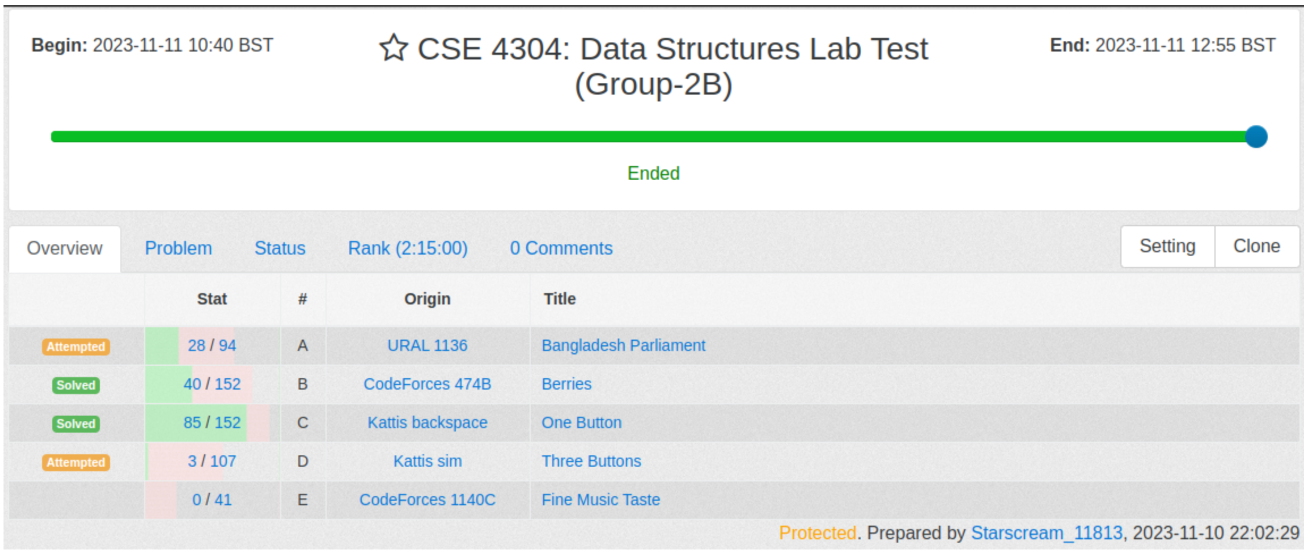
Improvement assignment for Lab Mid Exam

Name: Md. Abdullah Al Jubaer Gem
Student ID: 210041226
Lab Group: 2B
Screenshot showing your performance from the contest portal:



Begin: 2023-11-11 10:40 BST

☆ CSE 4304: Data Structures Lab Test (Group-2B)

