EXPERIMENT-5

INPUT:-

#include <iostream>

#include <cstdlib>

#include <cstdio>

#include <cstring>

using namespace std;

class TreeN {

public:

char d;

TreeN \*l, \*r;

TreeN(char d) {

this->d = d;

this->l = NULL;

this->r = NULL;

}

};

class StackNod {

public: TreeN \*treeN;

StackNod \*n;

StackNod(TreeN\*treeN) {

this->treeN = treeN;

n = NULL;

}

};

class ExpressionTree {

private: StackNod \*top;

public: ExpressionTree() {

top = NULL;

}

void clear() {

top = NULL;

}

void push(TreeN \*ptr) {

if (top == NULL)

top = new StackNod(ptr);

else {

StackNod \*nptr = new StackNod(ptr);

nptr->n = top;

top = nptr;

}

}

TreeN \*pop() {

if (top == NULL) {

cout<<"Underflow"<<endl;

} else {

TreeN \*ptr = top->treeN;

top = top->n;

return ptr;

}

}

TreeN \*peek() {

return top->treeN;

}

void insert(char val) {

if (isDigit(val)) {

TreeN \*nptr = new TreeN(val);

push(nptr);

} else if (isOperator(val)) {

TreeN \*nptr = new TreeN(val);

nptr->l = pop();

nptr->r= pop();

push(nptr);

} else {

cout<<"Invalid Expression"<<endl;

return;

}

}

bool isDigit(char ch) {

return ch >= 'a' && ch <= 'z';

}

bool isOperator(char ch) {

return ch == '+' || ch == '-' || ch == '\*' || ch == '/';

}

int toDigit(char ch) {

return ch - 'a';

}

void buildTree(string eqn) {

for (int i = eqn.length() - 1; i >= 0; i--)

insert(eqn[i]);

}

void postfix() {

postOrder(peek());

}

void postOrder(TreeN\*ptr) {

if (ptr != NULL) {

postOrder(ptr->l);

postOrder(ptr->r);

cout<<ptr->d;

}

}

void infix() {

inOrder(peek());

}

void inOrder(TreeN \*ptr) {

if (ptr != NULL) {

inOrder(ptr->l);

cout<<ptr->d;

inOrder(ptr->r);

}

}

void prefix() {

preOrder(peek());

}

void preOrder(TreeN \*ptr) {

if (ptr != NULL) {

cout<<ptr->d;

preOrder(ptr->l);

preOrder(ptr->r);

}

}

};

int main() {

string s;

ExpressionTree et;

cout<<"\nEnter equation in Prefix form: ";

cin>>s;

et.buildTree(s);

cout<<"\nPrefix : ";

et.prefix();

cout<<"\n\nInfix : ";

et.infix();

cout<<"\n\nPostfix : ";

et.postfix();

}

OUTPUT:-

Enter equation in Prefix form: +--a\*bc/def

Prefix : +--a\*bc/def

Infix : a-b\*c-d/e+f

Postfix : abc\*-de/-f+

=== Code Execution Successful ===