Experiment No.3
Aim: Implementation of Evaluation of Postfix Expression using stack ADT
Name: Nikhil Kamalaji Shingade
Roll no: 57
Date of Performance:
Date of Submission:

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>
#define MAX 100
int stack[MAX];
int top = -1;
void push(int item)
{
if (top >= MAX -
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;
    }
}
voidEvalPostfix(ch
ar 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());
}
void main()
{
int i;
char postfix[100];
printf("ASSUMPTI
ON: 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 <= 99;
i++) {
   scanf("%c",
&postfix[i]);
if (postfix[i] == ')')
{
break;
   }
}
EvalPostfix(postfix
);
}
```

Output:

```
Output                                                    Clear

/tmp/EgSsZvUzeE.o
ASSUMPTION: There are only four operators(*, /, +, -) in an expression and
    operand is single digit only.

Enter postfix expression,
press right parenthesis ')' for end expression : 52+62-*)

 Result of expression evaluation : 28


=== Code Exited With Errors ===
```

Conclusion:

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

AB+C-

ANS:

S.R.N0.	SYMBOLS	STACK	EVALUATION
1	A	A	
2	B	AB	
3	+	A+B	A+B
4	C	A+BC	
5	-	A+B-C	A+B-C
6			

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

ANS:

Yes, the input AB+C- will be accepted by the program for evaluation. The output will depend on the values assigned to A, B, and C.

Here's the step-by-step evaluation:

1. **Push** A.
2. **Push** B.
3. **Push** C.
4. **Encounter** +: Pop B and C, add them, and push the result.
5. **Encounter** -: Pop A and the result of B + C, subtract the result from A.

If we assign values:

- A = 1
- B = 2
- C = 3

The evaluation will be: \[1 - (2 + 3) = -4\]

So, the output will be -4.