Date : 24.03.2025
Lakshana Baskaran
24011103026

LAB - 9
Tree Data Structure - BST

QUESTION 1:

A. Utilize C++ STL to solve the following (K5),

Nene invented a new game based on an increasing sequence of integers $a_1, a_2, \ldots, a_k$.

In this game, initially n players are lined up in a row. In each of the rounds of this game, the following happens:

- Nene finds the a_1-th, a_2-th, ..., a_k-th players in a row. They are kicked out of the game simultaneously. If the i-th player in a row should be kicked out, but there are fewer than i players in a row, they are skipped.

Once no one is kicked out of the game in some round, all the players that are still in the game are declared as winners.

For example, consider the game with $a = [3, 5]$ and $n = 5$ players. Let the players be named player `A`, player `B`, ..., player `E` in the order they are lined up initially. Then,

- Before the first round, players are lined up as `ABCDE`. Nene finds the **3**-rd and the **5**-th players in a row. These are players `C` and `E`. They are kicked out in the first round.
- Now players are lined up as `ABD`. Nene finds the **3**-rd and the **5**-th players in a row. The **3**-rd player is player `D` and there is no **5**-th player in a row. Thus, only player `D` is kicked out in the second round.
- In the third round, no one is kicked out of the game, so the game ends after this round.
- Players `A` and `B` are declared as the winners.

Nene has not yet decided how many people would join the game initially. Nene gave you q integers $n_1, n_2, \ldots, n_q$ and you should answer the following question for each $1 \leq i \leq q$ **independently**:

- How many people would be declared as winners if there are n_i players in the game initially?

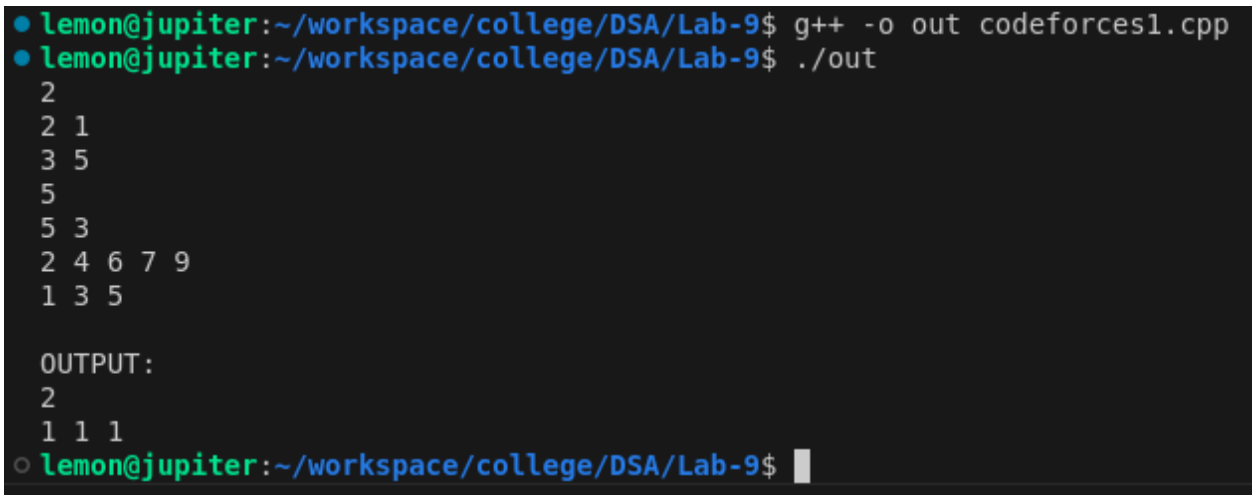
SOURCE CODE:

```
//how many would be declared as winners?
#include <iostream>
#include <vector>
#include <algorithm>
#include <set>
using namespace std;

int calculateWinners(int n, const vector<int>& a) {
    vector<int> players(n);
    for (int i = 0; i < n; ++i) players[i] = i; // initial lineup

    while (true) {
        vector<int> toKick;
```

```
        for (int pos : a) {
            if (pos <= players.size()) {
                toKick.push_back(pos - 1); // 0-based index
            }
        }

        if (toKick.empty()) break;

        set<int> kickSet(toKick.begin(), toKick.end());
        vector<int> newPlayers;
        for (int i = 0; i < players.size(); ++i) {
            if (kickSet.find(i) == kickSet.end()) {
                newPlayers.push_back(players[i]);
            }
        }

        if (newPlayers.size() == players.size()) break;
        players = newPlayers;
    }

    return players.size();
}

int main() {
    int t;
    cin >> t;

    vector<vector<int>> results; // store results for all test cases

    while (t--) {
        int k, q;
        cin >> k >> q;

        vector<int> a(k);
        for (int i = 0; i < k; ++i) cin >> a[i];

        vector<int> queries(q);
        for (int i = 0; i < q; ++i) cin >> queries[i];

        vector<int> currentResult;
        for (int n : queries) {
            currentResult.push_back(calculateWinners(n, a));
        }
```

```
        results.push_back(currentResult);
    }

    cout << "\nOUTPUT:" << endl;
    // Print all results at the end
    for (const auto& testCase : results) {
        for (int x : testCase) {
            cout << x << " ";
        }
        cout << endl;
    }

    return 0;
}
```

