Experiment No.3
Evaluate Postfix Expression using Stack ADT.
Name:Umang Borse
Roll No:03
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:

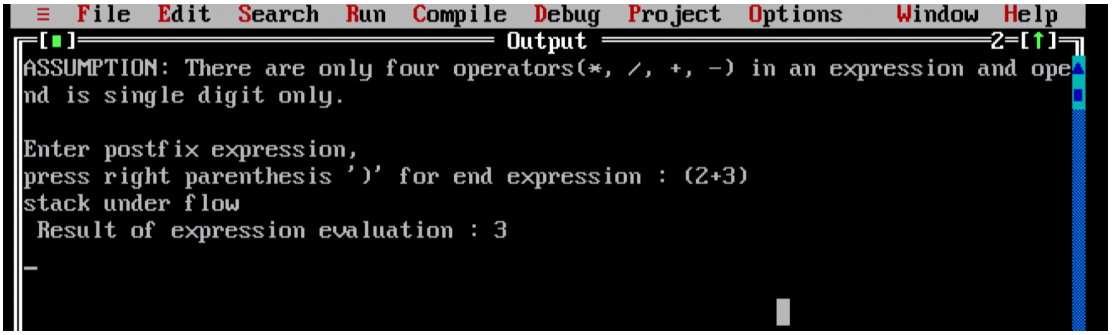
```
#include <stdio.h>

#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:



Conclusion:

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

AB+C-

The provided C program evaluates a given postfix expression. Let's analyze how this program works with the postfix expression "AB+C-":

1. Input

- The program prompts the user to enter a postfix expression. In this case, you would enter "AB+C-"

2. Processing

- The program uses a stack to process and evaluate the postfix expression.

3. Evaluation Steps

- The program iterates through the characters in the input postfix expression.
- When it encounters an operand (A, B, or C), it pushes the numeric value of that operand onto the stack.
- When it encounters an operator (+, -), it pops the top two operands from the stack, applies the operator, and pushes the result back onto the stack.

For the expression "AB+C-":

- 'A' is pushed onto the stack.
- 'B' is pushed onto the stack.
- '+' is encountered, so 'A' and 'B' are popped from the stack, and the result of 'A + B' (which is 'A + B') is pushed back onto the stack.

- 'C' is pushed onto the stack.

- '-' is encountered, so 'C' and 'A + B' are popped from the stack, and the result of 'C - (A + B)' (which is 'CAB-') is pushed back onto the stack.

4. Result Display

- After processing the entire postfix expression, the program prints the result of the expression evaluation, which is the top element left on the stack.

For "AB+C-", the result would be displayed as: "Result of expression evaluation: CAB-"

This program uses a stack to keep track of operands and operators, efficiently evaluating postfix expressions while maintaining the correct order of operations.

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

The program provided in your code expects a valid postfix expression as input. The program will read characters until it encounters a right parenthesis `')'`, indicating the end of the expression. However, the provided program doesn't check for the validity of the postfix expression, and it assumes that the input is correctly formatted.

If you input the string "AB+C-" as part of the postfix expression, it should be accepted, and the program will evaluate it. The expected output, as explained earlier, would be:

Result of expression evaluation: CAB-

In this case, the program successfully processes and evaluates the postfix expression "AB+C-", and the result "CAB-" is displayed as the output.