```
#include <iostream>
using namespace std;

struct Node {
   char data;
   Node* left;
   Node* right;

   Node(char val) : data(val), left(nullptr), right(nullptr) {}
};

int index = 0;

Node* constructTree(const string& prefix) {
   if (index >= prefix.length())
      return nullptr;

   Node* newNode = new Node(prefix[index]);
   index++;

   if (newNode->data == '+' || newNode->data == '-' || newNode->data == '*' || newNode->data ==
'/') {
      newNode->left = constructTree(prefix);
      newNode->right = constructTree(prefix);
   }
   return newNode;
}

void postOrderTraversal(Node* root) {
   if (root == nullptr)
      return;
```

```
    postOrderTraversal(root->left);
    postOrderTraversal(root->right);
    cout << root->data << " ";
}

void deleteTree(Node* root) {
    if (root == nullptr)
        return;

    deleteTree(root->left);
    deleteTree(root->right);
    delete root;
}

int main() {
    Node* root = nullptr;
    string prefix;
    int choice;

    cout << "### DSAL PRACTICAL 05 (B - 07) ###\n";

    do {
        cout << "\n### Menu ###\n";
        cout << "1. Enter Prefix Expression\n";
        cout << "2. Display Post-order Traversal\n";
        cout << "3. Delete Tree\n";
        cout << "4. Exit\n";
        cout << "Enter your choice: ";
        cin >> choice;
```

```
switch (choice) {
case 1:
   cout << "Enter prefix expression: ";
   cin >> prefix;
   index = 0;
   root = constructTree(prefix);
   break;

case 2:
   if (root) {
      cout << "Post-order Traversal: ";
      postOrderTraversal(root);
      cout << endl;
   } else {
      cout << "Tree is empty!\n";
   }
   break;

case 3:
   deleteTree(root);
   root = nullptr;
   cout << "Tree deleted.\n";
   break;

case 4:
   cout << "Exiting program.\n";
   break;

default:
   cout << "Invalid choice. Please try again.\n";
}
```

```
    } while (choice != 4);


    return 0;
}
```