OUTPUT:



Date : 24.03.2025
Lakshana Baskaran
24011103026

QUESTION 2:

B. Utilize C++ STL to solve the following (K5),

There are n participants in a competition, participant i having a strength of s_i.

Every participant wonders how much of an advantage they have over the other best participant. In other words, each participant i wants to know the difference between s_i and s_j, where j is the strongest participant in the competition, not counting i (a difference can be negative).

So, they ask you for your help! For each i ($1 \le i \le n$) output the difference between s_i and the maximum strength of any participant other than participant i.

Input
The input consists of multiple test cases. The first line contains an integer t ($1 \le t \le 1000$) — the number of test cases. The descriptions of the test cases follow.

The first line of each test case contains an integer n ($2 \le n \le 2 \cdot 10^5$) — the length of the array.

The following line contains n space-separated positive integers $s_1, s_2, ..., s_n$ ($1 \le s_i \le 10^9$) — the strengths of the participants.

It is guaranteed that the sum of n over all test cases does not exceed $2 \cdot 10^5$.

Output
For each test case, output n space-separated integers. For each i ($1 \le i \le n$) output the difference between s_i and the maximum strength of any other participant.

Input
The input consists of multiple test cases. The first line contains an integer t ($1 \le t \le 1000$) — the number of test cases. The descriptions of the test cases follow.

The first line of each test case contains an integer n ($2 \le n \le 2 \cdot 10^5$) — the length of the array.

The following line contains n space-separated positive integers $s_1, s_2, ..., s_n$ ($1 \le s_i \le 10^9$) — the strengths of the participants.

It is guaranteed that the sum of n over all test cases does not exceed $2 \cdot 10^5$.

Output
For each test case, output n space-separated integers. For each i ($1 \le i \le n$) output the difference between s_i and the maximum strength of any other participant.

SOURCE CODE:

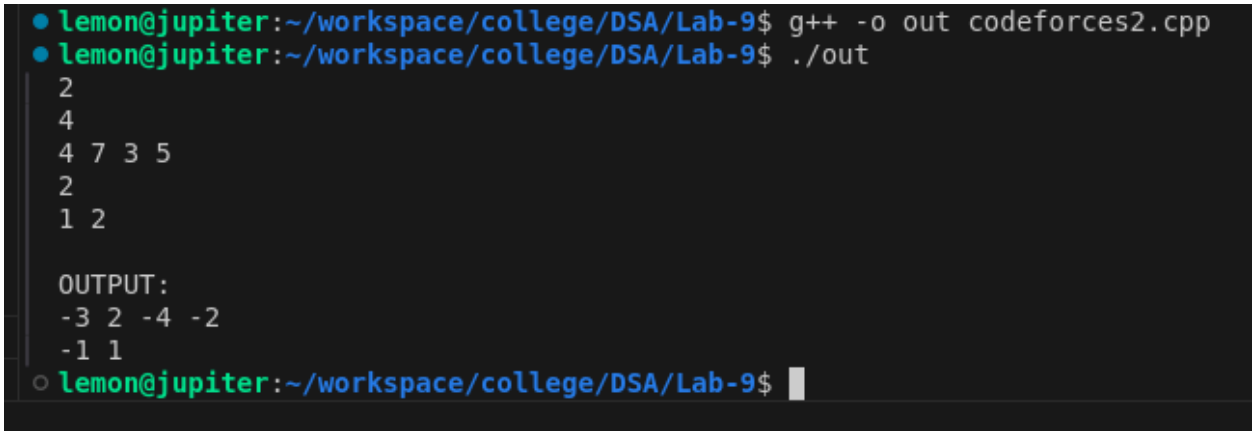
```
//maximum strength of any participant that is not i
#include <iostream>
#include <vector>
#include <algorithm>
#include <climits>
using namespace std;

int main() {
    int t;
    cin >> t;
```

