**Exercise 7:** **Expression Tree and Application**

**adt.h:**

struct node

{

char c;

struct node \*left,\*right;

};

struct stack

{

struct node \*s[30];

int size,top;

};

**impl.h:**

#include "adt.h"

#include<stdlib.h>

#include<stdio.h>

#include<string.h>

void init(struct stack \*s1)

{

s1->top=-1;

s1->size=30;

}

int isFull(struct stack s1)

{

if(s1.size-1==s1.top)

return 1;

return 0;

}

int isEmpty(struct stack s1)

{

if(s1.top==-1)

return 1;

return 0;

}

void push(struct stack \*s1,struct node \*ele)

{

if(isFull(\*s1))

printf("\nOverflow");

else

s1->s[++s1->top]=ele;

}

struct node\* pop(struct stack \*s1)

{

if(isEmpty(\*s1))

return NULL;

else

return s1->s[s1->top--];

}

struct node\* construct(char s[])

{

struct stack sta;

init(&sta);

struct node \*tmp;

tmp->right=tmp->left=NULL;

for(int i=0;i<strlen(s);i++)

{

tmp=(struct node\*)malloc(sizeof(struct node));

tmp->c=s[i];

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

{

tmp->right=pop(&sta);

tmp->left=pop(&sta);

}

push(&sta,tmp);

}

return pop(&sta);

}

void inorder(struct node \*ptr)

{

if(ptr==NULL)

{

printf("\nEmpty!!!");

return;

}

if(ptr->left!=NULL)

inorder(ptr->left);

printf(" %c",ptr->c);

if(ptr->right!=NULL)

inorder(ptr->right);

}

void preorder(struct node \*ptr)

{

if(ptr==NULL)

{

printf("\nEmpty!!!");

return;

}

printf(" %c",ptr->c);

if(ptr->left!=NULL)

preorder(ptr->left);

if(ptr->right!=NULL)

preorder(ptr->right);

}

void postorder(struct node \*ptr)

{

if(ptr==NULL)

{

printf("\nEmpty!!!");

return;

}

if(ptr->left!=NULL)

postorder(ptr->left);

if(ptr->right!=NULL)

postorder(ptr->right);

printf(" %c",ptr->c);

}

int eval(struct node \*T)

{

while(T!=NULL)

{

if(T->c=='\*')

{

return (eval(T->left))\*(eval(T->right));

}

if(T->c=='+')

{

return eval(T->left)+eval(T->right);

}

if(T->c=='/')

{

return eval(T->left)/eval(T->right);

}

if(T->c=='-')

{

return eval(T->left)-eval(T->right);

}

return T->c-48;

}

//return T;

}

**appl.c:**

#include<stdio.h>

#include "impl.h"

int main()

{

struct node \*root=(struct node\*)malloc(sizeof(struct node));

char s[20];

printf("\nEnter the postorder expression: ");

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

root=construct(s);

printf("\nInorder: ");

inorder(root);

printf("\nPreorder");

preorder(root);

printf("\nPostorder:");

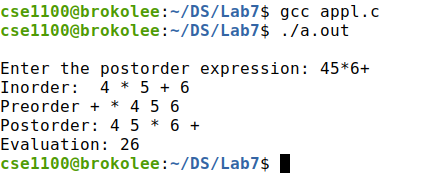
postorder(root);

int x=eval(root);

printf("\nEvaluation: %d\n",x);

}

*Sample I/O:*

**