Evaluation of Postfix

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

struct Node
{
    int data;
    struct Node *next;
}*top=NULL;


void push(int x)
{
    struct Node *t;
    t=(struct Node*)malloc(sizeof(struct Node));

    if(t==NULL)
        printf("stack is full\n");
    else
    {
        t->data=x;
        t->next=top;
        top=t;
    }

}

int pop()
{
    struct Node *t;
    int x=-1;

    if(top==NULL)
        printf("Stack is Empty\n");
    else
    {
        t=top;
```

```
        top=top->next;
        x=t->data;
        free(t);
    }
    return x;
}

void Display()
{
    struct Node *p;
    p=top;
    while(p!=NULL)
    {
        printf("%d ",p->data);
        p=p->next;
    }
    printf("\n");
}

int isBalanced(char *exp)
{
    int i;

    for(i=0;exp[i]!='\0';i++)
    {
        if(exp[i]=='(')
            push(exp[i]);
        else if(exp[i]==')')
        {
            if(top==NULL)
                return 0;
            pop();
        }
    }
    if(top==NULL)
        return 1;
    else
        return 0;
}
```

```
int pre(char x)
{
    if(x=='+' || x=='-')
        return 1;
    else if(x=='*' || x=='/')
            return 2;
    return 0;
}

int isOperand(char x)
{
    if(x=='+' || x=='-' || x=='*' || x=='/')
       return 0;
    else
       return 1;

}

char * InToPost(char *infix)
{
    int i=0,j=0;
    char *postfix;
    long len=strlen(infix);
    postfix=(char *)malloc((len+2)*sizeof(char));

    while(infix[i]!='\0')
    {
        if(isOperand(infix[i]))
            postfix[j++]=infix[i++];
        else
        {
            if(pre(infix[i])>pre(top->data))
                push(infix[i++]);
            else
            {
                postfix[j++]=pop();
            }
        }
    }
    while(top!=NULL)
```

```
        postfix[j++]=pop();
    postfix[j]='\0';
    return postfix;
}

int Eval(char *postfix)
{
    int i=0;
    int x1,x2,r=0 ;

    for(i=0;postfix[i]!='\0';i++)
    {
        if(isOperand(postfix[i]))
        {
            push(postfix[i]-'0');
        }
        else
        {
            x2=pop();x1=pop();
            switch(postfix[i])
            {
                case '+':r=x1+x2; break;
                case '-':r=x1-x2; break;
                case '*':r=x1*x2; break;
                case '/':r=x1/x2; break;
            }
            push(r);
        }
    }
    return top->data;
}


int main()
{
    char *postfix="234*+82/-";

    printf("Result is %d\n",Eval(postfix));

    return 0;
```

```
}
```