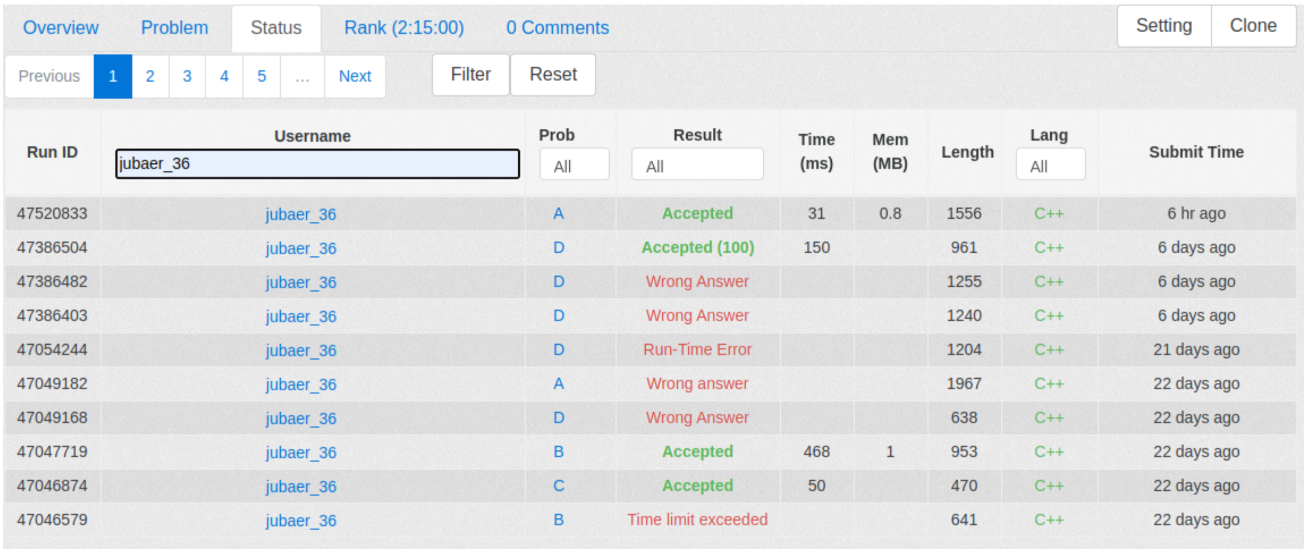
End: 2023-11-11 12:55 BST

Ended

Overview | Problem | Status | Rank (2:15:00) | 0 Comments | Setting | Clone

	Stat	#	Origin	Title
Attempted	28 / 94	A	URAL 1136	Bangladesh Parliament
Solved	40 / 152	B	CodeForces 474B	Berries
Solved	85 / 152	C	Kattis backspace	One Button
Attempted	3 / 107	D	Kattis sim	Three Buttons
	0 / 41	E	CodeForces 1140C	Fine Music Taste

Protected. Prepared by Starscream_11813, 2023-11-10 22:02:29



Overview | Problem | Status | Rank (2:15:00) | 0 Comments | Setting | Clone

Previous 1 2 3 4 5 ... Next | Filter | Reset

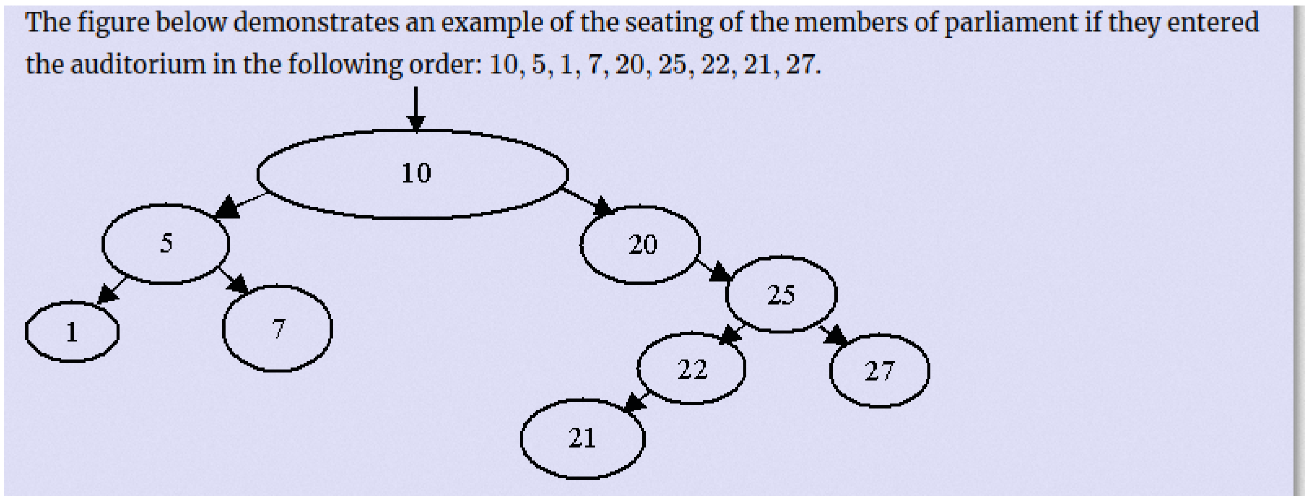
Run ID	Username (jubaer_36)	Prob (All)	Result (All)	Time (ms)	Mem (MB)	Length	Lang (All)	Submit Time
47520833	jubaer_36	A	Accepted	31	0.8	1556	C++	6 hr ago
47386504	jubaer_36	D	Accepted (100)	150		961	C++	6 days ago
47386482	jubaer_36	D	Wrong Answer			1255	C++	6 days ago
47386403	jubaer_36	D	Wrong Answer			1240	C++	6 days ago
47054244	jubaer_36	D	Run-Time Error			1204	C++	21 days ago
47049182	jubaer_36	A	Wrong answer			1967	C++	22 days ago
47049168	jubaer_36	D	Wrong Answer			638	C++	22 days ago
47047719	jubaer_36	B	Accepted	468	1	953	C++	22 days ago
47046874	jubaer_36	C	Accepted	50		470	C++	22 days ago
47046579	jubaer_36	B	Time limit exceeded			641	C++	22 days ago

Task A:
Solution approach:

The problem states that there is a parliament where members get seated according to their random positive number in a tree-like structure. The first of them took the chairman's seat and the rest of them gets seated following the rule :

If MP's positive number greater than chairman's then he goes to right
Else MP's seat goes to chairman's left

This process continued as long as there were MPs incoming.


The figure below demonstrates an example of the seating of the members of parliament if they entered the auditorium in the following order: 10, 5, 1, 7, 20, 25, 22, 21, 27.

Now if the number of sessions were odd then MPs spoke following the order left,right , root
Else they spoke in the order right , left , root.

