Exp-04

Implementation of Syntax Analysis: Infix, postfix, prefix notations

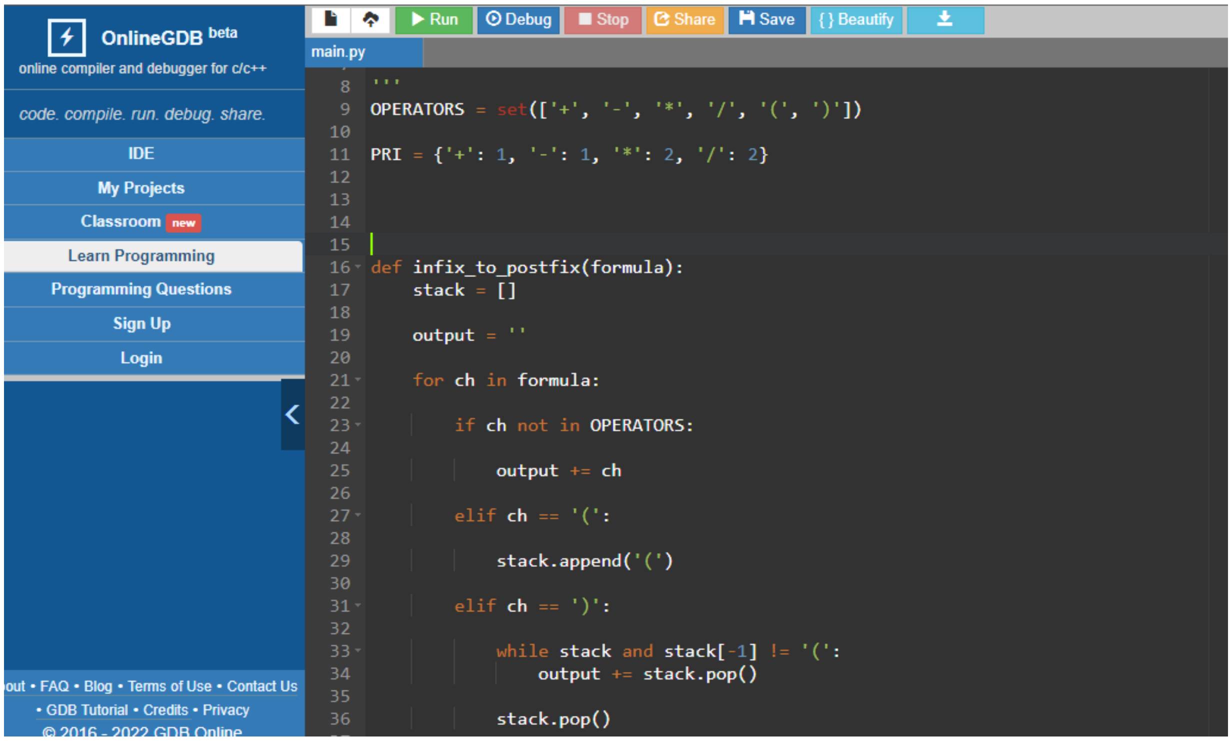
NAME: Ayush Jindal
REG NO: RA1911003010308
SECTION : E1

AIM:

