DATA STRUCTURES LAB EXPERIMENTS

SANSKRUTI SAMANT

USN: 2GI17EC113

DIV: B

# **EXPERIMENT 2 a:** Evaluation of postfix expression using C.

* **PROBLEM STATEMENT**: Write a C program to evaluate a Postfix Expression, using stack.
* **ALGORITHM:**

1) Create a stack to store operands (or values).

2) Scan the given expression and do following for every scanned element.

a) If the element is a number, push it into the stack

b) If the element is an operator, pop operands for the operator from stack. Evaluate the operator and push the result back to the stack

3) When the expression is ended, the number in the stack is the final answer.

# **PROGRAM CODE:**

# #include<stdio.h>

# #include<math.h>

# #include<ctype.h>

# #define stacksize 100

# #define exprsize 100

# int stack[stacksize];

# int top=-1;

# void push(int);

# int pop(void);

# int evaluate\_postfix(char[]);

# int main()

# {

# char postexpr[exprsize];

# int result;

# printf("\n Enter the valid postfix expression : \n ");

# gets(postexpr);

# result=evaluate\_postfix(postexpr);

# printf("\n\n the evaluation of post fix expression = %d",result);

# return 0;

# }

# void push(int item)

# {

# if(top==stacksize-1)

# printf("\n stack overflow!!!");

# else

# {

# top++;

# stack[top]=item;

# }

# }

# int pop(void)

# {

# int item;

# if(top==-1)

# {

# printf("\n stack is empty !!");

# exit(0);

# }

# else

# {

# item=stack[top];

# top--;

# }

# return(item);

# }

# int evaluate\_postfix(char postexpr[])

# {

# char ch;

# int op1,op2,i,res;

# for(i=0;postexpr[i]!='\0';i++)

# {

# ch=postexpr[i];

# if(isdigit(ch))

# push(ch-'0');

# else if(ch=='+'||ch=='-'||ch=='\*'||ch=='/')

# {

# op2=pop();

# op1=pop();

# switch(ch)

# {

# case '+':res=op2+op1;

# break;

# case '-':res=op1-op2;

# break;

# case '\*':res=op2\*op1;

# break;

# case '/':res=op1/op2;

# break;

# }

# push(res);

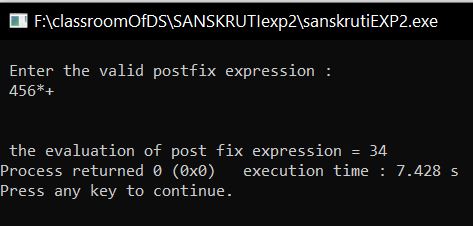
# }

# }

# return(res);

# }

* **PROGRAM OUTPUT:**

****

* **ANALYSIS (LIMITATIONS):**

There are following limitations of above implementation.

1. It supports only 4 binary operators ‘+’, ‘\*’, ‘-‘and ‘/’. It cannot be extended for more operators by adding more switch cases.
2. The allowed operands are only single digit operands. The program can be extended for multiple digits by adding a separator like space between all elements (operators and operands) of given expression.