Name: Khushal Patil Class: SE-II (R-Batch)

Roll No.: 64

**Code:**

#include<iostream>

using namespace std;

#include<string.h>

struct node{

    char data;

    node \*left;

    node \*right;

};

class tree{

    char prefix[50];

    public:

        node \*top;

        void expression(char []);

        void display(node \*);

        void rec\_d(node \*);

        void deletion(node \*node);

};

class stack1{

    public:

        node \*data[30];

        int top;

        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::rec\_d(node \*root){

    if(root!=NULL){

        cout<<root->data;

        rec\_d(root->left);

        rec\_d(root->right);

    }

}

void tree::display(node \*top){

    stack1 s1,s2;

    node \*T = top;

    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;

    }

    cout<<endl;

}

void tree::deletion(node \*node){

    if(node==NULL){

        return;

    }

    deletion(node->left);

    deletion(node->right);

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

    free(node);

}

int main(){

    tree t;

    char exp1[20];

    int ch;

    do{

        cout<<"1. Enter Prefix Expression"<<endl;

        cout<<"2. Display Postfix Expression"<<endl;

        cout<<"3. Deletion"<<endl;

        cout<<"4. Exit"<<endl;

        cout<<"Enter choice: ";

        cin>>ch;

        switch(ch){

            case 1:

                cout<<"enter the expression:";

               cin>>exp1;

                t.expression(exp1);

                break;

            case 2:

                t.display(t.top);

                break;

            case 3:

cout<<"Postfix Expression:- ";

                t.deletion(t.top);

                break;

            case 4:

                cout<<"Thank You..."<<endl;

                break;

            default:

                cout<<"Invalid Choice...";

                break;

        }

    }while(ch!=4);

}

**Output:**

1. Enter Prefix Expression

2. Display Postfix Expression

3. Deletion

4. Exit

Enter choice: 1

enter the expression:+--a\*bc/def

1. Enter Prefix Expression

2. Display Postfix Expression

3. Deletion

4. Exit

Enter choice: 2

Postfix Expression:- abc\*-de/-f+

1. Enter Prefix Expression

2. Display Postfix Expression

3. Deletion

4. Exit

Enter choice: 3

Deleting Node: a

Deleting Node: b

Deleting Node: c

Deleting Node: \*

Deleting Node: -

Deleting Node: d

Deleting Node: e

Deleting Node: /

Deleting Node: -

Deleting Node: f

Deleting Node: +

1. Enter Prefix Expression

2. Display Postfix Expression

3. Deletion

4. Exit

Enter choice: 4

Thank You...