Practical 6 Compiler Construction
20BCE057
Devasy Patel

Aim: Intermediate Code Generation: To generate Three-Address code for assignment statement.

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]

def generate3AC(pos):
    print("### THREE ADDRESS CODE GENERATION ###")
    exp_stack = []
    t = 1
    for i in pos:
        if i not in OPERATORS:
            exp_stack.append(i)
        else:
            print(f't{t} := {exp_stack[-2]} {i} {exp_stack[-1]}')
            exp_stack=exp_stack[:-2]
            exp_stack.append(f't{t}')
            t+=1

print("Welcome to Simple Three Address Code Generation by 20BCE057.")
```

```
expres = input("Please input the expression you would like to generate the
code for >")
pre = infix_to_prefix(expres)
pos = infix_to_postfix(expres)
generate3AC(pos)
```

Output:

```
@Devasy23 →/workspaces/College (master) $ python SEM7/CC/p6.py
Welcome to Simple Three Address Code Generation by 20BCE057.
Please input the expression you would like to generate the code for >(A*(B+D*Y)+Z)
PREFIX: +*A+B*DYZ
POSTFIX: ABDY*+*Z+
### THREE ADDRESS CODE GENERATION ###
t1 := D * Y
t2 := B + t1
t3 := A * t2
t4 := t3 + Z
@Devasy23 →/workspaces/College (master) $ python SEM7/CC/p6.py
Welcome to Simple Three Address Code Generation by 20BCE057.
Please input the expression you would like to generate the code for >A=B+D*(C)
PREFIX: +B*DC
POSTFIX: A=BDC*+
### THREE ADDRESS CODE GENERATION ###
t1 := D * C
t2 := B + t1
```