Now the speech order of MPs for the parliament was given in the order left , right , root and we are to find out the speech order if the number of sessions was even.

Approach:
To solve this problem, the first thing I noticed was that it would form a binary search tree where the right child is greater than the parent node and the left child is less than the parent node.
And the result of post order traversal of the tree is given as the input. Now we are to find out the post order traversal form of the tree if the tree is traversed like this right , left , root.
So first to construct the tree from a given post order traversal form, what I found is that the present input form is directly opposite of the original input given form . So I took all the input in a stack so that while inserting in a tree I can take inputs in the reverse order from the stack.

Now to implement the tree I took two classes one of which defines the node for the tree and the other one defines the whole binary tree. I implemented two functions: "insert" , "even order traversal".
To insert I traverse the whole tree and find the perfect position for the node by implementing the condition that if the node to be inserted is greater or less than the parent node then I insert the node at the position then fix the node along with its parent node.
Thus I built the desired BST (Binary Search Tree).
Now for the even order traversal function I basically implemented the post order traversal function with the slight modification that the right subtree should come before the left subtree and lastly the parent node.

<u>Main body of code</u> :

```
#include <bits/stdc++.h>
using namespace std;

class Node
{
public:
   int data;
   Node *left;
   Node *right;
   Node *parent;
   Node(int data = 0)
   {
       left = nullptr;
       right = nullptr;
       parent = nullptr;
       this->data = data;
   }
};

class Tree
{
public:
   Tree()
   {
       root = nullptr;
   }
   Node *root;
   void insert(int data)
   {
       Node *newNode = new Node(data);
       Node *temp = root;
       Node *target = nullptr;
       if (root == nullptr)
       {
           root = newNode;
           return;
       }

       while (temp != nullptr)
       {
           target = temp;
           if (temp->data > data)
               temp = temp->left;
           else
```

```
                temp = temp->right;
        }
        newNode->parent = target;

        if (data < target->data)
            target->left = newNode;
        else
            target->right = newNode;
    }
    void even_order_traversal(Node *temp)
    {
        if (temp == nullptr)
            return;
        even_order_traversal(temp->right);

        even_order_traversal(temp->left);
        cout << temp->data << endl;
    }
};

int main()
{
    int n, val;
    cin >> n;
    Tree t;
    stack<int> stk;
    for (int i = 0; i < n; i++)
    {
        cin >> val;

        stk.push(val);
    }
    for (int i = 0; i < n; i++)
    {
        val = stk.top();
        t.insert(val);
        stk.pop();
    }
    t.even_order_traversal(t.root);
}
```

Task D:
Solution approach:
This problem states that a person is typing and whenever he types the following characters some specific function happens
'<' - Backspace key is pressed and last character is deleted
'[' - Home key is pressed and cursor goes to the front of the string
']' - End key is pressed and cursor goes to the end of the string
I am to find out the string that is ultimately shown on display.
Approach:
To solve this problem there can be the simplest approach of taking a vector of character or a string where each operation will be done very easily using iterators .
But the problem with this solution is that in each insertion all the characters are shifted on to the right one by one which is very expensive.
So one better approach can be to take a deque to store the characters where if a character is inserted at the beginning of the string there is no need to shift every character . And if a value is inserted in the middle then only shifting is required which is comparatively less expensive than the previous approach.
So firstly I took a deque then took the characters as input one by one then inserted them in the string according to the iterator as the iterator was declared pointing to the very beginning of the deque and whenever a value is inserted the iterator gets updated to the inserted character position.
So if '[' is pressed then iterator is set to the beginning of the deque
If ']' is pressed then iterator is set to the end of the deque
If '<' is pressed then the value at the iterator is deleted
For any other character the value is inserted at the iterator position and the iterator value is updated for the next to be inserted character . That's why during deletion the iterator value is first checked if it is at the very beginning of the deque and decremented to the position where the previous character was inserted.
Then ultimately the deque is popped from the front one by one and displayed on the terminal which is the required string.

Main body of code :

```
#include <iostream>
#include <deque>
#include <string>

using namespace std;

int main()
{
   int t;
   cin >> t;
   cin.ignore();
   while (t--)
   {
       string ln;
       deque<char> d;
```

```
        getline(cin, ln);
        auto it = d.begin();

        for (int i = 0; i < ln.size(); i++)
        {
            if (ln[i] == '[')
            {
                it = d.begin();
            }
            else if (ln[i] == ']')
            {
                it = d.end();
            }
            else if (ln[i] == '<')
            {
                if (it != d.begin())
                {
                    it--;
                    it = d.erase(it);
                }
            }
            else
            {

                it = d.insert(it, ln[i]);
                it++;
            }
        }

        while (!d.empty())
        {
            cout << d.front();
            d.pop_front();
        }
        cout << endl;
    }

    return 0;
}
```