Name	**Hatim Yusuf Sawai**
UID no.	**2021300108**
Experiment No.	**1**

AIM:	**Evaluation of Post-fix expression using Stack**
Program	
THEORY:	**STACK:** A stack is an ordered list in which insertion and deletion are done at one end, called the top. The last element inserted is the first one to be deleted. Hence, it is called the Last in First out (LIFO). When an element is inserted in a stack, the concept is called push, and when an element is removed from the stack, the concept is called pop. Trying to pop out an empty stack is called underflow and trying to push an element in a full stack is called overflow. **Postfix:** A postfix expression (also called Reverse Polish Notation) is a single letter or an operator, preceded by two postfix strings. Every postfix string longer than a single variable contains first and second operands followed by an operator. **Postfix Evaluation Using Stack:** 1. Scan the Postfix string from left to right. 2. Initialize an empty stack. 3. Repeat steps 4 and 5 till all the characters are scanned. 4. If the scanned character is an operand, push it onto the stack. 5. If the scanned character is an operator, and if the operator is a unary operator, then pop an element from the stack. If the operator is a binary operator, then pop two elements from the stack. After popping the elements, apply the operator to those popped elements. Let the result of this operation be returned onto the stack. 6 After all characters are scanned, we will have only one element in the stack. 7 Return top of the stack as result.

ALGORITHM:

Class PostfixEval

Main Method:

1. Input equation as string from user
2. Store length of string in size
3. Convert string to CharArray
4. Initialise **IntStack** Object
5. Int a,b vars for saving popped operands
6. Start for loop: i=0 to i<size:
   a. If char is digit then push into stack
   b. Else If char is '~' then pop top and push with negation
   c. Else:
   d. A = pop()
   e. B = pop()
   f. Switch case to match operators: +,-,*,/,^
   g. Perform operation and push result to stack
7. Display final result stack[top]

Class IntStack

Members:

1. Int[] stack
2. Int capacity
3. Int front,rear

Constructor:

1. Initialize stack array
2. Top = -1

Push Method:

1. Check if stack is full or not, then add element to array

Pop Method:

1. Check if stack is empty or not, then decrease top by 1 and return stack[top--]

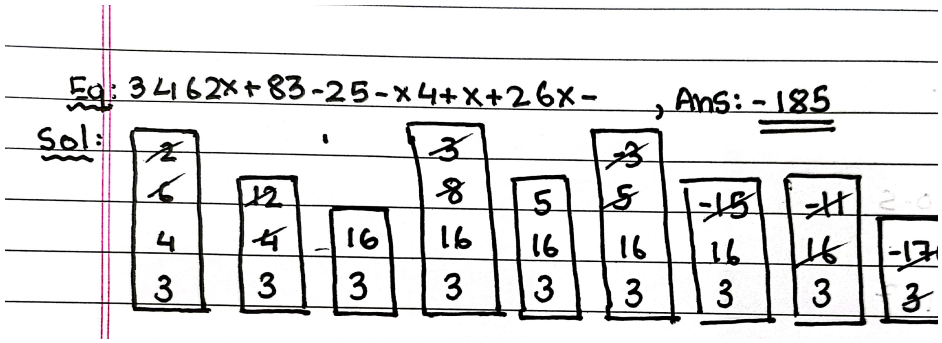
IsEmptyMethod:

1. Return true if top = -1

IsFull Method:

1. Return true if top = capacity-1

PROBLEM SOLVING:



PROGRAM:

PostfixEval.java:

```
import java.util.Scanner;
import java.lang.Math;
import stackds.IntStack;
public class PostfixEval {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter postfix equation: ");
        String exp = sc.nextLine();
        int size = exp.length();
        char[] eq = exp.toCharArray();
        int a,b;
        IntStack s = new IntStack(size);
        try {
            for(int i=0;i<size;i++) {
                if(Character.isDigit(eq[i])) {
                    s.push(eq[i]-'0');
                } else if(eq[i] == '~') {
                    a = s.pop();
                    a = -a;
                    s.push(a);
                } else {
                    b = s.pop();
                    a = s.pop();
                    switch(eq[i]) {
                        case '+':
```

```
                    s.push(a+b);
                    System.out.printf("%d+%d = %d\n",a,b,a+b);
                    System.out.println(s.printStack());
                    break;
                case '-':
                    s.push(a-b);
                    System.out.printf("%d-%d = %d\n",a,b,a-b);
                    System.out.println(s.printStack());
                    break;
                case '*':
                    s.push(a*b);
                    System.out.printf("%d*%d = %d\n",a,b,a*b);
                    System.out.println(s.printStack());
                    break;
                case '/':
                    s.push(a/b);
                    System.out.printf("%d/%d = %d\n", a, b, a / b);
                    System.out.println(s.printStack());
                    break;
                case '^':
                    s.push((int)Math.pow(a,b));
                    System.out.printf("%d^%d = %d\n", a, b, s.Top());
                    System.out.println(s.printStack());
                    break;
                }
            }
        }
        System.out.println("Result: "+s.Top());
    }
    catch(Exception ex) {
        System.out.println(ex.getMessage());
    }
    sc.close();
  }
}
```

IntStack.java:

```
package stackds;
public class IntStack {
   int[] stack;
   int top;
   int capacity;
   public IntStack(int size) {
      stack = new int[size];
      capacity = size;
      top = -1;
   }
   public void push(int e) throws Exception {
      if(isFullStack()) {
         throw new Exception("Stack is full!");
      }
      stack[++top] = e;
   }
   public int pop() throws Exception {
      if(isEmptyStack()) {
         throw new Exception("Stack is empty!");
      }
      return stack[top--];
   }
   public int Top() throws Exception {
      if(isEmptyStack()) {
         throw new Exception("Stack is empty!");
      }
      return stack[top];
   }
   public String printStack() {
      String s = "[";
      if(size()>0) {
         s += stack[0];
      }
      if(size()>1) {
         for(int i=1;i<size();i++) {
```

```
            s += "," + stack[i];
         }
      }
      return s += "]";
   }
   public boolean isEmptyStack() {
      return top == -1;
   }
   public boolean isFullStack() {
      return top == capacity - 1;
   }
   public int size() {
      return top+1;
   }
}
```

OUTPUT:

```
PS D:\Data Structures\Exp2> cd "d:\Data Structu
Enter postfix equation: 3462*+83-25-*4+*+26*-
6*2 = 12
[3,4,12]
4+12 = 16
[3,16]
8-3 = 5
[3,16,5]
2-5 = -3
[3,16,5,-3]
5*-3 = -15
[3,16,-15]
-15+4 = -11
[3,16,-11]
16*-11 = -176
[3,-176]
3+-176 = -173
[-173]
2*6 = 12
[-173,12]
-173-12 = -185
[-185]
Result: -185
PS D:\Data Structures\Exp1>
```

CONCLUSION:	In this experiment, we learned how to implement a stack using an array and use the stack to solve given postfix equations effectively.