

| Deletion in BST in C++  |      |   |                      |   |  |
|---|------|---|----------------------|---|--|
| <pre>#include &lt;iostream&gt; using namespace std;  class Node { public:     int key;     Node *left, *right;      Node(int item) {         key = item;         left = right = nullptr;     } };  class BST { public:     Node* root;      BST() {         root = nullptr;     }      Node* insert(Node* root, int x) {         if (root == nullptr) {             return new Node(x);         }          if (x &lt; root-&gt;key) {             root-&gt;left = insert(root-&gt;left, x);         } else if (x &gt; root-&gt;key) {             root-&gt;right = insert(root-&gt;right, x);         }          return root;     }      void inorder(Node* root) {         if (root != nullptr) {             inorder(root-&gt;left);             cout &lt;&lt; root-&gt;key &lt;&lt; " ";             inorder(root-&gt;right);         }     }      Node* deleteNode(Node* root, int x) {         if (root == nullptr) {             return root;         }          if (x &lt; root-&gt;key) {             root-&gt;left = deleteNode(root-&gt;left, x);         } else if (x &gt; root-&gt;key) {             root-&gt;right = deleteNode(root-&gt;right, &gt;right, x);         } else {             if (root-&gt;left == nullptr) {                 Node* temp = root-&gt;right;                 delete root;                 return temp;             } </pre> |      | Initial Tree Structure  |                      |   |  |
|   |      | You inserted values in this order:  |                      |   |  |
|   |      | 10, 30, 20, 40, 70, 60, 80  |                      |   |  |
|   |      | Resulting BST:  |                      |   |  |
|   |      |   |                      | <pre>      10        \         30        /  \       20   40        \   \         70        /  \       60   80</pre> |  |
|   |      | 🔄 Dry Run of deleteNode(root, 20)   |                      |   |  |
|   | Step | Function Call   | Current Node         | Comparison  | Action Taken                                   |
|   | 1    | deleteNode(root, 20)  | 10                   | 20 > 10   | Go right → call deleteNode(30, 20)             |
|   | 2    | deleteNode(30, 20)  | 30                   | 20 < 30   | Go left → call deleteNode(20, 20)              |
|   | 3    | deleteNode(20, 20)  | 20                   | Match found   | Node with no children, return nullptr          |
|   | 4    | Return to Step 2  | 30                   | Set left = nullptr  | 20 is deleted from left of 30                  |
|   | 5    | Return to Step 1  | 10                   | Set right = result  | Subtree rooted at 30 is updated after deletion |
|   |      | ✓ Final Tree Structure (After Deletion)   |                      |   |  |
|   |      | <pre>      10        \         30        \   \         40   70        \   /  \         60  80</pre> |                      |   |  |
|   |      | ⬆ Inorder Traversals  |                      |   |  |
|   |      | State   | Inorder Output       |   |  |
|   |      | Before Deletion   | 10 20 30 40 60 70 80 |   |  |

|   | State          | Inorder Output    |
|---|----------------|-------------------|
| <pre>         } else if (root-&gt;right == nullptr) {             Node* temp = root-&gt;left;             delete root;             return temp;         }          Node* succ = getSuccessor(root-&gt;right);         root-&gt;key = succ-&gt;key;         root-&gt;right = deleteNode(root-&gt;right, succ-&gt;key);     }      return root; }  Node* getSuccessor(Node* root) {     Node* curr = root;     while (curr != nullptr &amp;&amp; curr-&gt;left != nullptr) {         curr = curr-&gt;left;     }     return curr; } };  int main() {     BST tree;     tree.root = tree.insert(tree.root, 10);     tree.insert(tree.root, 30);     tree.insert(tree.root, 20);     tree.insert(tree.root, 40);     tree.insert(tree.root, 70);     tree.insert(tree.root, 60);     tree.insert(tree.root, 80);      cout &lt;&lt; "Inorder traversal before deletion: ";     tree.inorder(tree.root);     cout &lt;&lt; endl;      tree.deleteNode(tree.root, 20);     cout &lt;&lt; "Inorder traversal after deletion: ";     tree.inorder(tree.root);     cout &lt;&lt; endl;      return 0; } </pre> | After Deletion | 10 30 40 60 70 80 |
| Inorder traversal before deletion: 10 20 30 40 60 70 80   |                |                   |
| Inorder traversal after deletion: 10 30 40 60 70 80   |                |                   |