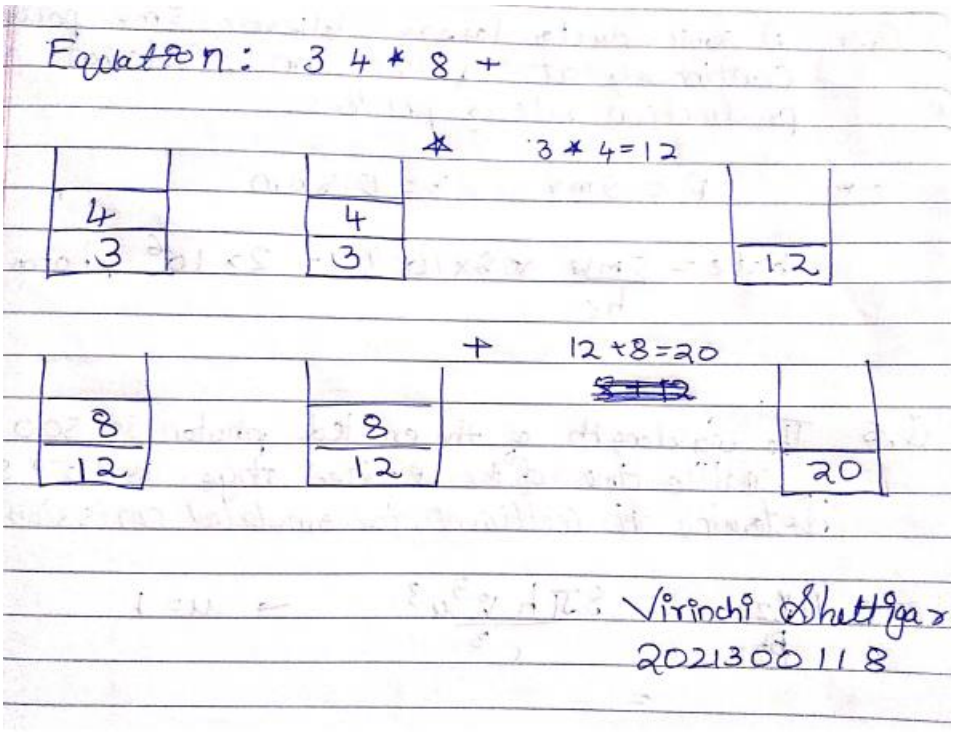
Name	**Virinchi Sadashiv Shettigar**
UID no.	**2021300118**
Experiment No.	1

AIM:	Evaluation of Postfix Expression using Stack
Program 1	
THEORY:	1. Stack: Stack is an abstract Data type, commonly used in programming languages. It is named stack as it behaves in the real world. Stack is a Last In First Out (LIFO) Data Structure. In stack terminologies, insertion is called a push operation and deletion is called a pop operation. 2. Polish and Reverse Polish Notation: Polish notation (PN), also known as normal Polish notation (NPN), Łukasiewicz notation, Warsaw notation, Polish prefix notation, or simply prefix notation, is a mathematical notation in which operators precede their operands, in contrast to the more common infix notation, in which operators are placed between operands, as well as reverse Polish notation (RPN), in which operators follow their operands. It does not need any parentheses as long as each operator has a fixed number of operands. Polish Notation – Prefix Reverse Polish Notation – Postfix Notation. 3. Evaluation Of PostFix: • **Read all the symbols one by one from left to right in the given Postfix Expression** • **If the reading symbol is an operand, then push it onto the Stack.** • **If the reading symbol is operator (+, -, *, / etc.,), then perform TWO pop operations and store the two popped operands in two different variables (operand1 and operand2). Then perform the reading symbol operation using operand1 and operand2 and push the result back onto the Stack.** • **Finally! perform a pop operation and display the popped value as the final result.**

ALGORITHM:	class EvalutaionOfPostfix Main method: 1. Input a postfix equation 2. Convert to a char array 3. Loop char array (repeat 4,5,6) till end of array 4. If arr[i] is a digit push number in stack 5. Else pop and store 2 elements of the stack in 2 variables 6. Push results of operation of 2 variables, operator being stored in arr[i] class intStack: int stack int size int top inSTACK method: 1. Size=100 2. Top=-1(initially) 3. Intialize the stack size to 200 Push method: 1. If stack is not full then increment top and insert element at top 2. Else Stack Overflow Pop method: 1. If stack is not empty then decrement top 2. Else Stack Underflow Peek method: 1. Return the topmost element isEmpty method: 1. Return true if top=-1 or else return false isFull method: 1. Return true if top=size-1 or else return false

PROBLEM-SOLVING:



PROGRAM:

