**Assignment No : 4**

Name: Rasika Mahure

PRN:72018258B

Div:B

Roll No: S1951093

**Aim :-**

Construct an Expression Tree from postfix and prefix expression. Perform recursive and non- recursive In-order, pre-order and post-order traversals.

**//PROGRAM**

#include <iostream> using namespace std; typedef struct node { char data; struct node \*left; struct node \*right; } node; typedef struct stacknode { struct node \*data; struct stacknode

\*next;

} stacknode; class stack { stacknode \*top; public: stack() { top = NULL; } int isempty() { if (top == NULL) return 1; return 0; } void push(node \*a) { stacknode \*p; p = new stacknode(); p->data = a; p->next = top; top = p; } node \*pop() { stacknode \*p; node

\*x; x = top->data; p = top; top = top->next; return x; } node \*topdata()

{ return top-

>data;

}

}; node \*create\_post(char postfix[10]); node \*create\_pre(char prefix[10]); void inorder(node \*p); void preorder(node \*p); void postorder(node \*p); void inorder\_non\_recursive(node \*t); void preorder\_non\_recursive(node \*t); void postorder\_non\_recursive(node \*t);

int main() { node \*r = NULL, \*r1; char postfix[10], prefix[10]; int

x; int ch, choice; do { cout << "\n\t\*\*TREE

OPERATIONS\*\*\n1.Construct tree from postfix expressio n/prefix expression

\n2.Inorder Traversal of tree \n3.Preorder Traversal of tree

\n4.Postorder Traversal of tree \n5.Exit"; cout << "\nEnter your choice : "; cin >> ch; switch (ch) { case 1:

cout << "\nENTER CHOICE:\n1.Postfix expression\n2.Prefix expression\n

choice="; cin >> choice; if (choice == 1)

{

cout << "\nEnter postfix expression : ";

cin >> postfix; r = create\_post(postfix); cout << "\nTree created successfully\n"; } else if (choice == 2)

{ cout << "\nEnter prefix expression : "; cin >> prefix; r = create\_pre(prefix); cout << "\nTree created successfully\n";

} else { cout << "\nInvalid choice";

} break; case 2:

cout << "\nInorder Traversal of tree : ";

inorder(r); cout << "\nInorder Traversal of tree without recursion : "; inorder\_non\_recursive(r); cout << "\n"; break; case 3:

cout << "\nPreorder Traversal of tree : ";

preorder(r); cout << "\nPreorder traversal of tree without recursion : "; preorder\_non\_recursive(r); cout << "\n"; break; case 4:

cout << "\nPostorder Traversal of tree : ";

postorder(r); cout << "\nPostorder traversal of tree without recursion : "; postorder\_non\_recursive(r); cout << "\n"; break; default:

cout << "\nInvalid choice";

}

} while (ch != 5); return 0; } node \*create\_post(char postfix[10])

{ node \*p; stack s; for (int i = 0; postfix[i] != '\0'; i++)

{ for (int i = 0; postfix[i] !=

'\0'; i++)

{ char token = postfix[i]; if

(isalnum(token))

{ p

= new node(); p>data = token; p->left = NULL; p->right = 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(); } node \*create\_pre(char prefix[10]) { node \*p; stack s; int i;

for (i = 0; prefix[i] != '\0'; i++)

{ } i = i - 1; for (; i >= 0;

i--) { { char token = prefix[i]; if

(isalnum(token))

{ p

= new node(); p>data = token; p->left = NULL; p->right = NULL;

s.push(p); } else { p = new node(); p>data = token; p-

>left = s.pop(); p-

>right = s.pop();

s.push(p); } } } return s.pop(); } void inorder(node \*p) { if (p != NULL) { inorder(p->left); cout << p->data; inorder(p->right);

}

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

} }

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

}

} void preorder\_non\_recursive(node \*t)

{ stack s; while (t != NULL) { cout << t->data;

s.push(t); t = t->left; } while (s.isempty() != 1) { t = s.pop(); t = t->right; while (t != NULL) { cout << t->data;

s.push(t); t = t->left; }

}

} void inorder\_non\_recursive(node

\*t) { stack s; while (t != NULL)