Date : 24.03.2025
Lakshana Baskaran
24011103026

```
    vector<string> results; // Store outputs for all test cases

    while (t--) {
        int n;
        cin >> n;
        vector<int> strengths(n);
        for (int i = 0; i < n; ++i) {
            cin >> strengths[i];
        }

        int max_strength = INT_MIN, second_max_strength = INT_MIN;
        for (int i = 0; i < n; ++i) {
            if (strengths[i] > max_strength) {
                second_max_strength = max_strength;
                max_strength = strengths[i];
            } else if (strengths[i] > second_max_strength &&
strengths[i] != max_strength) {
                second_max_strength = strengths[i];
            }
        }

        int count_max = count(strengths.begin(), strengths.end(),
max_strength);
        string output_line;

        for (int i = 0; i < n; ++i) {
            int diff;
            if (strengths[i] == max_strength) {
                diff = (count_max > 1) ? 0 : strengths[i] -
second_max_strength;
            } else {
                diff = strengths[i] - max_strength;
            }
            output_line += to_string(diff) + " ";
        }

        results.push_back(output_line);
    }

    cout << "\nOUTPUT:" << endl;
    // Output all results at the end
    for (const string& res : results) {
        cout << res << endl;
```

```
    }

    return 0;
}
```

OUTPUT:



Date : 24.03.2025
Lakshana Baskaran
24011103026

QUESTION 3:

C. Write a separate C++ menu-driven program to implement Tree ADT using a binary search tree. Maintain proper boundary conditions and follow good coding practices. The Tree ADT has the following operations,

1. Insert
2. Preorder
3. Inorder
4. Postorder
5. Search
6. Exit

SOURCE CODE:

```
//implementation of binary search tree
#include <stdio.h>

class BinaryTree{
    private:
        struct TreeNode {
            int data;
            struct TreeNode* left;
            struct TreeNode* right;
        };
        struct TreeNode* root;

    public:
        TreeNode* CreateNode(int);

        TreeNode* insert(TreeNode*, int);
        void Insert(int);

        void inorder_traversal(TreeNode*);
        void InorderTraversal();

        void preorder_traversal(TreeNode*);
        void PreorderTraversal();

        void postorder_traversal(TreeNode*);
```

```
        void PostorderTraversal();

        TreeNode* Search(int);
        TreeNode* search(TreeNode*, int);


        BinaryTree(){
            root = NULL;
        }
};

BinaryTree::TreeNode* BinaryTree::CreateNode(int key) {
    TreeNode* newnode = new TreeNode;
    newnode -> data = key;
    newnode -> left = NULL;
    newnode -> right = NULL;

    return newnode;
}


void BinaryTree::Insert(int key) {
    root = insert(root, key);
}

BinaryTree::TreeNode* BinaryTree::insert(TreeNode* root, int key) {
    if (root == NULL) {
        root = CreateNode(key);
        return root;
    }

    if (key < root -> data) {
        root -> left = insert(root->left, key);
    }

    else if (key > root -> data) {
        root -> right = insert(root->right, key);
    }

    return root;
}
```

Date : 24.03.2025
Lakshana Baskaran
24011103026

```
BinaryTree::TreeNode* BinaryTree::Search(int key) {
    return search(root, key);
}

BinaryTree::TreeNode* BinaryTree::search(TreeNode* root, int key) {
    if (root == NULL || key == root->data) {
        return root;
    }

    if (key < root -> data) {
        return search(root->left, key);
    }

    return search(root->right, key);

}


//inorder traversal
void BinaryTree::InorderTraversal() {
    inorder_traversal(root);
}

void BinaryTree::inorder_traversal(TreeNode* root) {
    if (root == NULL) {
        return;
    }

    inorder_traversal(root->left);
    printf("%d ", root->data);
    inorder_traversal(root->right);
}

//preorder traversal
void BinaryTree::PreorderTraversal() {
    preorder_traversal(root);
}

void BinaryTree::preorder_traversal(TreeNode* root) {
    if (root == NULL) {
        return;
    }
```

```
    printf("%d ", root->data);
    preorder_traversal(root->left);
    preorder_traversal(root->right);
}

//postorder traversal
void BinaryTree::PostorderTraversal() {
    postorder_traversal(root);
}

void BinaryTree::postorder_traversal(TreeNode* root) {
    if (root == NULL) {
        return;
    }

    postorder_traversal(root->left);
    postorder_traversal(root->right);
    printf("%d ", root->data);
}


int main() {
    BinaryTree tree;
    int x, choice = 0;

    printf("MENU\n1 - Insertion\n2 - Indorder Traversal\n3 - Preorder
Traversal\n4 - Postorder Traversal\n5 - Search\n6 - Exit\n");
    printf("Zero is not allowed in the queue. If the UnderFlowError
occurs zero will be returned.\n");
    while (true) {
        printf("\nEnter your choice: ");
        scanf("%d", &choice);
        switch (choice) {
            case 1:
                printf("Enter node to be inserted: ");
                scanf("%d", &x);
                tree.Insert(x);
                break;

            case 2:
                printf("Inorder Traversal: ");
                tree.InorderTraversal();
                printf("\n");
```

```
                break;

            case 3:
                printf("Preorder Traversal: ");
                tree.PreorderTraversal();
                printf("\n");
                break;

            case 4:
                printf("Postorder Traversal: ");
                tree.PostorderTraversal();
                printf("\n");
                break;

            case 5:
                printf("Enter node to be searched: ");
                scanf("%d", &x);
                if (tree.Search(x) != NULL) {
                    printf("Node is present in tree\n");
                }
                else {
                    printf("Node is NOT present in tree\n");
                }
                break;

            case 6:
                printf("Exiting...\n");
                return 0;
                break;

            default:
                printf("\nInvalid choice. Enter again.\n");
                break;
        }

    }

}
```