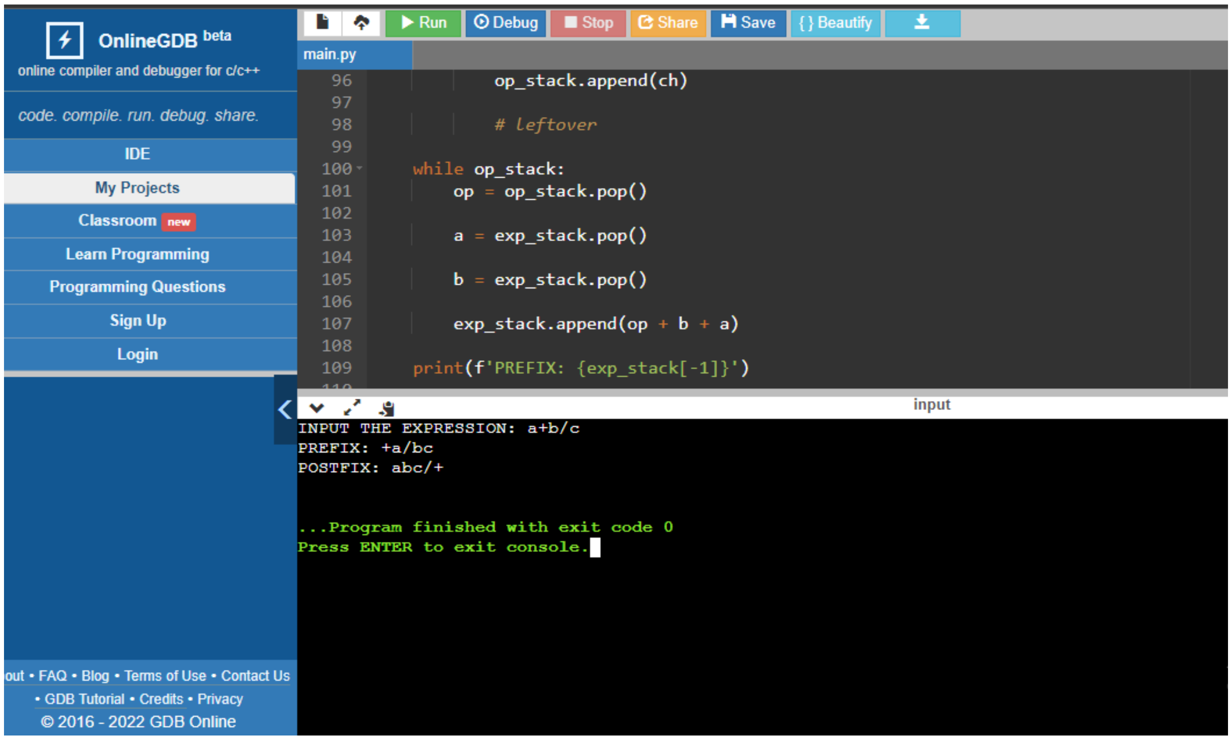
Implementation of Postfix, Prefix, Infix notation

Source Code:

```
OPERATORS = set(['+', '-', '*', '/', '(', ')'])

PRI = {'+': 1, '-': 1, '*': 2, '/': 2}



def infix_to_postfix(formula):
   stack = []

   output = ''

   for ch in formula:

       if ch not in OPERATORS:

           output += ch

       elif ch == '(':

           stack.append('(')
        elif ch == ')':

            while stack and stack[-1] != '(':
                output += stack.pop()

            stack.pop()

        else:

            while stack and stack[-1] != '(' and PRI[ch] <= PRI[stack[-1]]:
                output += stack.pop()

            stack.append(ch)



    while stack:
        output += stack.pop()

    print(f'POSTFIX: {output}')

    return output



def infix_to_prefix(formula):
    op_stack = []

    exp_stack = []

    for ch in formula:

        if not ch in OPERATORS:

            exp_stack.append(ch)

        elif ch == '(':

            op_stack.append(ch)

        elif ch == ')':

            while op_stack[-1] != '(':
                op = op_stack.pop()

                a = exp_stack.pop()

                b = exp_stack.pop()

                exp_stack.append(op + b + a)
            op_stack.pop()
        else:

            while op_stack and op_stack[-1] != '(' and PRI[ch] <= PRI[op_stack[-1]]:
                op = op_stack.pop()

                a = exp_stack.pop()

                b = exp_stack.pop()

                exp_stack.append(op + b + a)

            op_stack.append(ch)



    while op_stack:
        op = op_stack.pop()

        a = exp_stack.pop()

        b = exp_stack.pop()

        exp_stack.append(op + b + a)

    print(f'PREFIX: {exp_stack[-1]}')

    return exp_stack[-1]

expres = input("INPUT THE EXPRESSION: ")

pre = infix_to_prefix(expres)

pos = infix_to_postfix(expres)
```



```
'''
OPERATORS = set(['+', '-', '*', '/', '(', ')'])

PRI = {'+': 1, '-': 1, '*': 2, '/': 2}



def infix_to_postfix(formula):
    stack = []

    output = ''

    for ch in formula:

        if ch not in OPERATORS:

            output += ch

        elif ch == '(':

            stack.append('(')

        elif ch == ')':

            while stack and stack[-1] != '(':
                output += stack.pop()

            stack.pop()
```



```
            op_stack.append(ch)

        # leftover

    while op_stack:
        op = op_stack.pop()

        a = exp_stack.pop()

        b = exp_stack.pop()

        exp_stack.append(op + b + a)

    print(f'PREFIX: {exp_stack[-1]}')
```

```
INPUT THE EXPRESSION: a+b/c
PREFIX: +a/bc
POSTFIX: abc/+


...Program finished with exit code 0
Press ENTER to exit console.
```

RESULT:

Successfully implemented postfix, prefix, infix notations.