	PUNE INSTITUTE OF COMPUTER TECHNOLOGY PUNE - 411043	
	Department of Electronics & Telecommunication	
	ASSESMENT YEAR: 2021-2022	CLASS: SE-5
	SUBJECT: DATA STRUCTURES	
EXPT No:	LAB Ref: SE/2021-22/	Starting date: 1/11/2021
	Roll No: 22108	Submission date:8/11/2021
Title:	**Evaluate Postfix**	
Problem statement	Evaluate postfix expression (input will be postfix expression).	
Prerequisites:	Basics of C programming	
	Decision making and loop controls	
	Data Structures, Stack	
	Postfix, Infix, Prefix Expressions	
Objectives:	Evaluate a postfix expression	
Theory:		

Stack –

- It is an ordered group of homogeneous items of elements.
- Elements are added to and removed from the top of the stack (the most recently added items are at the top of the stack).
- Elements in between the stack cannot be removed or element cannot be added between the stack, for that all the elements above it needs to be removed and stored separately and then finally we will be able to 'pop' (remove/delete permanently) or 'push' (add element).
- The last element to be added is the first to be removed (LIFO: Last In, First Out).

Postfix Expression –

- A postfix expression is a collection of operators and operands in which the operator is placed after the operands. That means, in a postfix expression the operator follows the operands.
- For Example –
  a+b is an Infix expression,
  ab+ is a postfix expression
- A postfix expression is evaluated by scanning it form left to right.

To evaluate a postfix expression, stack is used to store Operands and perform operation.

Algorithm	1) Start 2) Declare a string variable exp[20] with a postfix expression stored in it and integer variable num, n1, n2 , n3. 3) declare *ch character pointer and ch=exp; 4) While(*ch!=‘\0’) 5) Check if *ch is a digit 6) Num = *ch - 48; 7) Push num to stack 8) Else 9) Pop 2 values from stack and hold in operands in n2, n1; 10) Switch case Case ‘+’: n3 = n1 + n2; break; Case ‘-’: n3 = n1 - n2; break; Case ‘*’: n3 = n1 * n2; break; Case ‘/’: n3 = n1 / n2; break; 11) End switch 12) End Else 13) Push n3 in stack 14) increment ch by 1 15) End of while loop 16) End
ERROR and REMEDY	-

Code

```
#include<stdio.h>
#include<string.h>
#include<ctype.h> int
stack[20];
int top = -1, i;

void push(int val)
{
    if (top >= 19){
        printf("\nStack Overflow");
    }
else
{
        top = top+1;
stack[top] = val;
    }
}

int pop()
{     int val;
if (top<0)
```

```
    {
        printf("\nStack Underflow");
    }
else
{
        val = stack[top];
        top--;
return val;
    }
}

void evaluate_postfix(char postfix[])
{     char *ch;
int a, b, num;
    float c;

    ch = postfix;

    for (i = 0; *ch != '>'; i++)
    {
        if (isdigit(*ch))
        {
            num = *ch-48;
            push(num);
        }
        else if (*ch == '+' || *ch == '-' || *ch == '*' || *ch == '/')
        {
a = pop();
b = pop();

            switch (*ch)
            {
case '+':
            {
c = a+b;
break;
            }
case '-':
            {
c = a-b;
break;
            }
case '*':
```

```
            {
c = a*b;
break;
            }
case '/':
```

```
            {
                if (a == 0)
                {
                    printf("\nError while performing division - Divide by Zero is not
possible!");
                    return;
                }
c = b/a;
break;
            }
}
            push(c);
        }
ch++;
    }
    printf("\nValue of expression = %d", pop());
}
int main()
{
    char postfix[20];
    printf("\nEnter the postfix expression and add '>' at the end indicating the
end of the expression: ");     scanf("%s", &postfix);
    printf("So here postfix expression is '%s'", postfix);

    evaluate_postfix(postfix);

    return 0;
}
```

Output

```
Enter the postfix expression and add '>' at the end indicating the end of the
expression: 123+*45-/>
So here postfix expression is '123+*45-/>'
Value of expression = 5
```

CONCLUSION:

Implemented Postfix Evaluation Algorithm and verified our result.

REFERENCES:

Seymour Lipschutz, Data Structure with C, Schaum’s Outlines, Tata McGrawHill

	E Balgurusamy - Programming in ANSI C, Tata McGraw-Hill (Third Edition)
	Yashavant Kanetkar- Let Us C, BPB Publication, 8th Edition.

Continuous Assessment for DS AY 2021-22			
RPP (5)	**SPO (5)**	**Total (10)**	**Signature:**
			Assessed By: Mr. V. B. Vaijapurkar
Start date	**Submission date**		**Date:**
1/11/2021	**8/11/2021**		**Roll. No.22108**
***Regularity, Punctuality, performance** ***Submission, Presentation, orals**			