Lab Assignment 6: CS2233

8th October, 2025

Kindly adhere to the following instructions.

- Please write a C/C++ program corresponding to each problem. Your code should be well commented, and variable names should be appropriately chosen. Also, prepare a `readme` text file where you can mention instructions to run the program/how to take input, etc.
- Create a folder and put all the code files and `readme` text file in it, give a name to the folder as "yourName_yourRollNo", zip the folder and submit it to the Google Classroom portal.
- Your code will also be checked against plagiarism (both from web and peers).
- Any form of plagiarism (web/ChatGPT/with peers) will be severely penalised and will result in an F grade.

Question:

Construct a Red-Black tree over the following keys (assuming you read the numbers from left to right). You can assume that there are no duplicates in the input.

$9, 8, 5, 4, 99, 78, 31, 34, 89, 90, 21, 23, 45, 77, 88, 112, 32.$

The Red-Black tree should be constructed by calling the `insert (root, key)` listed below.

Each `internal node` of the tree should use the following `struct` data type:

```
struct node
{
int data;
struct node *left;
struct node *right;
struct node *parent;
```

```
bool color;
int black_depth;
} ;
```

Note: Using parent pointers and black depth in the above structure is optional.

You can assume that you have stored the pointer to the `root` node.
Please, write the functions for:

1. `search (root, key)` – this function takes the pointer to the `root` node, and `key` as input, and returns the pointer to the node where `key` is present. If `key` is not present in the Red-Black tree, then the code should output an error message. Please run your function for searching nodes $32, 56, 90$.

2. `insert (root, key)` – this function takes the pointer to the `root` node, and key as input, and inserts the node at the appropriate position. Please run your function for inserting nodes $132, 156, 11, 7$. Print `inorder` and `preorder` traversal of the respective trees upon each insertion.

3. `delete (root, key)` – this function takes the pointer to the `root` node, and the key as input, and deletes the corresponding node. Please run your functions for deleting nodes $332, 51, 71, 67$. Your code should output an error message if the `key` is absent in the Red-Black tree.

Note: Avoid inserting duplicate elements in the tree.

Input Format:

The first integer will be n, the size of the array. This will be followed by the n elements of the array.

Output Format:

Print the inorder traversal, followed by the preorder traversal in the next line. Then display the following prompt:

```
Enter your choice:
1 - Search for an element
2 - Insert an element
3 - Delete an element
0 - Exit
```

The program should keep showing this menu until the user enters 0. For every action, take one integer input (the element to work with) and perform the corresponding operation.