PART 3
Problem: **Postfix Machine**
Data structure: **Stack**

Gist:

1. Infix to Postfix Converter
2. **Postfix Machine**
3. Balance Symbol Checker

Use stack to handle the operator precedence

1. Get input: Infix expression.
2. Convert Infix → Postfix.
3. Calculate result: Postfix expression
4. Output result.

Example:

Input infix exp: 4 + 2 * 3

Convert infix-postfix: 4 + 2 * 3 $\rightarrow$ 4 2 3 * +

Calculate postfix exp: = 4 2 3 * +
= 4 6 +
= 10

POSTFIX
MACHINE

Output: 10

Postfix Machine (Postfix Evaluator)

- **Objective** = to evaluate/get the result of the postfix notation
- Why Postfix form?
  - The postfix form represents a natural way to evaluate expressions because precedence rules are not required.
  - The postfix form do not required parenthesis.

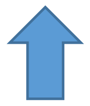
- **Example of Postfix Machine Codes**
  - **Input**: Postfix notation → Example: “2 3 1 * + 9 –”
    (assuming the posfix notation is correct)
  - **Output**: Result of the notation → -4

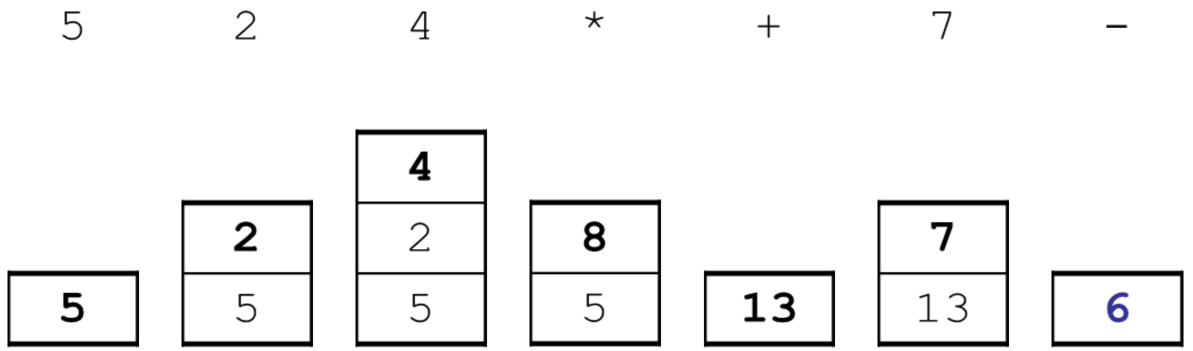
Algorithm for Postfix Machine

1. When an operand is seen, it is pushed onto a stack.
2. When an (binary) operator is seen, the appropriate number (2) of operands are popped from the stack,
3. The operator is evaluated, and the result is pushed back onto the stack.
4. When the complete postfix expression is evaluated, the result should be a single item on the stack that

Example: **Postfix machine**

- Example Expression : 5 2 4 * + 7 –





Codes for Postfix Machine

```
// Java proram to evaluate value of a postfix expression
import java.util.*;
public class PostfixMachine {
   static int evaluatePostfix(String exp) {
         Stack<Integer> stack=new Stack<>(); //create a stack

         for(int i=0;i<exp.length();i++) { //Scan all characters one by one
            char c=exp.charAt(i);
            if(Character.isWhitespace(c))
               ;
            else if(Character.isDigit(c))
                 stack.push(c - '0');                                              (B)
            else {
                int val1 = stack.pop();
                int val2 = stack.pop();

                switch(c) {
                    case '+': stack.push(val2+val1); break;
                    case '-': stack.push(val2- val1); break;                        (C)
                    case '/': stack.push(val2/val1); break;
                    case '*': stack.push(val2*val1); break;
               }
          }
      }  return stack.pop();
   }

   public static void main(String[] args) {
          String InStr="2 3 1 * + 9 -";                         (A)

          System.out.println("Postfix Notation: "+ InStr);
          System.out.println("Result: "+evaluatePostfix(InStr));    (D)
   }
}
```

Demonstration