Experiment No.3
Evaluate Postfix Expression using Stack ADT.
Name: Tejashree Anand Karekar
Roll No:20
Date of Performance:
Date of Submission:
Marks:
Sign:

Experiment No. 3: Evaluation of Postfix Expression using stack ADT

Aim : Implementation of Evaluation of Postfix Expression using stack ADT

Objective:

1) Understand the use of Stack.
2) Understand importing an ADT in an application program.
3) Understand the instantiation of Stack ADT in an application program.
4) Understand how the member functions of an ADT are accessed in an application program

Theory:

An arithmetic expression consists of operands and operators. For a given expression in a postfix form, stack can be used to evaluate the expression. The rule is whenever an operand comes into the string, push it onto the stack and when an operator is found then the last two elements from the stack are popped and computed and the result is pushed back onto the stack. One by one the whole string of postfix expressions is parsed and the final result is obtained at an end of computation that remains in the stack.

Algorithm

```
Step 1: Add a ")" at the end of the postfix expression
Step 2: Scan every character of the postfix expression and repeat Steps 3 and 4 until ")"is encountered
Step 3: IF an operand is encountered, push it on the stack
IF an operator 0 is encountered, then
        a. Pop the top two elements from the stack as A and B as A and B
        b. Evaluate BOA, where A is the topmost element and B is the element below A.
        c. Push the result of evaluation on the stack [END OF IF]
Step 4: SET RESULT equal to the topmost element of the stack
Step 5: EXIT
```

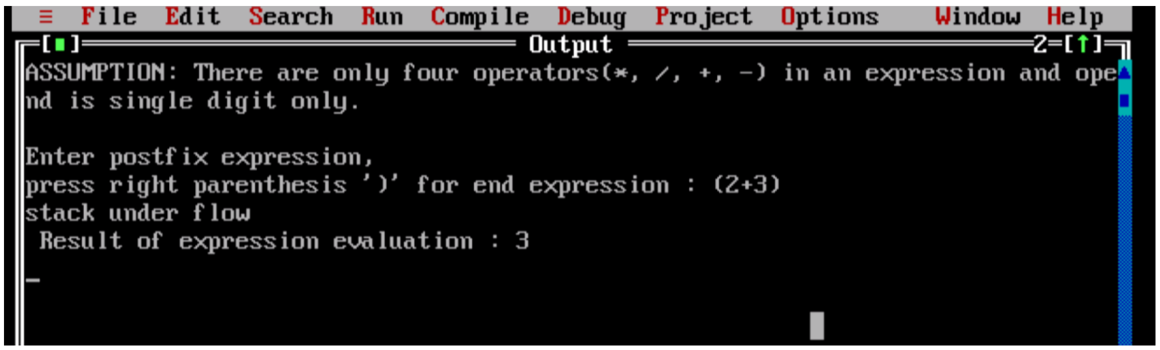
Code:

```
#include <stdio.h>

#include <ctype.h>

#define MAXSTACK 100

#define POSTFIXSIZE 100

int stack[MAXSTACK];

int top = -1;

void push(int item)

{

 if (top >= MAXSTACK - 1) {

 printf("stack over flow");

 return;

 }

 else {

 top = top + 1;

 stack[top] = item;

 }
```

```
}
int pop()
{
 int item;
 if (top < 0) {
 printf("stack under flow");
 }
 else {
 item = stack[top];
top = top - 1;
 return item;
 }
}
void EvalPostfix(char postfix[])
{
 int i;
 char ch;
 int val;
 int A, B;
 for (i = 0; postfix[i] != ')'; i++) {
 ch = postfix[i];
 if (isdigit(ch)) {
 push(ch - '0');
 }
 else if (ch == '+' || ch == '-' || ch == '*' || ch == '/') {
 A = pop();
 B = pop();
 switch (ch)
 {
 case '*':
 val = B * A;
 break;
 case '/':
 val = B / A;
 break;
 case '+':
val = B + A;
 break;
 case '-':
 val = B - A;
 break;
 }
 push(val);
 }
 }
 printf(" \n Result of expression evaluation : %d \n", pop());
}
int main()
```

```
{
 int i;
 char postfix[POSTFIXSIZE];
 printf("ASSUMPTION: There are only four operators(*, /, +, -) in an expression and operand is single
digit only.\n");
 printf(" \nEnter postfix expression,\npress right parenthesis ')' for end expression : ");
 for (i = 0; i <= POSTFIXSIZE - 1; i++) {
 scanf("%c", &postfix[i]);
 if (postfix[i] == ')')
 {
 break;
 }
 }
 EvalPostfix(postfix);
return 0;
}
```

Output:


≡ File Edit Search Run Compile Debug Project Options Window Help
Output

```
ASSUMPTION: There are only four operators(*, /, +, -) in an expression and ope
nd is single digit only.

Enter postfix expression,
press right parenthesis ')' for end expression : (2+3)
stack under flow
 Result of expression evaluation : 3
_
```

Conclusion:

1)Elaborate the evaluation of the following postfix expression in your program.

AB+C-

Push 'A' onto the stack.

Push 'B' onto the stack.

Encounter '+', pop 'B' and 'A', calculate 'A + B' (resulting in 'C'), and push 'C' onto the stack.

Encounter '-', pop 'C' and 'C', calculate 'C - C' (resulting in 0), and push 0 onto the stack.

2)Will this input be accepted by your program. If so, what is the output?

- At this point, the stack contains the final result, which is 0. The program returns this result as the output. When you run the program with the provided expression, it will print "Result: 0".