

/\*Implement stack as an abstract data type using singly linked list and use this ADT for conversion of infix expression to postfix,

prefix, and evaluation of postfix and prefix expression\*/

```
#include<stdlib.h>
```

```
#include<iostream>
```

```
#include<string.h>
```

```
#define max 30
```

```
using namespace std;
```

```
struct node
```

```
{
```

```
    char data;
```

```
    struct node *next;
```

```
};
```

```
class stack
```

```
{
```

```
    node *top;
```

```
    char x;
```

```
    public:
```

```
        stack()
```

```
        {
```

```
            top= NULL;
```

```
        }
```

```
        int empty ()
```

```
        {
```

```
            if (top==NULL)
```

```
            {
```

```
                return(1);
```

```
            }
```

```
        else
        {
            return(0);
        }
    }
}
```

```
void push(char x)
{
    node *p;
    p=new node;
    p->data=x;
    p->next=top;
    top=p;
}
```

```
char pop()
{
    if(!empty())
    {
        node *p;
        p=new node;
        p=top;
        top=top->next;
        x=p->data;
        return(x);
    }
    else
    {
        cout<<"Stack is empty now!!"<<endl;
        return(0);
    }
}
```

```

char pop1()
{
    if(!empty())
    {
        node *p;
        p=new node;
        p=top;
        //top=top->next;
        x=p->data;
        //delete p;
        return(x);
    }
    else
    {
        cout<<"Stack is empty now!!"<<endl;
        return(0);
    }
}
};

```

```

int precedence (char x);
void infix_to_prefix(char infix[],char prefix[]);
void infix_to_postfix(char infix[],char postfix[]);
void eval_prefix(char prefix[]);
void eval_postfix(char postfix[]) ;
int evaluate(char x,int op1, int op2) ;

```

```

void infix_to_prefix(char infix[],char prefix[])
{
    int i,j;

```

```

char temp, in1[30];
cout<<"\nEntered infix is...";
for(i=0;i<=strlen(infix)-1;i++)
{
    cout<<infix[i];
}
cout<<endl;
cout<<endl;
for (i=strlen(infix)-1,j=0;i>=0;i--,j++)
in1[j]=infix[i];
in1[j]='\0';

for (i=0;in1[i]!='\0';i++)
{
    if(in1[i]=='(')
        in1[i]=')';
    else if (in1[i]==')')
        in1[i]='(';
}

infix_to_postfix(in1,prefix);

for (i=0,j=strlen(prefix)-1;i<j;i++,j--)
{
    temp=prefix [i];
    prefix[i]=prefix[j];
    prefix[j]=temp;
}
}

void infix_to_postfix(char infix[],char postfix[])

```

```

{

    stack s;

    node *top;

    char x;

    int i,j;

    char token;

    cout<<"\nEntered infix is...";

    for(i=0;i<=strlen(infix)-1;i++)

    {

        cout<<infix[i];

    }

    cout<<endl;

    cout<<endl;


    j=0;

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

    {

        token=infix[i];

        if(isalnum(token))

        {

            postfix[j++]=token;

        }

        else if (token=='(')

        {

            s.push('(');

        }

        else if (token==')')

        {

            while((x=s.pop())!='(')

            {

                postfix[j++]=x;

            }

        }

    }

}

```

```

        }
    }
    else
    {
        x=s.pop1();
        while(precedence(token)<precedence(x) && !s.empty())
        {
            x=s.pop();
            postfix[j++]=x;
        }
        s.push(token);
    }
}
while(!s.empty())
{
    x=s.pop();
    postfix[j++]=x;
}
postfix[j]='\0';
}

```

```

int precedence(char x)
{
    if(x == '(')
    {
        return 0;
    }
    else if(x == '*' || x == '/' || x == '%')
    {
        return 2;
    }
}

```

```

else if(x == '+' || x == '-')
{
    return 1;
}

else
return 3;
}

```

```

void eval_prefix(char prefix[])

```

```

{
    stack s;
    char x;
    int op1, op2, val,i;
    for (i=strlen(prefix)-1;i>=0;i--)
    {
        x=prefix[i];
        if (isalpha(x))
        {
            cout<<"\nEnter the value of "<<x<<":";
            cin>>val;
            s.push(val);
        }
        else
        {
            op1=s.pop();
            op2=s.pop();
            val=evaluate(x, op1, op2);
            s.push(val);
        }
    }
    val=s.pop();
}

