<div align="center">

Experiment -8

</div>

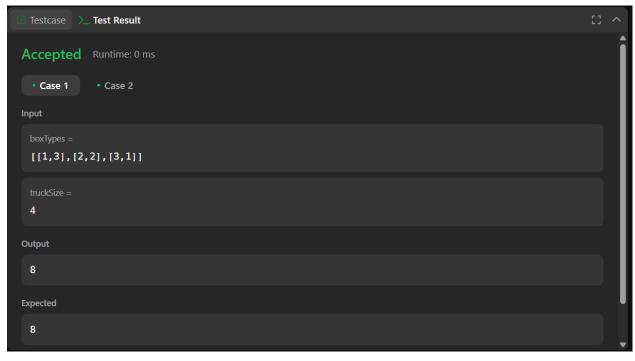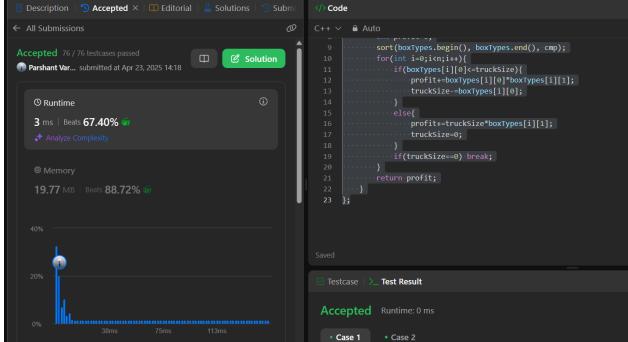|	
Student Name:Parshant Vardhan	**UID: 22BCS12914**
Branch: BE-CSE	**Section/Group: IOT-601(A)**
Semester: 5th	**Date of Performance: 23/04/25**
Subject Name: Advanced Programming Lab	**Subject Code: 22CSP-351**

Ques1. Maximum Units on a Truck

You are assigned to put some amount of boxes onto one truck. You are given a 2D array boxTypes, where boxTypes[i] = [numberOfBoxesi, numberOfUnitsPerBoxi]:

numberOfBoxesi is the number of boxes of type i.

numberOfUnitsPerBoxi is the number of units in each box of the type i.

```cpp
class Solution
{ public:
    unordered_map<int, int> inorderMap;
int preorderIndex = 0;

    TreeNode* Helper(vector<int>& preorder, vector<int>& inorder, int left, int
right) {          if (left > right) return NULL;

        int rootValue = preorder[preorderIndex++];
        TreeNode* root = new TreeNode(rootValue);

        int inorderIndex = inorderMap[rootValue];
         root->left = Helper(preorder, inorder, left, inorderIndex - 1);
root->right = Helper(preorder, inorder, inorderIndex + 1, right);

        return root;
    }

    TreeNode* buildTree(vector<int>& preorder, vector<int>& inorder) {
for (int i = 0; i < inorder.size(); i++) {
inorderMap[inorder[i]] = i;
        }          return Helper(preorder, inorder, 0,
inorder.size() - 1);      }
};
```

☑ Testcase   >_ **Test Result**   ⛶ ⌃

Accepted   Runtime: 0 ms

• **Case 1**   • Case 2

Input

boxTypes =
`[[1,3],[2,2],[3,1]]`

truckSize =
`4`

Output

`8`

Expected

`8`

📄 Description | ⟳ **Accepted** ✕ | 📖 Editorial | ⚗ Solutions | ⟳ Subm

← All Submissions   🔗

Accepted 76 / 76 testcases passed   📖 | ✏ **Solution**

👤 Parshant Var... submitted at Apr 23, 2025 14:18

🕐 Runtime   ⓘ

3 ms | Beats **67.40%** 👏

✦ Analyze Complexity

⚙ Memory

19.77 MB | Beats **88.72%** 👏

40%

20%

0%
      38ms   75ms   113ms

```cpp
 9          sort(boxTypes.begin(), boxTypes.end(), cmp);
10          for(int i=0;i<n;i++){
11              if(boxTypes[i][0]<=truckSize){
12                  profit+=boxTypes[i][0]*boxTypes[i][1];
13                  truckSize-=boxTypes[i][0];
14              }
15              else{
16                  profit+=truckSize*boxTypes[i][1];
17                  truckSize=0;
18              }
19              if(truckSize==0) break;
20          }
21          return profit;
22      }
23  };
```

Saved

☑ Testcase   >_ **Test Result**

Accepted   Runtime: 0 ms

• **Case 1**   • Case 2

Ques2. Lowest Common Ancestor of a Binary Tree

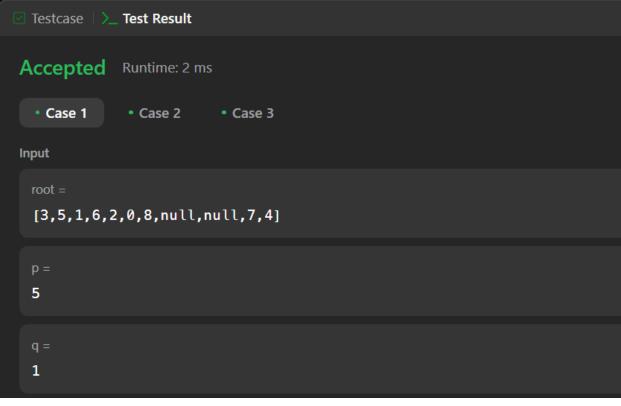Given a binary tree, find the lowest common ancestor (LCA) of two given nodes in the tree.

