

## ASSIGNMENT 7

```
#include <bits/stdc++.h>

using namespace std;

class Node {
private:
    char op;
    Node* left;
    Node* right;

public:
    Node(char o): op(o), left(nullptr), right(nullptr) {}

    ~Node() {
        delete left;
        delete right;
        this->left = nullptr;
        this->right = nullptr;
        this->op = '\0';
    }

    friend class AbstractSyntaxTree;
};

class AbstractSyntaxTree {
private:
    Node* root;

public:
    AbstractSyntaxTree(): root(nullptr) {}
```

```

void fromPrefixNotation(const string& expression) {
    stack<Node*> operations;
    Node* curr = root;
    for (char c : expression) {
        if (!curr) {
            curr = new Node(c);
            if (!root) {root = curr;}
            operations.emplace(curr);
            continue;
        }

        if (!curr->left) {
            curr->left = new Node(c);
            // Change current operation if c is an operator
            if (!isalnum(c)) {
                curr = curr->left;
                operations.emplace(curr);
            }
        } else if (!curr->right) {
            curr->right = new Node(c);
            // Remove operator from stack because it already has 2 operands
            operations.pop();
            if (!isalnum(c)) {
                curr = curr->right;
                operations.emplace(curr);
            } else {
                if (!operations.empty())
                    curr = operations.top();
                else {
                    break;
                }
            }
        }
    }
}

```

```

        }
    }
}
}
}

```

```

void postOrder() {
    if (!root) {
        return;
    }
}

```

```

stack<Node*> s, order;
s.emplace(root);
Node* temp;
while (!s.empty()) {
    temp = s.top();
    s.pop();
    order.emplace(temp);

    if (temp->left) {
        s.emplace(temp->left);
    }
    if (temp->right) {
        s.emplace(temp->right);
    }
}
}

```

```

while (!order.empty()) {
    auto curr = order.top();
    order.pop();
    cout << curr->op;
}

```

```

    }

    cout << endl;
}

void clear() {
    delete this->root;
    this->root = nullptr;
}

void display() {
    if (!root) {
        cout << "[]" << endl;
        return;
    }
    cout << "ARRAY REPRESENTATION: [ " << flush;

    queue<Node*> level;
    vector<string> representation;
    level.push(root);
    representation.emplace_back(1, root->op);

    Node* temp;
    while(!level.empty()) {
        temp = level.front();
        level.pop();

        if (!temp) {
            representation.emplace_back("NULL");
            representation.emplace_back("NULL");
            continue;

```

```

    }

    if (temp->left) {
        representation.emplace_back(1, temp->left->op);
        level.push(temp->left);
    } else {
        level.push(nullptr);
        representation.emplace_back("NULL");
    }

    if (temp->right) {
        representation.emplace_back(1, temp->right->op);
        level.push(temp->right);
    } else {
        level.push(nullptr);
        representation.emplace_back("NULL");
    }
}

for (auto& item : representation) {
    cout << item << ", ";
}

cout << "]" << endl;
}

};

```

```

int main() {
    AbstractSyntaxTree ast;
    ast.fromPrefixNotation("++a*bc/def");
    ast.display();
}

```

```

ast.postOrder();

ast.clear();

cout << "After delete operation: ";

ast.display();

return 0;

}0;

}

```

OUTPUT :

The screenshot shows the OnlineGDB C++ compiler interface. The code being executed is as follows:

```

139     }
140     }
141     for (auto& item : representation) {
142         cout << item << ", ";
143     }
144     cout << "]" << endl;
145 }
146
147 };
148
149
150 int main() {
151     AbstractSyntaxTree ast;
152     ast.fromPrefixNotation("+-a*bc/def");
153     ast.display();
154     ast.postOrder();
155     ast.clear();
156     cout << "After delete operation: ";
157     ast.display();
158     return 0;
159 }

```

The output of the program is shown in the console:

```

ARRAY REPRESENTATION: [ +, -, /, *, NULL, NULL, a, *, d, e, NULL, NULL, NULL, NULL, NULL, NULL, b, c, NULL, NULL, NULL, NULL, NULL,
NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, N
ULL, ]
abc*-de/-f+
After delete operation: []

...Program finished with exit code 0
Press ENTER to exit console

```