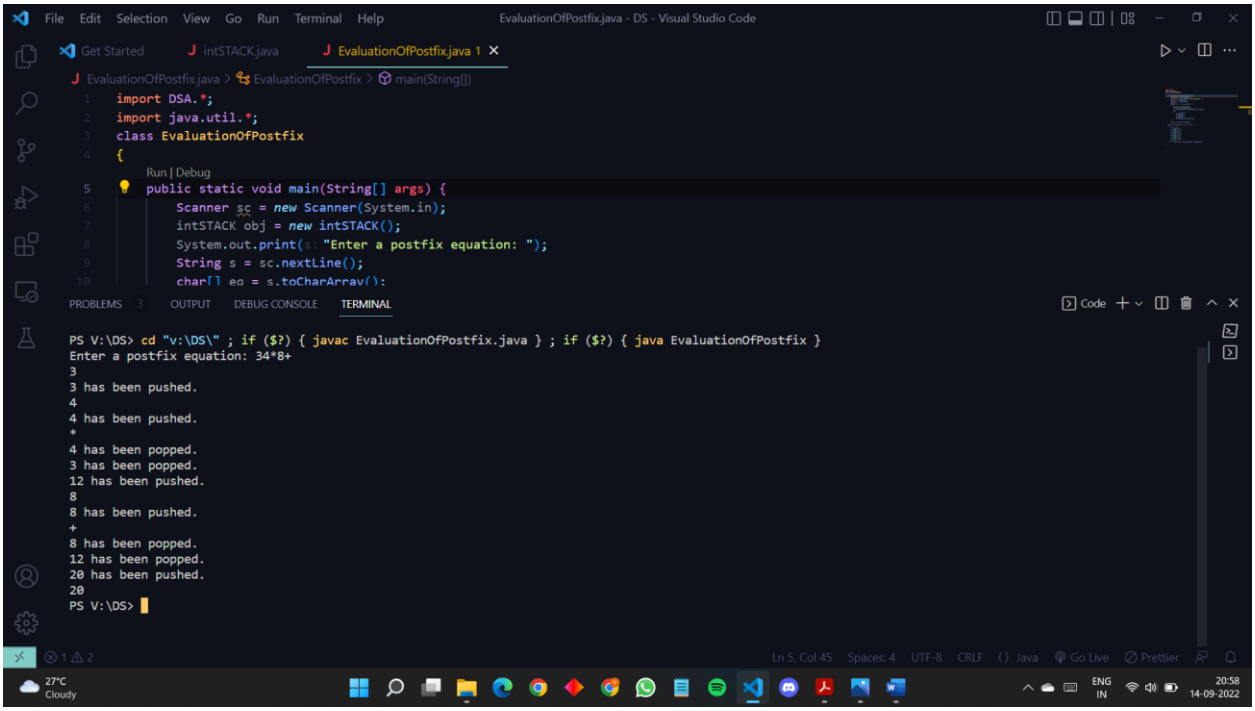
```
package DSA;
public class intSTACK
{
    int stack[];
    int size;
    int top;
    public intSTACK()
    {
        size = 100;
        top = -1;
        stack = new int[size];
    }
    public void push(int n)
    {
        if(!isFull())
        {
            stack[++top] = n;
            System.out.println(n + " has been pushed.");
        }
        else
        {
            System.out.println("STACK OVERFLOW.");
        }
```

```
    }
    public void pop()
    {
        if(!isEmpty())
        {
            System.out.println(peek() + " has been popped.");
            top--;
        }
        else
        {
            System.out.println("STACK UNDERFLOW.");
        }
    }
    public int peek()
    {
        return stack[top];
    }
    public boolean isEmpty()
    {
        return top == -1;
    }
    public boolean isFull()
    {
        return top == size - 1;
    }
}

import DSA.*;
import java.util.*;
class EvaluationOfPostfix
{
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        intSTACK obj = new intSTACK();
        System.out.print("Enter a postfix equation: ");
        String s = sc.nextLine();
        char[] eq = s.toCharArray();
        int a, b;
        for(int i = 0; i < eq.length; i++)
        {
            System.out.println(eq[i]);
            if(Character.isDigit(eq[i]))
                obj.push(Character.digit(eq[i],10));
```

```
            else
            {
                b = obj.peek();
                obj.pop();
                a = obj.peek();
                obj.pop();
                obj.push(operate(a,b,eq[i]));
            }
        }
        System.out.println(obj.peek());
    }
    static int operate(int a, int b, char c)
    {
        if(c == '+')
            return a+b;
        else if(c == '-')
            return a-b;
        else if(c == '*')
            return a*b;
        else if(c == '/')
            return a/b;
        else if(c == '%')
            return a%b;
        else
            return (int) Math.pow((double) a,(double) b);
    }
}
```

OUTPUT:



CONCLUSION:	In this experiment, we learned about the STACK data structure and learned about its basic operations such as push and pop, and the condition of stack empty and stack full. Using these we wrote a program for the Evaluation of Postfix expression and we stored the operands in the stack. According to the equation we push and pop them and got the desired output.