According to the definition of LCA on Wikipedia: "The lowest common ancestor is defined between two nodes   and   as the lowest node in   that has both   and   as descendants (where we allow **a node to be a descendant of itself**)."
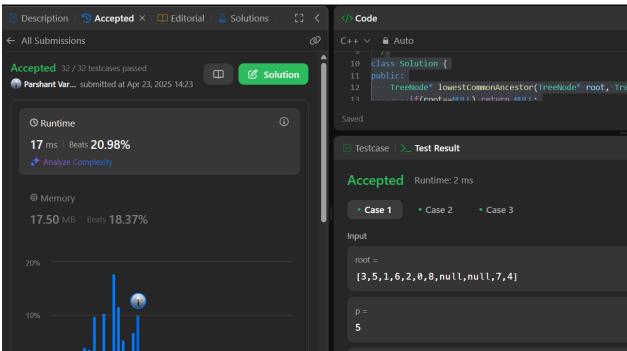
```cpp
class Solution {
public:
    TreeNode* lowestCommonAncestor(TreeNode* root, TreeNode* p, TreeNode* q) {
        if(root==NULL) return NULL;
        if(root->val==p->val)return p;
        if(root->val==q->val)return q;

        TreeNode* left=lowestCommonAncestor(root->left,p,q);
        TreeNode* right=lowestCommonAncestor(root->right,p,q);

        if(left==NULL && right==NULL) return NULL;
        else if(left==NULL && right!=NULL) return right;
        else if(left!=NULL && right==NULL) return left;
        else    return root;

    }
};
```

☑ Testcase | >_ **Test Result**

Accepted   Runtime: 2 ms

• Case 1   • Case 2   • Case 3

Input

root =
[3,5,1,6,2,0,8,null,null,7,4]

p =
5

q =
1

📄 Description | 🕘 Accepted ✕ | 📖 Editorial | ⚗ Solutions   ⛶ ‹

← All Submissions   🔗

Accepted 32 / 32 testcases passed
🌐 Parshant Var... submitted at Apr 23, 2025 14:23   📖 ✏ **Solution**

🕐 Runtime   ⓘ

17 ms | Beats **20.98%**
✦ Analyze Complexity

⚙ Memory

17.50 MB | Beats **18.37%**

20%

10%

</> **Code**

C++ ∨   🔒 Auto

```
 9     7
10    class Solution {
11    public:
12        TreeNode* lowestCommonAncestor(TreeNode* root, Tre
13        if(root==NULL) return NULL;
```

Saved

☑ Testcase | >_ **Test Result**

Accepted Runtime: 2 ms

• Case 1   • Case 2   • Case 3

Input

root =
[3,5,1,6,2,0,8,null,null,7,4]

p =
5

Ques 3: Maximum Score From Removing Substrings

You are given a string s and two integers x and y. You can perform two types of operations any number of times.

- Remove substring "ab" and gain x points.
- For example, when removing "ab" from "cabxbae" it becomes "cxbae".
- Remove substring "ba" and gain y points.
- For example, when removing "ba" from "cabxbae" it becomes "cabxe".

Return *the maximum points you can gain after applying the above operations on* s.

CODE:

```cpp
class Solution {
public:
    int removeSubstring(string& s, string sub, int score) {
        stack<char> st;
        int points = 0;

        for (char c : s) {
            if (!st.empty() && st.top() == sub[0] && c == sub[1]) {
                st.pop();
                points += score;
            } else {
                st.push(c);
            }
        }

        s = "";
        while (!st.empty()) {
            s += st.top();
            st.pop();
        }
        reverse(s.begin(), s.end());

        return points;
    }
};

int maximumGain(string s, int x, int y) {
    int totalPoints = 0;
    if (x > y) {
        totalPoints += removeSubstring(s, "ab", x);
        totalPoints += removeSubstring(s, "ba", y);
    } else {
        totalPoints += removeSubstring(s, "ba", y);
```

```
        totalPoints += removeSubstring(s, "ab", x);
    }

    return totalPoints;
}

};
```

☑ Testcase >_ **Test Result**

Accepted   Runtime: 0 ms

• **Case 1**   • Case 2

Input

s =
"cdbcbbaaabab"

x =
4

y =
5