Evaluate arithmetic expression by converting it from infix to postfix.

```
#include <stdio.h>
#include <stdlib.h>
#include<strings.h>
struct Node
{
char data;
struct Node *next;
}*top=NULL;

void push(char 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;
}
}
char pop()
{
struct Node *t;
char 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;
int 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 *infix="a+b*c-d/e";
push('#');
char *postfix=InToPost(infix);
printf("%s ",postfix);
printf("Result is %d\n",Eval(postfix));

return 0;
}
```

Output :

```
cd "/Users/amzamani/Desktop/sem3/DS/ds lab/assgn03/" && gcc
Opost
Abus-MacBook-Air:ds lab amzamani$ cd "/Users/amzamani/Deskto
sktop/sem3/DS/ds lab/assgn03/"infTOpost
abc*+de/-# Result is -13
Abus-MacBook-Air:assgn03 amzamani$
```

Check for balanced parenthesis in an expression.

```
#include <stdio.h>
#include <stdlib.h>
struct Node
{
char data;
struct Node *next;
}*top=NULL;

void push(char 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;
}
}
char pop()
{
struct Node *t;
char 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 main()
{
char *exp="(((a+b)*(c-d)))";
printf("%d ",isBalanced(exp));
return 0;
}
```

Output

//for (((a+b)*(c-d)))

```
cd "/Users/amzamani/Desktop/sem3/DS/ds lab/as
paranthesis
1 Abus-MacBook-Air:assgn03 amzamani$
```

Implement stack using arrays (push & pop).

```c
#include <stdio.h>
#include <stdlib.h>
struct Stack
{
int size;
int top;
int *S;
};
void create(struct Stack *st)
{
printf("Enter Size");
scanf("%d",&st->size);
st->top=-1;
st->S=(int *)malloc(st->size*sizeof(int));
}
void Display(struct Stack st)
{
int i;
```

```
for(i=st.top;i>=0;i--)
printf("%d ",st.S[i]);
printf("\n");
}
void push(struct Stack *st,int x)
{
if(st->top==st->size-1)
printf("Stack overflow\n");
else
{
st->top++;
st->S[st->top]=x;
}

}
int pop(struct Stack *st)
{
int x=-1;
if(st->top==-1)
printf("Stack Underflow\n");
else
{
x=st->S[st->top--];
}
return x;
}
int peek(struct Stack st,int index)
{
int x=-1;
if(st.top-index+1<0)
printf("Invalid Index \n");
x=st.S[st.top-index+1];
return x;
}
int isEmpty(struct Stack st)
{
if(st.top==-1)
return 1;
return 0;
```

```
}
int isFull(struct Stack st)
{
return st.top==st.size-1;
}
int stackTop(struct Stack st)
{
if(!isEmpty(st))

return st.S[st.top];
return -1;
}

int main()
{
struct Stack st;
create(&st);
push(&st,10);
push(&st,20);
push(&st,30);
push(&st,40);
printf("%d \n",peek(st,2));

Display(st);
return 0;
}
```

Output :

```
Enter Size4
30
40 30 20 10
Abus-MacBook-Air:assgn03 amzamani$
```