**AIM: Construct an expression tree from the given prefix and traverse it using post order traversal and then delete the entire tree.**

**PROGRAM**

#include <iostream>

#include <string.h>

using namespace std;

struct node

{

char data;

node \*left;

node \*right;

};

class tree

{

char prefix[20];

public:

node \*top;

void expression(char[]);

void display(node \*);

void non\_rec\_postorder(node \*);

void del(node \*);

};

class stack1

{

node \*data[30];

int top;

public:

stack1()

{

top = -1;

}

int empty()

{

if (top == -1)

return 1;

return 0;

}

void push(node \*p)

{

data[++top] = p;

}

node \*pop()

{

return (data[top--]);

}

};

void tree::expression(char prefix[])

{

char c;

stack1 s;

node \*t1, \*t2;

int len, i;

len = strlen(prefix);

for (i = len - 1; i >= 0; i--)

{

top = new node;

top->left = NULL;

top->right = NULL;

if (isalpha(prefix[i]))

{

top->data = prefix[i];

s.push(top);

}

else if (prefix[i] == '+' || prefix[i] == '\*' || prefix[i] == '-' || prefix[i] == '/')

{

t2 = s.pop();

t1 = s.pop();

top->data = prefix[i];

top->left = t2;

top->right = t1;

s.push(top);

}

}

top = s.pop();

}

void tree::display(node \*root)

{

if (root != NULL)

{

cout << root->data;

display(root->left);

display(root->right);

}

}

void tree::non\_rec\_postorder(node \*top)

{

stack1 s1, s2; /\*stack s1 is being used for flag . A NULL data implies that the right subtree has not been visited \*/

node \*T = top;

cout << "\n";

s1.push(T);

while (!s1.empty())

{

T = s1.pop();

s2.push(T);

if (T->left != NULL)

s1.push(T->left);

if (T->right != NULL)

s1.push(T->right);

}

while (!s2.empty())

{

top = s2.pop();

cout << top->data;

}

}

void tree::del(node \*node)

{

if (node == NULL)

return;

/\* first delete both subtrees \*/

del(node->left);

del(node->right);

/\* then delete the node \*/

cout <<endl<<"Deleting node : " << node->data<<endl;

free(node);

}

int main()

{

char expr[20];

tree t;

cout <<"Enter prefix Expression : ";

cin >> expr;

cout << expr;

t.expression(expr);

//t.display(t.top);

//cout<<endl;

t.non\_rec\_postorder(t.top);

t.del(t.top);

// t.display(t.top);

}