Assignment No. 7

**TITLE : Expression Tree**

Name : METHIKA M Date : 14/10/2025

UNo : UIT2024936

Class: IT- I DIV

Batch : I2  
  
  
**SOURCE CODE**  
/\*

\* exp\_tree.c

\*

\* Created on: 13-Oct-2025

\* Author: Admin

\*/

#include<stdio.h>

#include <stdlib.h>

struct tree

{

struct tree \*left;

char ch;

struct tree \*right;

};

struct stack

{

struct tree \*address;

struct stack \*next;

};

int push(struct stack \*\*ftop, struct tree \*\*fidata);

int pop(struct stack \*\*ftop, struct tree \*\*fodata);

void create(struct tree \*\*froot, char fpostfix[10]);

void inorder(struct tree \*\*froot);

void preorder(struct tree \*\*froot);

int push(struct stack \*\*ftop, struct tree \*\*fidata)

{

struct stack \*top=NULL;

struct stack \*newnode=NULL;

int fsuccess=0;

top=\*ftop;

newnode=(struct stack \*)calloc(1,sizeof(struct stack));

if (newnode==NULL)

{

printf("Memory allocation failed. Stack full");

fsuccess=1;

}

else

{

newnode->address=\*fidata;

if (top==NULL)

{

top=newnode;

}

else

{

newnode->next=top;

top=newnode;

}

}

\*ftop=top;

return fsuccess;

}

int pop(struct stack \*\*ftop, struct tree \*\*fodata)

{

struct stack \*top=NULL;

struct stack \*X=NULL;

int fsuccess=0;

top=\*ftop;

if (top==NULL)

{

printf("Nothing to pop. Stack Empty");

fsuccess=1;

}

else

{

\*fodata=top->address;

X=top;

top=top->next;

free(X);

}

\*ftop=top;

return fsuccess;

}

void create(struct tree \*\*froot, char fpostfix[10])

{

struct tree \*root=NULL;

struct tree \*newnode=NULL;

struct stack \*top=NULL;

struct tree \*oprnd1=NULL;

struct tree \*oprnd2=NULL;

int i=0;

root=\*froot;

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

{

newnode=(struct tree \*)calloc (1,sizeof(struct tree));

newnode->left=NULL;

newnode->right=NULL;

if((fpostfix[i]>='a' && fpostfix[i]<='z') || (fpostfix[i]>='A' && fpostfix[i]<='Z'))

{

newnode->ch=fpostfix[i];

push(&top,&newnode);

}

else

{

pop(&top,&oprnd2);

pop(&top,&oprnd1);

newnode->ch=fpostfix[i];

newnode->left=oprnd1;

newnode->right=oprnd2;

push(&top,&newnode);

}

}

pop(&top,froot);

}

void inorder(struct tree \*\*froot)

{

struct tree \*root=NULL;

root=\*froot;

if (root==NULL)

{

return;

}

inorder(&root->left);

printf("%c",root->ch);

inorder(&root->right);

\*froot=root;

}

void preorder(struct tree \*\*froot)

{

struct tree \*root=NULL;

root=\*froot;

if (root==NULL)

{

return;

}

printf("%c",root->ch);

preorder(&root->left);

preorder(&root->right);

\*froot=root;

}

void postorder(struct tree \*\*froot)

{

struct tree \*root=NULL;

root=\*froot;

if (root==NULL)

{

return;

}

postorder(&root->left);

postorder(&root->right);

printf("%c",root->ch);

\*froot=root;

}

int menu(void)

{

int choice;

do

{

printf("\nChoose Traversal Type");

printf("\n1. Inorder \n2. Preorder \n3. Postorder \n0. Exit");

printf("\nEnter your choice: ");

fflush(stdout);

scanf("%d", &choice);

}while(choice<0 || choice >4);

return choice;

}

int main(void)

{

struct tree \*root=NULL;

char postfix[10];

printf("Enter the postfix expression: ");

fflush(stdout);

scanf("%[^\n]%\*c",postfix);

create(&root,postfix);

int traversal=0;

do

{

traversal=menu();

switch(traversal)

{

case 1:

printf("\nThe infix expression is: ");

inorder(&root);

break;

case 2:

printf("\nThe prefix expression is: ");

preorder(&root);

break;

case 3:

printf("\nThe postfix expression is: ");

postorder(&root);

break;

case 0:

printf("\nExiting...");

printf("\nThank you");

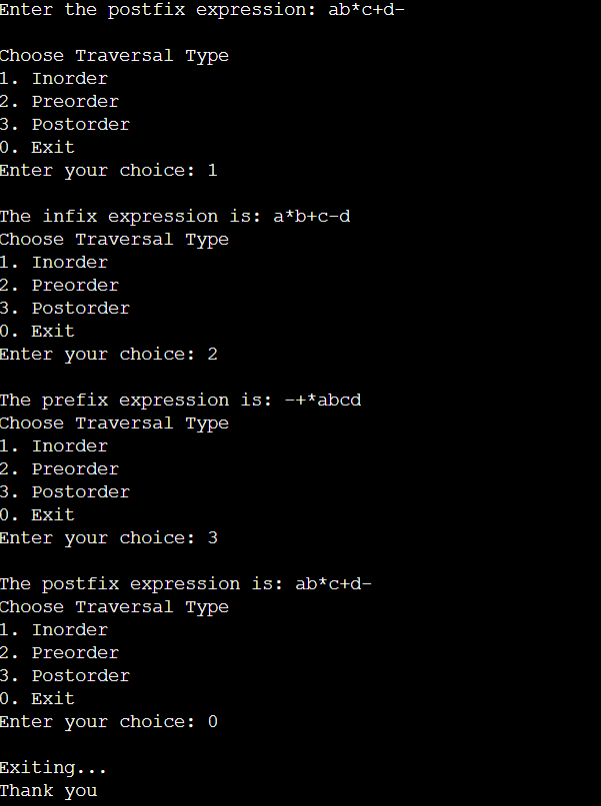
break;

}

}while(traversal!=0);

return 0;

}

**OUTPUT  
  
**