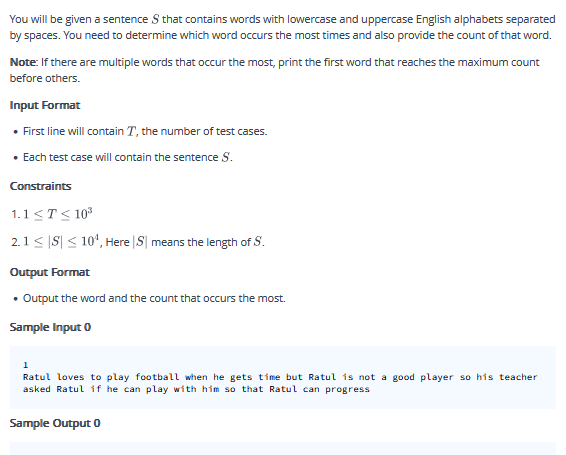
cout << p.first << " " << p.second << endl;

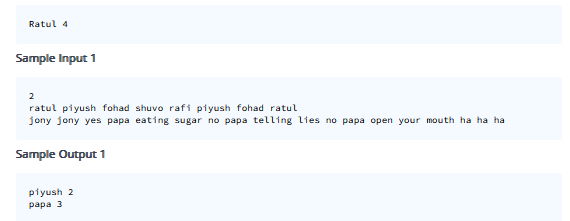
}

return 0;

}

99. <https://www.hackerrank.com/contests/final-exam-a-basic-data-structure-a-batch-06/challenges/count-me-2>





#include <bits/stdc++.h>

using namespace std;

int main() {

int t;

cin >> t;

cin.ignore();

while (t--) {

string str;

getline(cin, str);

stringstream ss(str);

string word;

map<string, int> mp;

int maxCount = 0;

string name;

while (ss >> word) {

mp[word]++;

if (mp[word] > maxCount) {

maxCount = mp[word];

name = word;

}

}

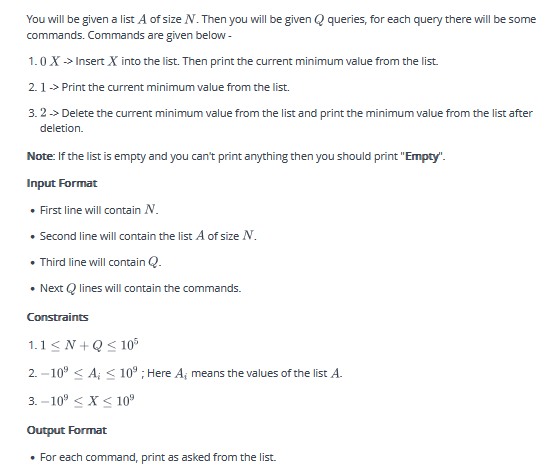
cout << name << " " << maxCount << endl;

}

return 0;

}

100. <https://www.hackerrank.com/contests/final-exam-a-basic-data-structure-a-batch-06/challenges/give-max-min>



**Sample Input 0**

4

10 -10 -5 -20

10

1

2

2

2

2

0 10

1

2

0 20

1

**Sample Output 0**

-20

-10

-5

10

Empty

10

10

Empty

20

20

**Sample Input 1**

6

45 -30 83 -99 19 75

9

1

2

2

0 32

0 6

2

2

0 -86

1

**Sample Output 1**

-99

-30

19

19

6

19

32

-86

-86

#include <bits/stdc++.h>

using namespace std;

int main() {

priority\_queue<int, vector<int>, greater<int>> p;

int n, q;

cin >> n;

for (int i = 0; i < n; i++) {

int val;

cin >> val;

p.push(val);

}

cin >> q;

while (q--) {

int x;

cin >> x;

if (x == 0) {

int y;

cin >> y;

p.push(y);

cout << p.top() << endl;

}

else if (x == 1) {

if (p.empty()) {

cout << "Empty" << endl;

} else {

cout << p.top() << endl;

}

}

else if (x == 2) {

if (p.empty()) {

cout << "Empty" << endl;

} else {

p.pop();

if (p.empty()) {

cout << "Empty" << endl;

} else {

cout << p.top() << endl;

}

}

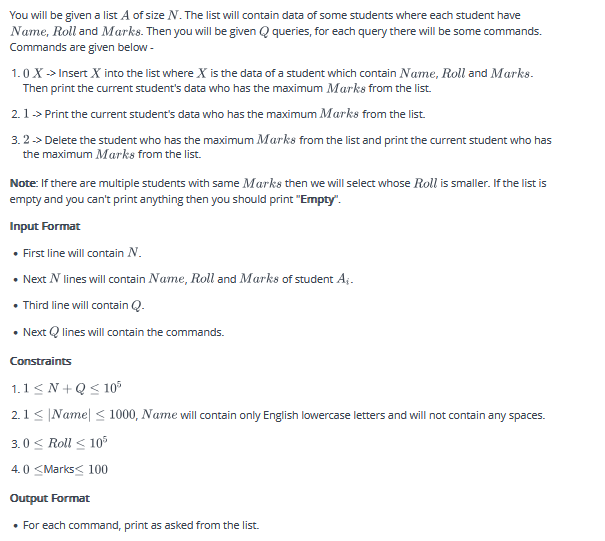
}

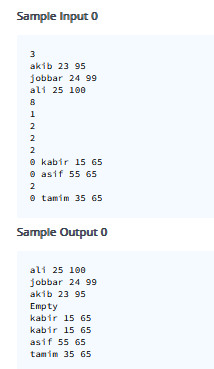
}

return 0;

}

101. <https://www.hackerrank.com/contests/final-exam-a-basic-data-structure-a-batch-06/challenges/get-current-max>





#include <bits/stdc++.h>

using namespace std;

class Student {

public:

string name;

int roll, marks;

Student(string name, int roll, int marks) : name(name), roll(roll), marks(marks) {}

};

class cmp {

public:

bool operator()(Student l, Student r) {

return l.marks < r.marks || (l.marks == r.marks && l.roll > r.roll);

}

};

int main() {

priority\_queue<Student, vector<Student>, cmp> pq;

int n, q;

cin >> n;

while (n--) {

string name;

int roll, marks;

cin >> name >> roll >> marks;

pq.push(Student(name, roll, marks));

}

cin >> q;

while (q--) {

int x;

cin >> x;

if (x == 0) {

string name;

int roll, marks;

cin >> name >> roll >> marks;

pq.push(Student(name, roll, marks));

cout << pq.top().name << " " << pq.top().roll << " " << pq.top().marks << endl;

} else if (x == 1) {

if (pq.empty()) cout << "Empty" << endl;

else cout << pq.top().name << " " << pq.top().roll << " " << pq.top().marks << endl;

} else if (x == 2) {

if (pq.empty()) cout << "Empty" << endl;

else {

pq.pop();

if (pq.empty()) cout << "Empty" << endl;

else cout << pq.top().name << " " << pq.top().roll << " " << pq.top().marks << endl;

}

}

}

return 0;

}