REPORT

STUDENT NAME: OENG SUNHAK

GROUP: 2

ID: IDTB100103

TREE - IMPLEMENTATION

LAB 8

Generate 1 million citizen ID and store in a file csv.

Read data from file and store these citizen ID of person into a binary search tree. Create ADT operations below:

a-Insert data to BST

b-Display this BST using pre-order traversal (DLR)

c-Display this BST using in-order traversal (LDR)

d-Display this BST using post-order traversal (LRD)

e-Search data in a BST. Ask a user for citizen ID then the program tell whether this ID is exist in the BST.

f-Delete data from BST. Ask a user for citizen ID then delete from BST if it exists. If not exist, just display "Can not delete. ID not found!".

Source Code:

```cpp
#include <iostream>
#include <fstream>
#include <vector>
#include <algorithm> // For shuffle
#include <ctime>     // For random seed

using namespace std;

struct Node{
    int data;
    Node *left, *right;
};

class BST{
public:
    int size;
    Node *root;

    BST() {
        size = 0;
        root = NULL;
    }

    Node* insert(Node *root, int newData){
        if(root == NULL){
            root = new Node;
            root->data = newData;
            root->left = NULL;
            root->right = NULL;
            size++;
        }else if(newData > root->data){
            root->right = insert(root->right, newData);
        }else{
            root->left = insert(root->left, newData);
        }
        return root;
    }
```

```
38
39      void add(int newData){
40          root = insert(root, newData);
41      }
42
43      // Pre-order Traversal(DLR)
44      void preorder(Node *root){
45          if(root != NULL){
46              cout << root->data << " ";
47              preorder(root->left);
48              preorder(root->right);
49          }
50      }
51
52      void preorder_(){
53          preorder(root);
54          cout << endl;
55      }
56
57      // In-order Traversal(LDR)
58      void inorder(Node *root){
59          if(root != NULL){
60              inorder(root->left);
61              cout << root->data << " ";
62              inorder(root->right);
63          }
64      }
65
66      void inorder_(){
67          inorder(root);
68          cout << endl;
69      }
70
71      // Post-order Traversal(LRD)
72      void postorder(Node *root){
73          if(root != NULL){
74              postorder(root->left);
75              postorder(root->right);
76              cout << root->data << " ";
77          }
78      }
79
80      void postorder_(){
81          postorder(root);
82          cout << endl;
83      }
84
85      bool search(Node *root, int data){
86          if(root == NULL){
87              return false;
88          }else if (root->data == data){
89              return true;
90          }else if (data > root->data){
91              return search(root->right, data);
```

```
92          }else{
93              return search(root->left, data);
94          }
95      }
96
97      bool search_(int data){
98          return search(root, data);
99      }
100
101     Node* deleteNode(Node *root, int data){
102         if(root == NULL) return root;
103
104         if(data < root->data){
105             root->left = deleteNode(root->left, data);
106         }else if(data > root->data) {
107             root->right = deleteNode(root->right, data);
108         }else{
109             // Node with one or no child
110             if(root->left == NULL){
111                 Node *temp = root->right;
112                 delete root;
113                 size--;
114                 return temp;
115             }else if(root->right == NULL){
116                 Node *temp = root->left;
117                 delete root;
118                 size--;
119                 return temp;
120             }
121
122             // Node with two children: find inorder successor
123             Node *temp = root->right;
124             while(temp->left != NULL) temp = temp->left;
125
126             root->data = temp->data;
127             root->right = deleteNode(root->right, temp->data);
128         }
129         return root;
130     }
131
132     void delete_(int data){
133         root = deleteNode(root, data);
134     }
135 };
136
137 // Function to load and shuffle IDs
138 void loadRandomIDs(const string &filename, BST &tree){
139     ifstream file(filename);
140     if(!file){
141         cerr << "Error opening file!" << endl;
142         return;
143     }
144
145     vector<int> ids;
146     int id;
147     while(file >> id){ // Read integers line by line
148         ids.push_back(id);
149     }
150
151     if(ids.empty()){
152         cerr << "No valid IDs found in the file!" << endl;
153         return;
154     }
```

```
155
156     // Shuffle and insert the first 10 random IDs into BST
157     srand(time(0));
158     random_shuffle(ids.begin(), ids.end());
159
160     cout << "a. Insert data to BST:\n";
161     for(int i = 0; i < min(10, (int)ids.size()); i++){
162         cout << ids[i] << "\n";
163         tree.add(ids[i]);
164     }
165     cout << endl;
166
167     file.close();
168 }
169
170 int main() {
171     const string filename = "Lab08_Data_ID_Numbers_1M.csv";
172     BST tree;
173
174     loadRandomIDs(filename, tree);
175
176     int choice, id;
177     do {
178         cout << "\n1. Display Pre-order Traversal, In-order Traversal and Display Post-order Traversal\n";
179         cout << "2. Search for ID\n";
180         cout << "3. Delete an ID\n";
181         cout << "Enter your choice: ";
182         cin >> choice;
183         cout<<endl;
184
185         switch (choice) {
186             case 1:
187                 cout << "b. Pre-order Traversal of BST (DLR):\n";
188                 tree.preorder_();
189                 cout<<endl;
190
191                 cout << "c. In-order Traversal of BST (LDR):\n";
192                 tree.inorder_();
193                 cout<<endl;
194
195                 cout << "d. Post-order Traversal of BST (LRD):\n";
196                 tree.postorder_();
197                 cout<<endl;
198                 break;
199
200             case 2:
201                 cout << "e. Enter ID to search data: ";
202                 cin >> id;
203                 if (tree.search_(id))
204                     cout << "ID " << id << " exists in the BST.\n";
205                 else
206                     cout << "ID " << id << " not found in the BST.\n";
207                 break;
208
209             case 3:
210                 cout << "f. Enter ID to delete data: ";
211                 cin >> id;
212                 if (tree.search_(id)) {
213                     tree.delete_(id);
214                     cout << "ID " << id << " deleted from the BST.\n";
215                 } else {
216                     cout << "Cannot delete. ID not found!\n";
217                 }
218                 break;
219
220             default:
221                 cout << "Invalid choice. Try again.\n";
222         }
223     } while (choice != 3);
224
225     return 0;
226 }
```