```

```

        cout<<"\nValue of expression is "<<val;
    }

void eval_postfix(char postfix[])
{
    stack s;
    char x;
    int op1, op2, val,i;
    for (i=0;postfix[i]!='\0';i++)
    {
        x=postfix[i];
        if (isalpha(x))
        {
            cout<<"\nEnter the value of "<<x<<":";
            cin>>val;
            s.push(val);
        }
        else
        {
            op2=s.pop();
            op1=s.pop();
            val=evaluate(x, op1, op2);
            s.push(val);
        }
    }
    val=s.pop();
    cout<<"\nValue of expression is "<<val;
}

```

```

int evaluate (char x, int op1, int op2)
{

```



```

        if (x=='+'){
            return(op1+op2);
        }
        if (x=='-')
        {
            return(op1-op2);
        }
        if (x=='*')
        {
            return (op1*op2);
        }
        if (x=='/')
        {
            return(op1/op2);
        }

        if (x=='%')
            return(op1%op2);

    }

int main(){

    char infix[30],prefix[30],postfix[30];
    int ch;
    do
    {
        cout<<"\n\nMenu :\n1.Infix-Prefix-evaluation\n2.Infix-Postfix-
evaluation\n3.Exit\nEnter the choice:";
        cin>>ch;
    }

```

```

switch(ch){
    case 1:
        cout<<"\nEnter the infix expression: "<<endl;
        cin>>infix;
        infix_to_prefix(infix,prefix);
        cout<<"\nPrefix expression is: "<<prefix<<endl;
        cout<<"\n\nEvaluation of Prefix expression: ";
        eval_prefix(prefix);

        break;
    case 2:
        cout<<"\nEnter the infix expression: "<<endl;
        cin>>infix;
        infix_to_postfix(infix,postfix);
        cout<<" \nPostfix expression is: "<<postfix<<endl;
        cout<<" \n\nEvaluation of Postfix expression: ";
        eval_postfix(postfix);

        break;
    }
}while(ch!=3);
return 0;
}

```

```
C:\Users\tejas\OneDrive\Documents\Infix-Postfix_1.exe

Menu :
1.Infix-Prefix-evaluation
2.Infix-Postfix-evaluation
3.Exit
Enter the choice:1

Enter the infix expression:
A*(B+C)-D/E

Entered infix is...A*(B+C)-D/E

Entered infix is...E/D-(C+B)*A

Stack is empty now!!

Prefix expression is: -*A+BC/DE

Evaluation of Prefix expression:
Enter the value of E:5

Enter the value of D:10

Enter the value of C:4

Enter the value of B:3

Enter the value of A:2
```

```
C:\Users\tejas\OneDrive\Documents\Infix-Postfix_1.exe

Entered infix is...A*(B+C)-D/E

Entered infix is...E/D-(C+B)*A

Stack is empty now!!

Prefix expression is: -*A+BC/DE

Evaluation of Prefix expression:
Enter the value of E:5

Enter the value of D:10

Enter the value of C:4

Enter the value of B:3

Enter the value of A:2

Value of expression is 12

Menu :
1.Infix-Prefix-evaluation
2.Infix-Postfix-evaluation
3.Exit
Enter the choice:
```

```
C:\Users\tejas\OneDrive\Documents\Infix-Postfix_1.exe
Menu :
1.Infix-Prefix-evaluation
2.Infix-Postfix-evaluation
3.Exit
Enter the choice:2

Enter the infix expression:
A*(B+C)-D/E

Entered infix is...A*(B+C)-D/E

Stack is empty now!!

Postfix expression is: ABC+*DE/-

Evaluation of Postfix expression:
Enter the value of A:2

Enter the value of B:3

Enter the value of C:4

Enter the value of D:10

Enter the value of E:5

Value of expression is 12
```

```
C:\Users\tejas\OneDrive\Documents\Infix-Postfix_1.exe
Entered infix is...A*(B+C)-D/E

Stack is empty now!!

Postfix expression is: ABC+*DE/-

Evaluation of Postfix expression:
Enter the value of A:2

Enter the value of B:3

Enter the value of C:4

Enter the value of D:10

Enter the value of E:5

Value of expression is 12

Menu :
1.Infix-Prefix-evaluation
2.Infix-Postfix-evaluation
3.Exit
Enter the choice:3

-----
Process exited after 428.1 seconds with return value 0
Press any key to continue . . .
```