{

s.push(t); t = t->left; } while (s.isempty() != 1)

{ t =

s.pop(); cout << t->data; t = t-

>right; while (t

!= NULL) {

s.push(t); t = t->left; }

}

} void postorder\_non\_recursive(node \*t) { stack s, s1; node \*t1; while

(t != NULL)

{

s.push(t); s1.push(NULL); t = t->left; } while (s.isempty() != 1)

{ t =

s.pop(); t1 = s1.pop(); if

(t1 == NULL)

{

s.push(t); s1.push((node \*)1);

t = t->right; while

(t != NULL) {

s.push(t); s1.push(NULL); t

= t->left;

} } else cout << t-

>data;

}

}

**//OUTPUT**

\*\*TREE OPERATIONS\*\*

1.Construct tree from postfix expression/prefix expression

2.Inorder Traversal of tree

3.Preorder Traversal of tree

4.Postorder Traversal of tree

5.Exit

Enter your choice : 1

ENTER CHOICE:

1.Postfix expression 2.Prefix expression choice=1

Enter postfix expression : 51-24\*+92/+

Tree created successfully

\*\*TREE OPERATIONS\*\*

1.Construct tree from postfix expression/prefix expression

2.Inorder Traversal of tree

3.Preorder Traversal of tree

4.Postorder Traversal of tree

5.Exit

Enter your choice : 2

Inorder Traversal of tree : 5-1+2\*4+9/2

Inorder Traversal of tree without recursion : 5-1+2\*4+9/2

\*\*TREE OPERATIONS\*\*

1.Construct tree from postfix expression/prefix expression

2.Inorder Traversal of tree

3.Preorder Traversal of tree

4.Postorder Traversal of tree

5.Exit

Enter your choice : 3

Preorder Traversal of tree : ++-51\*24/92

Preorder traversal of tree without recursion : ++-51\*24/92

\*\*TREE OPERATIONS\*\*

1.Construct tree from postfix expression/prefix expression

2.Inorder Traversal of tree

3.Preorder Traversal of tree

4.Postorder Traversal of tree

5.Exit

Enter your choice : 4

Postorder Traversal of tree : 51-24\*+92/+

Postorder traversal of tree without recursion : 51-24\*+92/+

\*\*TREE OPERATIONS\*\*

1.Construct tree from postfix expression/prefix expression

2.Inorder Traversal of tree

3.Preorder Traversal of tree

4.Postorder Traversal of tree

5.Exit

Enter your choice : 1

ENTER CHOICE:

1.Postfix expression 2.Prefix expression choice=2

Enter prefix expression : ++-51\*24/92

Tree created successfully

\*\*TREE OPERATIONS\*\*

1.Construct tree from postfix expression/prefix expression

2.Inorder Traversal of tree

3.Preorder Traversal of tree

4.Postorder Traversal of tree

5.Exit

Enter your choice : 2

Inorder Traversal of tree : 5-1+2\*4+9/2

Inorder Traversal of tree without recursion : 5-1+2\*4+9/2

\*\*TREE OPERATIONS\*\*

1.Construct tree from postfix expression/prefix expression

2.Inorder Traversal of tree

3.Preorder Traversal of tree

4.Postorder Traversal of tree

5.Exit

Enter your choice : 3

Preorder Traversal of tree : ++-51\*24/92

Preorder traversal of tree without recursion : ++-51\*24/92

\*\*TREE OPERATIONS\*\*

1.Construct tree from postfix expression/prefix expression

2.Inorder Traversal of tree

3.Preorder Traversal of tree

4.Postorder Traversal of tree

5.Exit

Enter your choice : 4

Postorder Traversal of tree : 51-24\*+92/+

Postorder traversal of tree without recursion : 51-24\*+92/+

\*\*TREE OPERATIONS\*\*

1.Construct tree from postfix expression/prefix expression

2.Inorder Traversal of tree

3.Preorder Traversal of tree

4.Postorder Traversal of tree 5.Exit

Enter your choice : 5

Invalid choice