Output

```
PS C:\Users\TUF\Documents\C++> cd "c:\Users\TUF\Documents\C++\Tree\" ; if ($?) { g++ test.cpp -o test } ; if ($?) { .\test }
a. Insert data to BST:
519145005
424282528
403717368
342133679
631109613
506004383
978320541
842815643
201057964
829355330


1. Display Pre-order Traversal, In-order Traversal and Display Post-order Traversal
2. Search for ID
3. Delete an ID
Enter your choice: 1

b. Pre-order Traversal of BST (DLR):
519145005 424282528 403717368 342133679 201057964 506004383 631109613 978320541 842815643 829355330

c. In-order Traversal of BST (LDR):
201057964 342133679 403717368 424282528 506004383 519145005 631109613 829355330 842815643 978320541

d. Post-order Traversal of BST (LRD):
201057964 342133679 403717368 506004383 424282528 829355330 842815643 978320541 631109613 519145005
```

```
1. Display Pre-order Traversal, In-order Traversal and Display Post-order Traversal
2. Search for ID
3. Delete an ID
Enter your choice: 2

e. Enter ID to search data: 519145005
ID 519145005 exists in the BST.

1. Display Pre-order Traversal, In-order Traversal and Display Post-order Traversal
2. Search for ID
3. Delete an ID
Enter your choice: 2

e. Enter ID to search data: 1939122
ID 1939122 not found in the BST.

1. Display Pre-order Traversal, In-order Traversal and Display Post-order Traversal
2. Search for ID
3. Delete an ID
Enter your choice: 3

f. Enter ID to delete data: 201057964
ID 201057964 deleted from the BST.
PS C:\Users\TUF\Documents\C++\Tree>
```