ASSIGNMENT 4 – EXPRESSION TREE

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

struct node
{
    char data;
    struct node *left;
    struct node *right;
};

class stack
{
    node *data[10];
    int top;

public:
    stack()
    {
        top = -1;
    }
    node *topdata()
    {
        return (data[top]);
    }
    void push(node *p)
    {
        data[++top] = p;
    }
    node *pop()
    {
        return (data[top--]);
    }
};

node *create(char postfix[10])
{
    node *p;
    stack s;
    for (int i = 0; postfix[i] != '\0'; i++)
    {
        char token = postfix[i];
        if (isalnum(token))
        {
            p = new node;
            p->data = token;
            p->right = NULL;
            p->left = NULL;
            s.push(p);
        }
        else
        {
            p = new node;
            p->data = token;
            p->right = s.pop();
            p->left = s.pop();
            s.push(p);
        }
    }
    return s.pop();
}

void preorder(node *p)
{
    if (p != NULL)
    {
        cout << p->data;
        preorder(p->left);
        preorder(p->right);
    }
}

void inorder(node *p)
{
    if (p != NULL)
    {
        inorder(p->left);
        cout << p->data;
        inorder(p->right);
    }
}

void postorder(node *p)
{
    if (p != NULL)
    {
        postorder(p->left);
        postorder(p->right);
        cout << p->data;
    }
}

int main()
{
    node *r = NULL;
    int ch;
    char postfix[10];
    do
    {
        cout << "\n1.CONSTRUCT TREE \n2.PREORDER
\n3.INORDER \n4.POSTORDER \n5.EXIT";
        cout << "\nEnter your choice: ";
        cin >> ch;

        switch (ch)
        {
        case 1:
            cout << "\nEnter your postfix expression: " <<
endl;
            cin >> postfix;

            r = create(postfix);
            cout << "\n Tree created successfully!!!";
            break;

        case 2:
            cout << "Preorder traversal: ";
            preorder(r);
            break;

        case 3:
            cout << "\n INORDER TRAVERSAL: ";
            inorder(r);
            break;

        case 4:
            cout << "POSTORDER TRAVERSAL: ";
            postorder(r);
            break;
        }
    } while (ch != 5);

    return 0;
}
```

```
OUTPUT:
1.CONSTRUCT TREE
2.PREORDER
3.INORDER
4.POSTORDER
5.EXIT
Enter your choice: 1

Enter your postfix expression:
ABC+-

 Tree created successfully!!!
1.CONSTRUCT TREE
2.PREORDER
3.INORDER
4.POSTORDER
5.EXIT
Enter your choice: 2
Preorder traversal: -A+BC
1.CONSTRUCT TREE
2.PREORDER
3.INORDER
4.POSTORDER
5.EXIT
Enter your choice: 3

 INORDER TRAVERSAL: A-B+C
1.CONSTRUCT TREE
2.PREORDER
3.INORDER
4.POSTORDER
5.EXIT
Enter your choice: 4
POSTORDER TRAVERSAL: ABC+-
1.CONSTRUCT TREE
2.PREORDER
3.INORDER
```

```
4.POSTORDER
5.EXIT
```