Date : 24.03.2025
Lakshana Baskaran
24011103026

OUTPUT:

```
lemon@jupiter:~/workspace/college/DSA/Lab-9$ g++ -o out binary_search_tree.cpp
lemon@jupiter:~/workspace/college/DSA/Lab-9$ ./out
MENU
1 - Insertion
2 - Indorder Traversal
3 - Preorder Traversal
4 - Postorder Traversal
5 - Search
6 - Exit
Zero is not allowed in the queue. If the UnderFlowError occurs zero will be returned.

Enter your choice: 1
Enter node to be inserted: 10

Enter your choice: 1
Enter node to be inserted: 9

Enter your choice: 1
Enter node to be inserted: 11

Enter your choice: 2
Inorder Traversal: 9 10 11

Enter your choice: 3
Preorder Traversal: 10 9 11

Enter your choice: 4
Postorder Traversal: 9 11 10

Enter your choice: 5
Enter node to be searched: 10
Node is present in tree

Enter your choice: 5
Enter node to be searched: 30
Node is NOT present in tree

Enter your choice: 6
Exiting...
lemon@jupiter:~/workspace/college/DSA/Lab-9$
```

Date : 24.03.2025
Lakshana Baskaran
24011103026

QUESTION 4:

D. Add a "construct expression tree" method to the binary tree data structure from the previous lab code—import stack from the standard template library (STL) to construct the expression tree. Import the Tree ADT program into another program that gets a valid postfix expression, constructs, and prints the expression tree. It consists of the following operations.

1. Postfix Expression
2. Construct Expression Tree
3. Preorder
4. Inorder
5. Postorder
6. Exit

SOURCE CODE:

```
//conducting conversions using .h file that performs actions on
expression trees
#include "exptree.h"
int main() {
    ExpressionTree tree;
    int choice;

    while (true) {
        cout << "\n1. Postfix Expression\n2. Construct Expression
Tree\n3. Preorder\n4. Inorder\n5. Postorder\n6. Exit\nChoice: ";
        cin >> choice;

        switch (choice) {
            case 1:
                tree.readPostfix();
                break;
            case 2:
                tree.constructTree();
                break;
            case 3:
                tree.displayPreorder();
                break;
```

```
            case 4:
                tree.displayInorder();
                break;
            case 5:
                tree.displayPostorder();
                break;
            case 6:
                return 0;
            default:
                cout << "Invalid choice.\n";
        }
    }
}
```

Date : 24.03.2025
Lakshana Baskaran
24011103026

OUTPUT:

```
lemon@jupiter:~/workspace/college/DSA/Lab-9$ g++ -o out conversions.cpp
lemon@jupiter:~/workspace/college/DSA/Lab-9$ ./out

 1. Postfix Expression
 2. Construct Expression Tree
 3. Preorder
 4. Inorder
 5. Postorder
 6. Exit

 Choice: 1
 Enter postfix expression: 12+34-*

 Choice: 2
 Expression Tree Constructed.

 Choice: 3
 Preorder: *+12-34

 Choice: 4
 Inorder: ((1+2)*(3-4))

 Choice: 5
 Postorder: 12+34-*

 Choice: 6
lemon@jupiter:~/workspace/college/DSA/Lab-9$
```

—